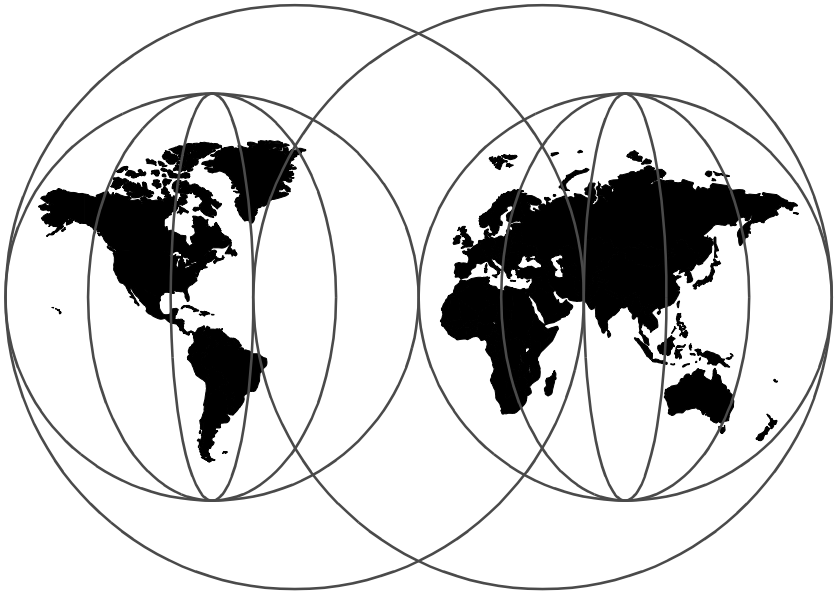
Tivoli

IBM

Tivoli Enterprise Internals and Problem Determination

Richard Hawes, Victoria Stevens, Bob Cashion, Rhonda Childress, Gary Louw, Morten Moeller



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2034-01

SG24-2034-01

International Technical Support Organization

Tivoli Enterprise Internals and Problem Determination

March 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 689.

Second Edition (March 1999)

This edition was written between November 1998 and February 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 Enterprise Framework and applications. Where information is very specific to release level, a mention of that fact is included in the text. This edition completely supercedes the previous version titled *TME 10 Internals and Problem Determination*.

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

If you can discover more about how products work, then you can easily resolve problems when things go wrong and, more importantly, prevent problems in the first place. This redbook provides under-the-covers information and techniques for problem determination and problem source isolation (PD/PSI) in the Tivoli Enterprise Management environment. You will also find many examples of log files, trace, and other output from both normal and failing scenarios. This redbook is based on internal Tivoli training documentation yet expands on that information and includes scenarios that will help those implementing Tivoli solutions perform PD/PSI. This redbook details the Tivoli Framework and the major Tivoli Enterprise applications including TEC, Software Distribution, and Distributed Monitoring.

This material is invaluable to customers, systems integrators, and field service personnel when assisting with problem determination in a Tivoli environment.

This book is divided into three parts. The first part provides introductory material, documentation on product installation including the Software Installation Service (SIS), and details on core Framework components. Chapter 1 contains an introduction to the topics covered elsewhere in this redbook. Part two provides detailed discussions on each of the major Tivoli Enterprise Products that work with the Framework. The third part of the book provides additional information on related topics.

This book's primary function is to be used as a reference tool. However, because each chapter provides a great deal of background information, it would be beneficial to read through this entire redbook.

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.

Richard Hawes is a Senior Tivoli Security Specialist at the International Technical Support Organization, Austin Center. He holds a BSc in Computer Science and writes extensively on Tivoli framework and security issues. Richard is a Tivoli Certified Enterprise Consultant. Before joining the ITSO in early 1997, Richard was based in the UK providing on-site server troubleshooting and problem management across Europe.

Victoria Stevens is a Senior Engineer in level three support at Tivoli Systems, Austin. She has been with Tivoli since joining customer support in 1994. Victoria was the originator of the material upon which this redbook is based, and she continues to teach Tivoli Internals classes to Tivoli Support and Development personnel. Victoria holds a Computer Science degree from Texas A&M University.

Bob Cashion is a Senior Systems Engineer with Perot Systems Corporation, a Tivoli Business Partner based in Dallas, Texas. He has 20 years experience in the Information Technology field, working in application development, operations, systems programming, and systems management. He has more than three years experience working the Tivoli product set and is a Tivoli Certified Enterprise Consultant. His areas of expertise are Framework, RIM, Distributed Monitoring, and TEC. Bob has consulted for several large corporations implementing Tivoli Enterprise environments in the finance and services industries.

Rhonda Childress is an I/T Architect with IBM Global Services Strategic Outsourcing in St. Louis, MO. She has over 15 years experience in the information technology field and has worked in Application Development, System Programming, System Administration, and System Architecture. Rhonda has worked with the Tivoli product suite for over three years architecting Tivoli solutions and leading implementation teams for IBM Global Services. She has also consulted on several US domestic and international IBM Global Services engagements. Her areas of expertise in Tivoli are Architecture planning and implementation, Framework, Software Distribution, and Distributed Monitoring.

Gary Louw is a Senior Systems Engineer with IBM in Ontario, Canada. Gary has over 18 years experience in support, first with hardware, and more recently, with UNIX. For the last two years, he has been providing technical support for Tivoli products. Gary often performs troubleshooting on customer sites. His areas of expertise include the Tivoli Framework, TEC, and the Deployment products as well as a variety of operating system platforms including Windows NT, AIX, OS/2, and NetWare.

Morten Moeller is an Advisory I/T Specialist with IBM, Denmark. He has 10 years of experience in the Distributed Systems Management field designing and implementing solutions to centrally manage workstations and LANs. Morten has been working with the TIvoli products for the past year. Prior to entering the DSM arena, Morten primarily worked with DB2 and application development on the MVS platform. Today, his areas of expertise include Framework, Software Distribution, Inventory, Distributed Monitoring, and Remote Control. Morten has consulted major enterprises spanning form

manufacturing and retail/distribution to I/T outsourcers and financial institutions.

This is the second edition of this publication. The authors of the first edition (February 1998) were:

Richard Hawes, Victoria Stevens, Ana Lina Bernal, Peter Dominke, Simon Moore, and Ivan Zabala.

A project of this scale and technical depth requires time and assistance from many people. We would like to extend our thanks to the following people for their invaluable contributions:

Ben Allums
David Hooks
David Parrish
Elizabeth Bagley
John Pozdro
Kendall Collett
Kurt McBride
Mark Goewey
Mike Hahn
Nancy Ball
Patrick Hykkonen
Roland Reed
Russ Cunningham
Sadu Bajekal
Sean Starke
Terence Quinn
Tim Little
Caroline McDonnell
Tivoli Systems

Rick Rhea
Rusty Myers
Scott Dengler
IBM Global Services

Tivoli Management Product Names

In an effort to eliminate any confusion about the names for Tivoli's expanding line of management products, Tivoli has recently been through a brand naming review. Those already familiar with the products mentioned in this

publication will be used to seeing the names as TME 10 Security Management, TME 10 Distributed Monitoring, and so on.

The new naming convention for these enterprise software management products will replace TME 10 with Tivoli. The new names are Tivoli Security Management and Tivoli Distributed Monitoring. This change may seem trivial, but the consistency comes from more dramatic changes on other products, such as Unison Destiny, which is now Tivoli Output Manager. Most of the products previously referred to as “TME 10 *Name*” are now part of *Tivoli Enterprise*, a collection of products aimed at major system management solutions across large corporations distinct from other areas, such as Tivoli CrossSite and Tivoli Service Desk.

Many Tivoli products current at the time of writing (3.6 and 3.6.1 releases) are still using the older names. However, throughout this publication we have endeavored to use the new names wherever practical, as Tivoli publicity material and future versions will be using the new names. This includes references to the Tivoli Management Agent often referred to in the past as the Lightweight Client Framework (LCF).

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 “ITSO Redbook Evaluation” on page 715 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Part 1. Introduction, Installation, and Framework Components

Chapter 1. Overview

The Tivoli Enterprise Management suite aims to make distributed systems and application management relatively easy. It achieves this through a consistent interface and the use of models, such as management by subscription. While the systems administrator can perform many tasks with relative ease, the code Tivoli and their partners provide to achieve those tasks is extraordinarily complex. With the solid foundation of the Tivoli Management Framework, this complexity can remain largely masked from the administrator. However, with such a sophisticated set of products, there will be occasions when those designing, testing, and implementing Tivoli solutions will encounter situations that are not resolved by reference to product manuals alone.

In problem-solving situations, you need to understand what is going on between the product components, what messages and trace output means, and what extra actions you can take to try to resolve a problem. This book starts with what is probably the most difficult subject: the core of the Tivoli Management Environment, the Tivoli Object Database. One of the reasons applications fit in so well to the Framework paradigm is the object-oriented nature of Tivoli Enterprise. No discussion of advanced problem-solving techniques would be complete without an understanding of the database Tivoli maintains for all objects in the management environment. If you are new to objects, methods, and the like, Chapter 2, “Tivoli Object Database Architecture” on page 9 will be difficult reading. However, you should try to gain some level of understanding of what we are trying to describe there. If you are already familiar with a particular Tivoli application, you should find the information in the subsequent product chapters useful even without a clear understanding of Chapter 2.

If there are any problems installing a Tivoli product, then it is not likely to work in the desired manner. Therefore, the next topic we discuss is the installation process. The emphasis in Chapter 3 is the installation of the Framework, but much of the information will be relevant to patches and products since they all use the same or similar installation methods. This is extended to multiple installations through the Software Installation Service described in Chapter 4.

If we had to pick only one message to get across in this book, it would be the importance of regular and appropriate backups. This is especially true when troubleshooting. You may need a good backup to return to, or you may need to take backups before performing some procedure. We talk about the backup process in Chapter 5.

Whatever problem you are investigating, there is a core set of commands and log files that you are likely to use. Before we talk about specific products, we cover general problem determination commands, log files, and procedures in Chapter 6.

Most of the applications make regular use of Tivoli Framework services. In Chapter 7, we take a look at how these services function and the steps you can take if they are not functioning correctly. A lot can depend on getting the right administrative authority, checking in the right place for notices, understanding the implications of interconnected TMRs, and so on.

The introduction of the Tivoli Management Agent has been the most significant innovation in the systems management arena since Tivoli first introduced Framework based management. As an important part of making systems management really scalable, we intended to dedicate a chapter to this topic. However, we found that this topic could fill an entire book. Therefore, we cover only certain problem determination tips throughout this book.

Tivoli recently introduced a new product specifically designed for managing systems outside of a company's firewall, Tivoli CrossSite. There are still many cases when some form of Tivoli Enterprise management will involve firewalls, and because of this, Chapter 8 covers this topic.

The RDBMS Interface Module (RIM) is a component of the Framework. We decided to devote Chapter 9 entirely to RIM for two reasons: Getting it set up correctly can be tricky but is vital for the applications that use it to function correctly. Furthermore, RIM will play a more significant role in the future as more of the Framework and applications make use of it.

This concludes the first part of this book. The second part looks at the Enterprise applications themselves.

In Chapter 10, we start on the core Tivoli applications with a look at Tivoli Software Distribution. Understanding its use of repeaters and file package definitions will help to explain many types of failure. A discussion of Software Distribution is not complete without AutoPack, and we take a detailed look at this feature in Chapter 11.

Tivoli Distributed Monitoring is arguably one of the simpler products to implement in a distributed environment. Nevertheless, like any product that relies on customization, timing dependencies, and other factors, there may be occasions when you will need to know more about items, such as the Sentry engine. Chapter 12 provides this information.

Many of the problems associated with products, such as Tivoli Inventory, can be related to the setup or configuration of a RIM database connection. Chapter 13, “Inventory” on page 431 covers this topic and other aspects of problem-solving one of Tivoli’s most popular applications.

Tivoli User Administration is often implemented with a great deal of customization. In Chapter 14,we discuss a number of techniques you might employ if User Administration is not behaving as you expect.

Although relatively new, Tivoli Security Management is becoming a significant part of the Tivoli Enterprise suite. The product has excellent documentation, particularly in the UNIX arena, but we offer more tips in Chapter 15.

Chapter 16, “Enterprise Console” on page 507 is near the end of the book, but it is by no means the least important. The Tivoli Enterprise Console (TEC) is a cornerstone of Tivoli’s product set working with almost every other Tivoli product. TEC is also perhaps one of the most expandable products in the suite, and this can lead to inconsistencies. TEC could probably fill another book in itself, but we have selected a few key topics to discuss, such as TEC server, Event Console, and Rule Base problems.

Tivoli Output Manager is new to the Tivoli portfolio and is perhaps less well known. For those working with this important addition to the Tivoli product line, Chapter 17, “Tivoli Output Manager” on page 539 will prove invaluable.

Another product being widely deployed in Tivoli Enterprise environments is Tivoli Remote Control. This application is very sensitive to different network configurations; so, understanding more about how it works can help make the most of this product in any environment. We cover Remote Control in Chapter 18.

Since the first edition, this redbook has already grown significantly. We could probably produce several volumes if we tried to cover every Tivoli product, so instead, we have instead concentrated on a core set (covering most of Tivoli Enterprise). However, information used by Tivoli support personnel on other topics can benefit the wider audience of a redbook, and so we conclude with a set of appendices in part 3 on related aspects of Tivoli Enterprise Management. These include a discussion describing how Tivoli utilizes Windows NT, and appendices on database management and database installation.

1.1 Generic Problem Determination Outline

If you start to receive errors, and you have questions about the cause, this generic outline for problem determination may help. You will need to review other chapters of this book (such as Chapter 6, “Commands and Logs for Troubleshooting” on page 131, and in particular Section 6.3.1, “Trace Usage Overview” on page 141) in order to understand what to do at each stage. If you have a scenario that you can re-create, the following is a generic list of steps to perform to gather documentation:

To obtain an overall picture:

1. Do `odadmin odlist` to determine the number of machines and to keep for reference purposes.
2. Do `odadmin` alone to get information, such as the port range restrictions (if any) in place.
3. Do `odadmin environ get` to determine the environment the oserv is using.

To gather data from each suspected machine:

1. Logon as root and as a Tivoli root administrator.

   This helps ensure you are not experiencing authority problems.
2. Do `odadmin trace errors` then `odadmin trace objcalls`.
3. Re-create the problem.

On every involved machine, including the TMR server:

4. Do `odstat -v >odstat.txt`.
5. Do `wtrace -jk $DBDIR >wtrace.txt` (or `%DBDIR%` for Windows NT).
6. Collect above txt files plus `oserv.log` and any useful system logs.
7. Set `odadmin trace off`.
8. Then `odadmin trace errors` to revert to the default of just logging errors.

The trace should help you determine the failing objcall.

1.2 Sources of Additional Information

The primary source for information for a Tivoli customer is their Tivoli or IBM representative. Customers can also register for Tivoli’s support Web site at `http://www.support.tivoli.com`. This site is a rich source of help, frequently asked question (FAQ) lists, and so on. Other external sources of information

include Tivoli user groups - one particularly useful site is that of the Tivoli South West Users Group at `http://www.tivoli.isis2000.com` where you can subscribe to a digest of Tivoli information.

> **Note**
>
> Scripts and commands for problem determination could make matters worse if used incorrectly. Always ensure complete backups have been taken and seek the advice of qualified Tivoli personnel (for example, Support or Tivoli Professional Services) for any significant actions.

Also see Appendix E, “Related Publications” on page 693 regarding other publications. The Tivoli product manuals from Version 3.6 include problem determination information, and there are many redbooks covering most Tivoli products in more detail.

1.3 When Is an Endpoint not an Endpoint?

The terms client, endpoint, and node are often used interchangeably when people talk about machines in a Tivoli environment. To avoid confusion, we define here the terms we use in this publication:

LCF — Abbreviation for Lightweight Client Framework, an old term for what is now referred to as the Tivoli Management Agent (TMA). File names and other references still use `lcf` in some places (such as the TMA daemon, `lcfd`).

Endpoint — In general, usage in this book endpoint means the target of a management operation - typically a TMA endpoint or a managed node (see below). For example, for Tivoli User Administration, an endpoint might include the OS/390 TMA endpoint or a Windows NT server configured as either a TMA endpoint or a managed node. For inventory, an endpoint might be any supported managed node, proxy, or TMA endpoint.

TMA Endpoint — Is a system with the Tivoli Management Agent installed on it - a system that communicates with the TMR through an endpoint gateway. A TMA endpoint can also be a managed node if the managed node code is installed on it. Sometimes, we may refer to the code installed on a TMA endpoint as just the TMA. To

	reduce repetition, we may just use endpoint to mean a TMA endpoint when a TMA endpoint has been identified by the context.
Managed Node	Is a system with the full Tivoli Framework installed. At the time of writing, this could be a Windows NT, UNIX, or OS/2 system (although most applications do not recognize OS/2 as a managed node). Note that the TMR server is also a managed node.
Client	A Managed Node.
PC Managed Node	Is the object maintained on a managed node to manipulate data for a PC on which the PC Agent is installed.
PC Agent	The code on a PC allowing it to be managed from a TMR through a managed node.

Chapter 2. Tivoli Object Database Architecture

This chapter explains Tivoli Enterprise's object-oriented environment. This is crucial to help gain an in-depth understanding of how Tivoli works and to perform advanced problem determination activities without jeopardizing the integrity of the Tivoli management database. Reading this material will also help you become familiar with the terminology that is commonly used when working with Tivoli's platform and applications. This is a very complex subject. If you find this chapter too complex, but have Tivoli application experience, you should find plenty of other information in the later chapters that will still be of use to you.

The definitions we have given are not necessarily 100 percent compliant with the CORBA specifications. Instead, we have tried to relate the various terms used to illustrate the hierarchy and the Tivoli implementation. An existing knowledge of Object Oriented technology will help you to understand this chapter. Otherwise, persevere! You may need to read the chapter through more than once to grasp all the concepts presented.

Much of this chapter is based on information provided in the Tivoli Advanced Development Environment (ADE) documentation. If you want to explore this subject more fully, you should refer to that source.

2.1 The Tivoli Enterprise Management Challenge

Tivoli Enterprise is a suite of distributed systems management products that address the following system management needs in a distributed computing enterprise:

Heterogeneity Runs on many different platforms. System administrators need not be concerned with the machine architecture. The network environment can support multiple architectures, so the management platform must be heterogeneous to reduce administrative complexity.

Interoperability Enables many different platforms to operate together. A system administrator using one machine type can manage resources on other machine types regardless of the architecture. Such interoperability extends heterogeneity, enabling system administrators to seamlessly manage any type of machine from any other type of machine.

Scalability Handles large computing enterprises. Managing networks comprising thousands of nodes can produce serious

difficulties for system administrators. Using Tivoli Managed Regions (TMRs) system administrators can easily distribute changes (such as creating a new user) to large networks.

Distributed Provides services across distributed systems, spreading the systems management work load. Not only do managed nodes maintain their own object databases, applications share the management burden by allowing major tasks to be handled by separate machines. Examples include TEC and RIM servers and Distributed Monitoring engines.

Robust APIs Enables all products and customer-developed applications to work together and leverage standard APIs.

Dependability System-management transactions ensure consistency and can back out half-completed operations across the network. This can be very important in large distributed environments where multiple administrators can simultaneously perform operations.

CORBA The Object Management Group (OMG) proposed CORBA 1.1 as a standard for all common Object Request Broker (ORB) systems. Tivoli Enterprise is compliant with this standard.

2.2 Tivoli Enterprise Architecture Overview

The Tivoli Enterprise applications all share a common framework, the Tivoli Framework. The Tivoli Framework is an open, object-oriented framework that includes a set of managers, brokers, and agents that conform with the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) specifications.

This technology allows major differences between computer operating systems to be hidden from the Tivoli user and, to some extent, the applications. It allows key services to be encapsulated in objects that can be used by multiple management applications. The Tivoli Framework provides platform-independence, a unifying architecture for all applications, and the ability for third-party vendors to easily adapt their offerings or plug them into the framework, allowing systems administrators to manage a wide variety of IT resources in a consistent way. In addition, a robust set of APIs and services enables customers to write their own applications that plug into and leverage the Tivoli Framework.

Figure 1 illustrates the Tivoli product structure.

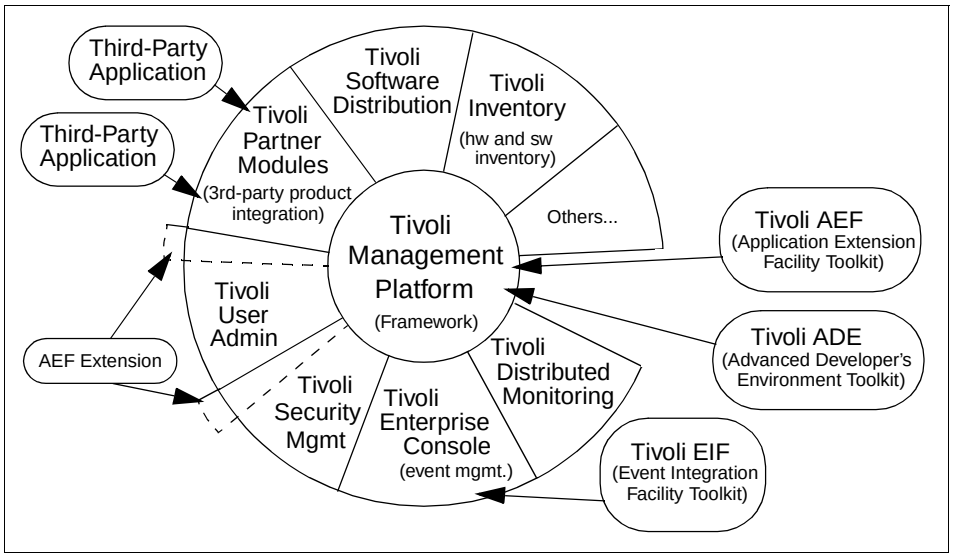


Figure 1. Tivoli Management Environment Software Components

Tivoli Enterprise represents several major advancements for managing large networks of heterogeneous, distributed systems. It has two primary components: a comprehensive management platform (Tivoli Framework), and a set of X/Open-compliant APIs.

The Tivoli Framework is built around an implementation of the OMG CORBA 1.1 environment. It also provides an implementation of the enabling services adopted by X/Open as the basis for a systems management framework.

2.2.1 About CORBA 1.1

The Object Management Group (OMG) is a non-profit, international association of more than 300 companies. Its goal is to define an architectural object framework through a series of detailed interface specifications.

OMG's CORBA specification introduces the Interface Definition Language (IDL) and the concepts of an object request broker (ORB) and basic object adaptor (BOA). The ORB and BOA provide a mechanism for invoking objects and returning the results to requestors.

CORBA 1.1 specification presents an open system of service requestors and service providers in which the requestors are isolated from the providers. Requests are initiated without regard to the location or implementation of the

service provider. The service provider could be on the same machine or on another system of a different architecture somewhere across the network.

CORBA 1.1 specifies interfaces to a set of low-level object services. It does not, however, specify implementation, security, or installation. Nor does it offer a means for multi-vendor ORB interoperability or C++ language bindings. These are up to the implementor to determine and, as in the case of the Tivoli Framework, can significantly enhance the function of a CORBA-based product. The architecture uses three concepts to achieve the integration of a wide variety of object services in many different languages and systems:

Object encapsulation: The object providing a requested service does so within its own context, which means that each object has the ability to respond differently to the same request. Thus, two different objects can support the same interface, and each can maintain a different implementation of that interface.

Complete service requestor/provider isolation: Allows service requestors to make requests of a provider regardless of the provider location or implementation. A service request includes a service identifier (operation name), a provider identifier (object reference), and other optional data.

Interface and implementation separation: Interfaces are defined without regard to the way in which they are implemented.

Within the CORBA 1.1 architecture, there are three primary components: The client, the object implementation, and the ORB/BOA.

The client is the requestor of a service that an object implementation provides. The ORB delivers the request from the client to the object implementation through the BOA. The object implementation then performs the requested service, and any return data is delivered back to the client. Note that the client and object may or may not reside on the same physical computer system.

The client and the object implementation are isolated from each other. Neither has any knowledge of the other except through their interfaces to the ORB and BOA. Client requests are independent of the object implementation location and the programming language in which they are implemented.

Furthermore, clients and object implementations are not capable of direct communication (clients can only initiate requests, and object implementations can only provide services at the request of a client). Figure 2 on page 13

shows the steps involved when a client requests an operation of some object implementation. The request is shown as step 1.

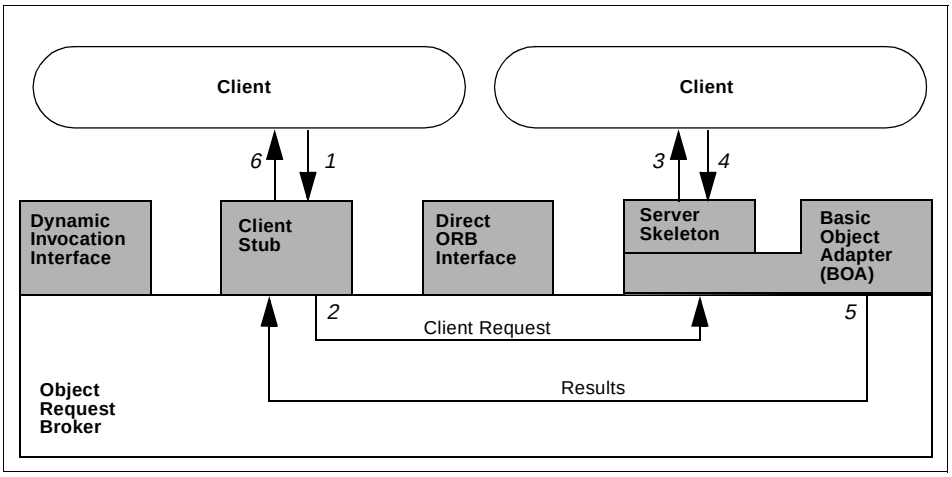


Figure 2. CORBA Operation Request

The ORB delivers the request to the BOA (step 2) that activates the process under which the object implementation runs. The BOA then invokes the method associated with the request by way of the server skeleton (step 3).

When the method is completed, the BOA manages the termination of the method (step 4) and coordinates the return of any results to the client (steps 5 and 6). Alternatively, if a request is unknown until run-time, the Dynamic Invocation Interface (DII) is used to build a request that is used in place of a client stub linked at compile time.

2.2.2 Tivoli Enterprise CORBA Implementation

On top of the CORBA ORB and the enabling services are a set of management services, user interface services, and advanced application services. These combined services form the application programming interface to which systems management applications are written and make up the Tivoli Advanced Development Environment (ADE). If you really want to understand methods and the Tivoli implementation, it is recommended that you spend some time reviewing the ADE documentation and keep it handy (all 16+ books!) for reference purposes.

Figure 3 on page 14 identifies the application interfaces and their relationships to each other:

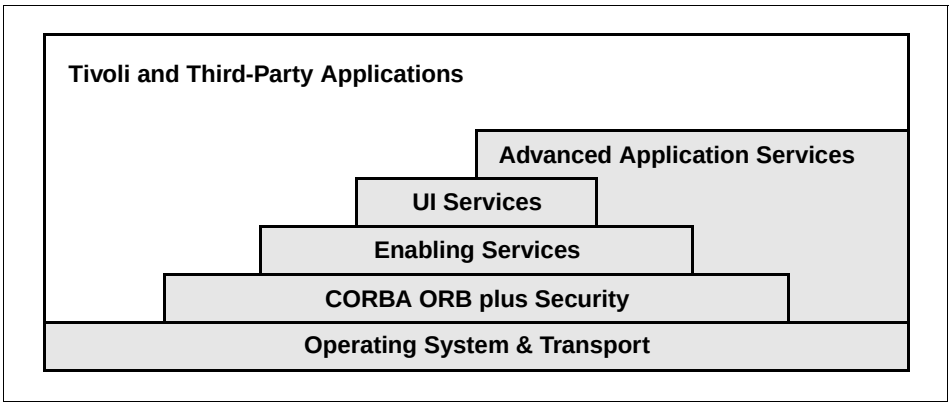


Figure 3. Tivoli Enterprise Application Interfaces

You can choose to write to one or more layers from this API or substitute alternate services and libraries from third parties as appropriate. Tivoli provides support for the same programming interfaces across all Framework-supported architectures, which provides a significant portability layer. Currently, some of these interfaces are better documented than others. Tivoli continues to strive to be a genuinely open platform and API documentation gets better with every revision.

2.2.3 Tivoli Enterprise Heterogeneity and Interoperability

Tivoli Enterprise supports heterogeneous systems management. This means that the Tivoli server can be any supported architecture type, and that Tivoli clients can be any mix of supported architecture types.

In heterogeneous networks, applications have traditionally been required to explicitly cope with the data requirements of each platform. Each time the application transmits data, it must be able to convert the data from the native format to the destination format.

The Tivoli Framework ORB removes this consideration from application programming. When a request to run an operation on some object is made, a client stub initiates the request, collects the data associated with the request, and converts the data from its current format to a common format.

This process (known as data marshalling) is performed in accordance with the ASN.1 standard. ASN.1 converts data to a canonical or simplistic form for transmission to a machine of an undetermined platform type.

When the data conversion is complete, the client stub passes the marshalled data to the ORB. The ORB then sends the data to the BOA and ultimately to the appropriate server skeleton. The server skeleton then reformats the data according to the requirements of the destination object implementation.

2.2.4 Management Services

The system management framework fits into the X/Open reference model and is built on top of an OMG CORBA 1.1 foundation. It provides a set of enabling management services for applications.

These services include policy, extensibility, scheduling, collections, and instance management. When used with other interfaces and services found in Tivoli, they enable the development of sturdy, feature-rich systems management applications.

The following figure illustrates the X/Open reference model and shows the management services component indicating those areas that the Tivoli Framework targets.

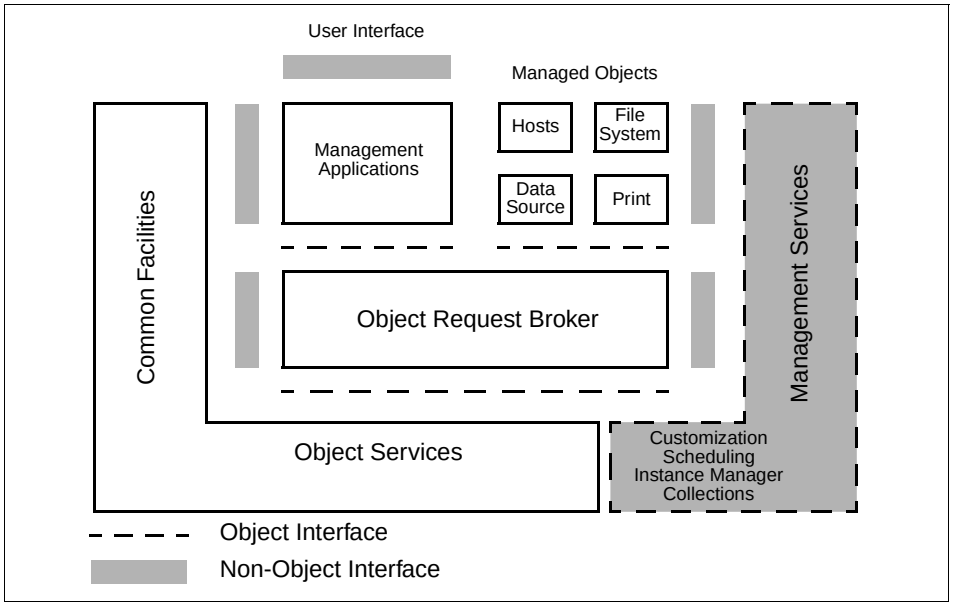


Figure 4. Management Services in the X/Open Reference Model

TME10 specifically focuses on managing policy-driven objects. This management includes the mechanisms and facilities that enable the establishment and enforcement of policy on these objects.

2.3 Tivoli Object Architecture Implementation

Having looked at the basics of the CORBA ORB model, we can start to look in more detail at the Tivoli implementation of an object-oriented management environment

2.3.1 Tivoli Object Request Broker

CORBA 1.1 only specifies the architecture of the ORB model and the provision of the ORB interfaces but not the ORB implementation. The ORB could, for example, be implemented as part of an operating system or as a stand-alone process.

The Tivoli Framework provides object, management, and desktop services and includes an implementation of the APIs adopted by X/Open for a systems management framework.

The Tivoli object dispatcher (oserv) is the main component of the Tivoli Framework run-time. It is implemented as a single multi-threaded process and runs on each TME client within a TMR and on the Tivoli server for that TMR (commonly referred to as the TMR server). The object dispatcher consists of an ORB, the BOA, and related services. The object dispatcher running on the TMR server provides additional services, including security, such as administrator authentication, ORB and object location resolution and implementation inheritance resolution. You may see the acronym ALI (Authentication, Location, and Inheritance) referring to the TMR server. The Tivoli ORB is a continually-running program separate from the operating system.

The Tivoli ORB communicates with the server and the client through separate stubs and skeletons through an inter-process communication facility. A secure remote procedure call (RPC) used to invoke operations on remote objects provides principal authentication and authorization.

The framework services support all the major CORBA 1.1 components:

- An ORB
- A BOA
- A Tivoli-extended IDL compiler (TEIDL) with both ANSI C and Bourne shell language bindings
- An interface repository
- The interfaces required for a Dynamic Invocation Interface (DII)

2.3.2 Tivoli Authorization Principals

Tivoli maintains a database of principal privileges and defines a simple means of controlling which principals have access to which methods. For authorization purposes, a TME principal is a Tivoli administrator or user with an entry in the Tivoli database of principals. If Kerberos is in use, the Tivoli principal is a Kerberos principal name. For more information about Kerberos in Tivoli see the *Tivoli Framework Planning and installation Guide*. If Kerberos is not in use, the principal name has the format `user@machinename` for UNIX or `Domain\user@machinename` for Windows NT. You can check the principal of the user you are using by issuing:

```
objcall 0.0.0 o_get_principal
```

When an administrator or a user invokes a method, an operation on an object, Tivoli looks up the principal's name (using the appropriate form). This name is then cross-referenced to a Tivoli administrator, and the roles are checked. A Tivoli administrator is defined to have one or more roles over the objects in one or more object groups.

2.3.2.1 Security Object Groups

To simplify security management, objects created in Tivoli are assigned as members of one or more object groups. A security object group is a named logical entity to which one or more objects belong and over which an administrator is granted one or more roles. Security object groups typically, but not always, align with policy regions in which managed resources are grouped.

The result looks like an access matrix in which a role for a principal can be verified for many resources in one step. The use of object groups simplifies role management by substantially reducing the number of entries that must be changed when a principal's credentials are modified.

2.3.3 Communication between Objects

As already stated, the object dispatcher (oserv) provides communication between objects in Tivoli Enterprise. The Tivoli ORB uses a secure RPC service layered on top of TCP. This provides secure peer-to-peer communication between ORBs when an operation is invoked on a remote object. The secure RPC service is layered on top of TCP/IP using either domain sockets or the transport layer interface (TLI). An application is never aware of the particular protocol in use; it just invokes the operation on an object by calling the client stub (produced by the TEIDL compiler) or using the DII to build a request at run-time.

The following figure depicts the interaction between two ORBs when an object on one machine invokes an operation on an object that resides on a remote machine:

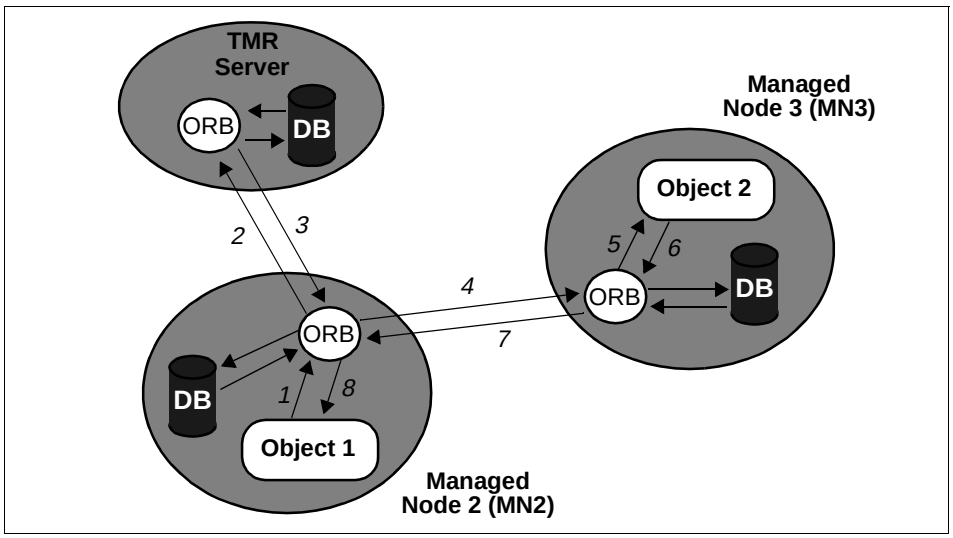


Figure 5. Object Communication across ORBs

A client can run any object method for which a client stub exists locally. The actual location of the object implementation is irrelevant to the invoking client. When a method of object 1 invokes the client stub of a method of object 2, a message is sent to the local ORB. The message specifies the method, object, and arguments of the request (step 1).

The ORB on MN2 communicates with the TMR server to determine whether the principal is authorized to invoke the operation on object 2 (step2). If the principal is authorized to invoke the operation, the server also determines the location of object 2 and resolves any implementation inheritance as necessary.

This information is then returned to the ORB in MN2 in a cryptographically sealed credentials package (step 3). MN2's ORB then forwards the request to MN3's ORB (step 4), which in turn, invokes the desired method of object 2 (step 5).

When the method completes, the results are passed back to MN3's ORB (step 6), which returns them to MN2's ORB (step 7). Finally, the results are delivered to the invoking object (step 8). Database interaction occurs for object query operations or for modifications taking place on object attributes.

As already mentioned, the normal interaction between ORBs is through a secure RPC service layered on top of TCP (using port 94). The next section gives more detail about an additional mechanism used to exchange information between objects.

Note

The ports used by the oserv are selected from the range available. Note that `odadmin set_port_range` can be used to limit the port ranges used by an oserv. The port range can be reset to not restricted using `odadmin set_port_range ""`.

2.3.3.1 Inter-Object Messages

The Tivoli Inter-Object Message (IOM) service provides a connection-oriented bi-directional service between methods of different objects that is independent of the object services layer and operating system. This service is designed for applications that require an asynchronous bi-directional method-to-method communication and transport facility. It is used either when an application explicitly chooses to use it or when the object-to-ODB-to-ODB-to-object mechanism described above results in data transfer of greater than 16 KB.

The IOM service requires participating methods or client programs to be threaded. Using threaded methods is necessary for one of two reasons:

- A thread in the client is required for method invocation.
- A thread in the server is needed for communication after the client has been started.

Refer to Section 7.6.3, “Bulk Data Transfer and Inter-Object Messaging” on page 271 for more information about IOM.

2.3.4 Transactions

In order to provide reliable management operations, Tivoli makes use of transactions. A failure in an operation means the transaction can be rolled back to a known state (automatically) and ensures object database updates remain as consistent as possible.

This section provides some background information about Tivoli transactions. Refer to 6.4.1, “Transaction Log Files and tmstat” on page 172 for more information.

Note

Modifying Tivoli transactions should NOT be used as a routine method of problem determination. Altering transaction data can have serious and irreversible affects on the operation of a TMR. We advise you to use this information to find out more about what is going on in the TMR and use other methods to attempt to rectify problems.

There are three types of transactions:

Top transaction Runs independently of other transactions

Sub transaction Runs as dependent process of parent Top transaction

Revocable transaction Can run as either Top or Sub transaction

Within the boundaries of the context, any object request that is made is executed in the *context* of the transaction. Each object request can have a Top, Sub, Revocable, or None relationship with the current transaction. If the object request has a Top, then that request operates completely independent of the transaction and exists in its own separate transaction context. If the object request has a Sub relationship with the current transaction, then a failure of that request will cause the entire transaction to abort and all changes to be *rolled back*. If the object request has a Revocable relationship with the current transaction, then the calling process must trap the exception thrown by the failing request and determine whether or not to abort or commit the current transaction. If the object request has a None relationship, then its success or failure has no effect on the current transaction, and the request itself is not executed in the context of any transaction.

2.3.4.1 Transaction Locks

To summarize the transaction process:

- No changes are committed until all Sub transactions have completed successfully.
- All Sub transactions must complete before the resources used by the transaction are released back to the system.
- As changes are not final until all Sub transactions are complete and the changes are committed, they could be revoked through a roll-back, and so a resource that has been updated as part of a transaction cannot be accessed by any other process or transaction.

Transaction locks protect against racing conditions and protect data integrity in the case of an aborted transaction. A racing condition can exist when two processes attempt to update the same resource at the same time. In this

case, the last process to finish determines the final result. The data integrity problem can arise if a process outside a transaction is allowed to access a resource that has been changed as part of a separate transaction. If a process addresses a resource, assuming it to be correct, and then a transaction aborts and *rolls back* to an old value, the process has just acted on *bad* data.

2.3.4.2 Process Locks

A process lock can occur when a process begins a transaction, and that transaction is waiting for a resource that has been allocated or is in use by another transaction.

As locked resources can impact active transactions, long running transactions with many Sub transactions are more likely to cause process locks. Therefore, long running transactions should be avoided wherever possible. An example of an administrator initiating a long-running transaction would be a shell script that creates large numbers of PC managed nodes using `wcrtpcmngnode`.

The side-effect of transaction based resource locks is that any given process that requires access to a resource locked by a transaction will be denied access until the locking transaction completes. If the transaction fails to complete, the process can remain locked indefinitely. It is possible that user defined tasks/jobs that execute `w` commands or make object requests can inadvertently cause a long running transaction.

2.3.4.3 Lock Relationships

Locks on resources can be inherited from other transactions and can be passed through the transaction cycle. Inherited locks can cause current and future transactions to stall while waiting for resources to be released back to the system.

Locks are passed between object requests that are organized in a hierarchy, such as request A calls request B, request B calls request C. Any resource lock obtained by request C is passed back to request B when C completes and then back to request A when request B completes.

2.3.5 Persistent Storage - The Tivoli Object Database

Tivoli stores object information in an internal, distributed object-oriented database. The database stores transient object information in a physical location. This is in the directory referenced by the `$DBDIR` environment variable in UNIX or `%DBDIR%` in Windows NT. The main file is `odb.bdb`. The TMR

server also uses an inherited method database called `imdb.bdb` to resolve implementation inheritance.

Having all management data stored in a database file means that if the process operating on the object is destroyed, the object is still intact. There is also persistent storage for object attributes. Distributed transaction services included in the framework make sure that all modifications to the database occur. The TMR server database contains TMR-wide management objects, such as policy region objects as well as references to objects located in client databases. Each TMR client maintains a database of local objects and objects that have been distributed to it, such as profiles. Database commands, such as `wchkdb` and `wbkupdb` are included to perform consistency checks on the database and any restoration it might need. See Chapter 5, “Tivoli Object Database Backup” on page 113 for more information.

2.3.5.1 Tivoli Name Registry

The Tivoli Name Registry (TNR) is a quick lookup table of object labels and object IDs. For example, a process can use the TNR to find an object ID from an object’s label. It could then use that ID to work with the object. The TNR is also the link between objects residing in separate TMRs. If you have interconnected TMRs, you periodically update the resources between the TMRs (using `wupdate` or the GUI) to make one TMR aware of the other’s objects. What is actually happening is the name registry is being updated to contain a reference to the OID of the object on the remote TMR. Tivoli implements the TNR in an object in the object database.

The name registry is the focal point for the prevention of name-space conflicts. As a lookup service, it provides a portable and efficient means for determining the object reference of a named resource. The TNR is also a single point from which applications can quickly and efficiently generate lists of resources, such as Available Machines.

Within Tivoli, most objects in the TNR can be divided into two categories: Distinguished objects associated with a single TMR, and resource objects that are typically associated with an application that models some system resource.

A distinguished object is usually an object unique to a TMR with a well-known name. Distinguished objects are frequently used as service providers (for example, the notification registry or the scheduler). Distinguished objects relate specifically to a TMR and are not exchanged during an update of resources between interconnected TMRs.

The `wlookup` command is our command line interface into the TNR. We can use `wlookup -a` or `wlookup -ar distinguished` to list all the distinguished objects that are registered in a database. Remember that these will always be local to a TMR, and you will not see distinguished objects from a connected TMR.

New distinguished types for V3.6 are:

SWDistUserLink This distinguished object class contains the interfaces (methods) used when an enduser uses the UserLink function from a Web browser to pull distribution packages.

SwdistRIMSupport This object is used to write distribution reports to the inventory database. It is added by the `wswdistrim -c` command to enable the historical database feature and is deleted by `wswdistrim -d`.

Figure 6 is an abridged listing showing some of the types returned by `wlookup -a` (with the new ones highlighted):

```
# wlookup -a            !Display a list of distinguished types that are registered

Administrators  1212391543.1.168#TMF_Administrator::Collection_GUI#
Regions 1212391543.1.194#SharedPolicyRegions::Engine#
RepeaterManager 1212391543.1.365
SWDistUserLink  1212391543.1.825#SWD_Ulink::UserLink#
Scheduler       1212391543.1.157#TMF_Scheduler::scheduler#
ServerManagedNode       1212391543.1.347#TMF_ManagedNode::Managed_Node#
SwdistRIMSupport        1212391543.1.854#SwdistRIM#
TME_server      1212391543.0.0
TMRBackup       1212391543.1.371#TMF_SysAdmin::InstanceManager#
TaskRepository  1212391543.1.214#TMF_Task::TaskRepository#
lost-n-found    1212391543.1.524#TMF_TGC::CollectionGUI#
```

Figure 6. Distinguished Objects List from wlookup

Resource objects are instances of Tivoli classes that are generally used to represent some system resource. There is usually more than one instance of a resource type. Examples of resources include Tivoli administrators, managed nodes, profile managers, and so on. The name of a resource object is usually specified by the administrator when the object is created.

Resource objects are exchanged between TMRs during an interconnected TMR update of resources. We can list resource objects through `wlookup -R` as shown in Figure 7 on page 24.

New Resource objects for V3.6 are:

DefaultCGIItems For Inventory and Software Distribution

LCF-NtLcfInstall	Executes method `NtLcfInstall` found in the lcf_bundle directory to configure and create the Windows NT TMA Endpoint
Uninstall	Contains an entry for each product that can be automatically removed with `wuninstall`.

EndpointManager is not new but is now in the resource list and can be seen with `wlookup -ar EndpointManager`. In V3.2, this was not listed in the resource list and could not be seen with `wlookup -ar`.

Figure 7 is an abridged list of resource types from the `wlookup -R` command:

```
# wlookup -R              !Display a list of resource types that are registered

ACP
Classes
DefaultCGIItems
DefaultHTMLItems
distinguished
Endpoint
EndpointManager
EnterpriseClient
EventServer
LCF-NtLcfInstall
LocatorDatabase
Uninstall
```

Figure 7. Resource Objects List from wlookup

When a TMR is initially installed, the TNR is created. The user-friendly names are then registered with their associated object references for the local distinguished objects. See “Object IDs” on page 34 and “Object References” on page 31 for more information about object names and references.

When additional resource types (such as policy regions, Tivoli administrators, and profile managers) are created, they are registered with the TNR. When an application is installed, new resource type (class) names are also registered with the TNR. Note that it is up to applications to determine whether their objects need to be registered in the TNR.

When an instance of a resource is deleted, the entry in the TNR is removed.

As already mentioned, we use `wlookup` to look through the TNR. A set of directory service functions in the application services run-time library (`wls`, `wcd`, and so on) provide a similar direct interface to the operations of the object database itself. However, the object database does not include any reference to connected TMRs (except through the NameRegistry object); so,

`wls` and similar commands will only show resources in the current TMR. See Section 2.4.1.1, “Registered Names” on page 32 for further details on referencing objects by name.

2.3.5.2 Managed Resources

Tivoli uses object technology to model real-world resources. In the context of Tivoli Enterprise, resources are Tivoli representations of elements in a computing enterprise. They may be things, such as computers, or they may be a set of rules that govern a system or set of systems. Resources that are subject to certain sets of rules within Tivoli are called *managed resources*. Tivoli calls the predefined rules that govern them *policies*.

Managed resources provide a model of the physical resources to be managed by Tivoli applications. In addition, they can mask platform-specific characteristics that present problems for portability and interoperability. It doesn’t necessarily matter to a Tivoli administrator whether a set of file system-type managed resources includes UNIX, Windows NT, or even MVS instances since the model alleviates problems associated with platform specifics.

2.3.5.3 Collections

A collection is a set of objects or references to objects. Objects, such as resource objects within a collection, are called members. You can query a collection or have an operation applied to the members of the collection.

Because collections can refer to other collections, they can be organized into hierarchies. Moreover, because collections are groups of references to objects, an object could belong to any number of collections. Note that some collection types, such as policy regions, alter the membership rules. It is not possible to be a member of multiple policy regions. A collection can be heterogeneous, meaning that it can collect objects of different types.

You usually see this collection hierarchy through the Tivoli desktop. When you start the desktop, the GUI shows you the contents of your own administrator collection. When you open a region, the GUI shows the contents of the region collection, and so on.

When working with collections and objects, Tivoli draws a distinction between removing and deleting. If you remove an object from the desktop, or use `wrm`, the object is removed from that desktop collection. If you delete an object from the desktop, or use `wdel`, that object is deleted. You can theoretically delete a collection without deleting its members and remove the last collection to which an object belongs without deleting the object, orphaning

the object. In practice, Tivoli prevents you from performing such an operation where it does not make sense.

In the *Tivoli Framework Reference Manual* you can see various commands for managing collections (`wdel`, `wcd`, `wln`, `wls`, `wmv`, `wpwd`, `wrefresh`, and `wrm`). These commands are operating directly on the object database and use the UNIX file system analogy to manage the collections and the objects that belong to the collection.

Note

Working with `wls`, `wcd`, and so on, refers to the local database. You will not see objects using these commands that do not contain a reference in the local database. Use `wlookup` to list resources in interconnected TMRs.

The following example illustrates this hierarchy. The names used are all case-sensitive:

```
# wls /             ! List the root 'directory' of the object collection hierarchy
Administrators
CurrentNtRepeat
EndpointManager
FpblockEngine
Installation
InterRegion
InterfaceRepository
InventoryUserLink
Library
MessageCatalog
NameRegistry
NetloadFactory
NotificationServer
PasswordModifier
Regions
RepeaterManager
SWDistUserLink
Scheduler
ServerManagedNode
SwdistRIMSupport
TME_server
TMRBackup
TaskRepository
lost-n-found

# wls /Administrators          ! Show the contents of the Administrators container
Root_gblnt-region
Root_tivdev01-region

# wls /Administrators/Root_tivdev01-region  ! Look at Root_tivdev01-region's 'Desktop'
Notices
Administrators
EventServer
UserLocator
EndpointManager
Scheduler
Root_tivdev01-region
```

```
tivdev01-region
# wls /Administrators/Root_tivdev01-region/tivdev01-region
NT_Indicator             ! Indicator Collection
PcMN_Inventory           ! Inventory Profile
NT-MN_Sentry             ! Sentry Profile
PcMN01                   ! PC Managed Node
```

The containers may be used a number of ways to refer to the same objects. For example, we could use `wls /Regions` to start looking down the tree from the highest level of regions.

Implementation Tip

Working with containers in this way demonstrates one area where a good naming convention helps. Using names ending in some indicator of what the object represents, such as -PM for a profile manager, -Region for a policy region, and so on, makes them easy to distinguish. Although this is good naming practice, note that you could also use the `-l` switch when using the `wls` command. This will show the object label, which usually identifies the type of the object.

2.3.5.4 Interconnected TMR Resource Exchange

TMRs and the name registry are closely related. As mentioned in “Tivoli Name Registry” on page 22, when two TMRs are connected to each other in a two-way connection, the TNR in each region can exchange information with the other about registered resources. This must be initiated through the GUI or using `wupdate`.

For one-way TMR connections, the information is passed only from the managed TMR to the managing TMR. The reason for such resource exchanges is that an application may need access to all the known resources in all connected TMRs. This information is obtained solely from information contained within the TNR of the local TMR. See Section 7.3, “Interconnected TMRs” on page 216 for more information about interconnected TMRs.

2.3.6 Instance Management

So far, we have looked at how objects can be stored hierarchically in collections. There is a hierarchy in another dimension that has more to do with how an object is created and how it behaves (you may find it useful to refer to Figure 8 on page 29 while reading this section). The instance management service provides much of the required infrastructure for objects to be logically associated with and managed by other objects. These objects support a common interface, and they are subject to a set of common policies.

The instance management service starts with a library interface and zero or more object's instances supporting the instance manager interface. Remember, an object is either a distinguished object or a model for some system resource type. The instance management interface defines the fundamental operations that support the management of multiple instances of an object type (or class). This support includes policy management and maintaining a record of the managed instances of the object type.

The relationship between an instance and an associated instance manager is one-to-one. The application services, however, support several instance managers that manage objects of the same type, one for each TMR.

Multiple instance managers are used for a variety of reasons, including scalability, availability, and configuration control. An object that supports the library interface (a library object) maintains a list of all instances of instance manager objects within a TMR. In this way, a library object serves as the common source of information about the known types of objects within a TMR. Compare this with the object database collection hierarchy that represents how those objects relate to the installed environment.

A library object also maintains information about each type of object managed by instance managers, including whether they have policies associated with them. Finally, there is one library object for each TMR.

The objects provided by a management application can be divided into one or more object types (classes) that may be managed by one or more instance managers. An object type (such as the ManagedNode type) is a set of objects that share a common interface. Each object of the same type (such as a defined managed node) is called an instance of the type.

To summarize the systems management framework, object types (classes) are associated with a particular instance manager. The instance managers are registered with, and stored in, the library, which provides a central repository for system administration object information within a TMR.

An instance manager can have references to one or more policy regions to encapsulate a set of management policies. In this case, the manager is said to represent a managed resource type (see Section 2.3.5.2, “Managed Resources” on page 25). A Tivoli Policy Region defines default and validation management policies. These are encapsulated and implemented in the policy default objects and policy validation objects associated with an instance manager. They also support high-level operations that manage a particular object type.

It is common knowledge that a default policy, through policy default objects, supports operations to coordinate the creation of objects as well as defines the initial (default) values of the policy-driven object attributes. Likewise, for validation policy, policy validation objects support methods that validate initial values or changes to object attributes.

The following figure summarizes the library, instance manager, object policy, and object instance hierarchy:

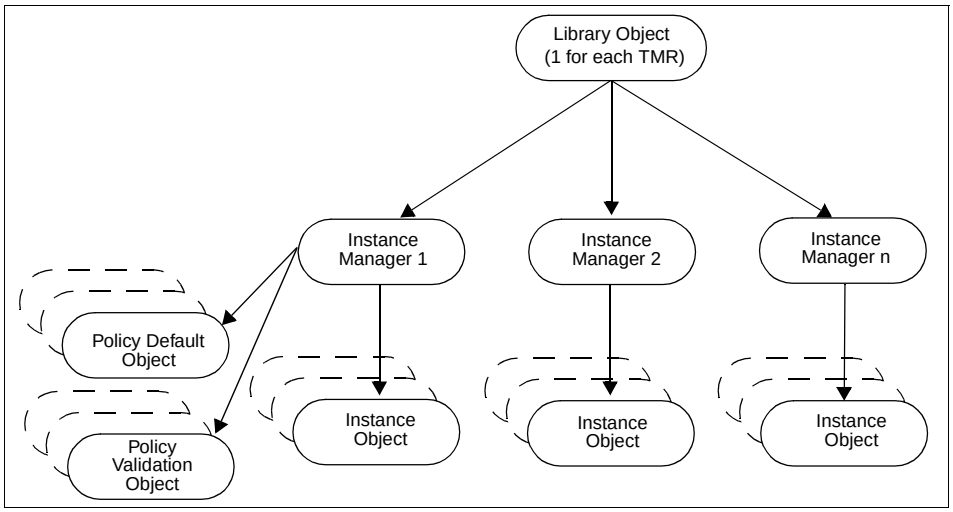


Figure 8. Instance Manager Layout

We can use `wls -l /Library` to list all the instance managers by showing all the members of the TMRs library collection. The `-l` option displays the object ID (OID) and label of each member:

```
# wls -l /Library        !List all the instance managers below the object library.
.
.        !  Partial list follows:
.
1212391543.1.287#TMF_SysAdmin::InstanceManager# ProfileManager
1212391543.1.287#TMF_SysAdmin::InstanceManager# ManagedNode
1212391543.1.287#TMF_SysAdmin::InstanceManager# PcManagedNode
1212391543.1.287#TMF_SysAdmin::InstanceManager# Endpoint
1212391543.1.287#TMF_SysAdmin::InstanceManager# PolicyRegion
.
```

Note that if you wanted to display a list of all instances of the managed node object type, you could use `wls -l /Library/ManagedNode`.

2.3.6.1 Policy Regions

A policy region is a special type of collection; it is a collection of policy-driven objects. Like all collections, policy regions can be arranged hierarchically according to organization and administrator-specific criteria. Furthermore, they can contain any set of managed objects as specified by an administrator.

A policy region can also contain other policy region objects. A policy region object that is a member of another policy region is known as a subregion and may inherit its parents' policies or specialize them.

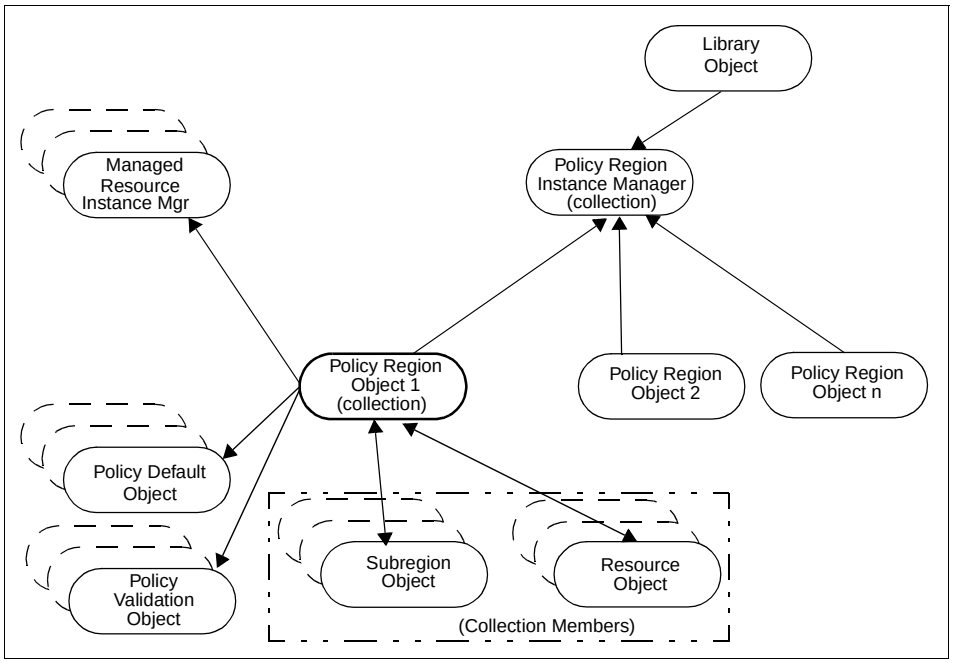


Figure 9. Policy Region Object Relationships

This figure shows the object relationships between a policy region and its managed resources, and at the same time, identifies which are also linked by collections and memberships. There is a reference to an instance manager for each managed resource type. Instances of the managed resources are shown as members of the policy region. They will also be members of their instance manager's collection. For example, profile manager resource objects will be found in `/Library/PolicyRegion/regionname` as well as in `/Library/ProfileManager`. The library container provides collections for all resources grouped by resource type. The enforced policies are shown separated into default and validation types.

The commands `wls -l` and `wlookup -ar` produce similar results:

```
# wls -l /Library/ProfileManager
1212391543.1.606#TMF_CCMS::ProfileManager#      ACPdefault
1212391543.1.868#TMF_CCMS::ProfileManager#      TivoliDefaultPhoneProfileMgr
1212391543.1.969#TMF_CCMS::ProfileManager#      TivoliDefaultPM
1212391543.1.979#TMF_CCMS::ProfileManager#      Inventory
1212391543.1.1256#TMF_CCMS::ProfileManager#     TME 10 Security
1212391543.1.1281#TMF_CCMS::ProfileManager#     Sentry
1212391543.1.1290#TMF_CCMS::ProfileManager#     Sentry2
1212391543.1.1293#TMF_CCMS::ProfileManager#     Inventory1
1212391543.1.1409#TMF_CCMS::ProfileManager#     NT_FilePackage2
1212391543.1.1473#TMF_CCMS::ProfileManager#     NT_UserAdmin2
1212391543.1.1475#TMF_CCMS::ProfileManager#     NT_SecAdmin2

# wlookup -ar ProfileManager
ACPdefault      1212391543.1.606#TMF_CCMS::ProfileManager#
Inventory       1212391543.1.979#TMF_CCMS::ProfileManager#
Inventory1      1212391543.1.1293#TMF_CCMS::ProfileManager#
NT_FilePackage2 1212391543.1.1409#TMF_CCMS::ProfileManager#
NT_SecAdmin2    1212391543.1.1475#TMF_CCMS::ProfileManager#
NT_UserAdmin2   1212391543.1.1473#TMF_CCMS::ProfileManager#
Sentry  1212391543.1.1281#TMF_CCMS::ProfileManager#
Sentry2 1212391543.1.1290#TMF_CCMS::ProfileManager#
TME 10 Security 1212391543.1.1256#TMF_CCMS::ProfileManager#
TivoliDefaultPM 1212391543.1.969#TMF_CCMS::ProfileManager#
TivoliDefaultPhoneProfileMgr    1212391543.1.868#TMF_CCMS::ProfileManager#
ep_inventory    1998892590.1.998#TMF_CCMS::ProfileManager#
ep_monitors     1998892590.1.1121#TMF_CCMS::ProfileManager#
```

The main difference here is that `wlookup` includes objects from interconnected TMRs. The last two objects in the `wlookup` output (ep_inventory and ep_monitors) are from another TMR. As you can see, they have a different region number.

2.4 Tivoli Objects

Now we can look in more detail at how objects reference each other and how we can reference objects from the CLI. We will also describe the Tivoli concept of a unique object ID (OID) and the relationship between objects in the Tivoli management object database.

2.4.1 Object References

When an object is referenced from a CLI command, the reference is not usually going to be an absolute object reference like those used in programming. Instead, a user-friendly name is used. This user-friendly name comes from a name given to the object by the user of the application, such as when a policy region is created. Here, we will describe the two different forms of names that can be used with CLI commands:

- Registered names
- Object paths

Tivoli CLI programs support both naming schemes. Sometimes, you will find it more convenient to use one form over the other. If you receive an error message indicating that a resource cannot be found, try a different naming convention.

2.4.1.1 Registered Names

A registered name is the name by which a resource instance is registered with the Tivoli Name Registry (TNR) when it is created (see also Section 2.3.5.1, “Tivoli Name Registry” on page 22). Every resource has a name and is an instance of some particular type. For example, a policy region called rh0255b-region has a name rh0255b-region and is of type PolicyRegion. The syntax for specifying a resource using the registered name facility is `@type:name`, where `type` is the resource type and `name` is the particular instance of that resource on which you wish to perform some operation.

Some examples of using registered names as arguments for directory-style commands are:

```
wls @PolicyRegion:rh0255b-region
wmv @ManagedNode:ayers-rock @PolicyRegion:rh0255b-region
```

These commands list the contents of the rh0255b-region policy region and then move the ayers-rock managed node to that region. This is equivalent to the drag-and-drop method of the GUI. The TNR is responsible for resolving the name type and finding the referenced resource.

Note

To see all the resource types listed in the TNR, use the `wlookup -R` command. Since all resource types must also have an instance manager, you could also use `wls /Library` to get a list of instance managers that will use the same name. Note that the Tivoli Framework and applications can choose not to include objects in the TNR.

The name registry does not allow two resources of the same type to have the same name within a single TMR. However, it is possible for resource names to be duplicated within two (or more) connected TMRs. This would most likely happen when connecting TMRs that had been operating independently. Once the name registry is updated between the TMRs, Tivoli prevents you from creating a new resource that has the same name in the same resource type in another interconnected TMR.

If you attempt to perform an action on a resource with a duplicated name, an error message is returned, and the action is not performed. To avoid this

situation, you should either rename one of the resources or differentiate between the resources by appending a region name to the resource name, as follows:

```
wls @ManagedNode:moria#moria-Region
```

This qualification by TMR name is usually handled automatically for operations initiated through the GUI.

2.4.1.2 Object Paths

We have already seen examples of object paths in 2.3.6, “Instance Management” on page 27. Object Paths are similar to path names in file systems, and here you are manually traversing the object hierarchy rather than relying on the TNR to resolve registered names. A path reference can be relative or absolute. A relative path can start with any character including the special path components dot (`.`) and double-dot (`..`). An absolute path is one that starts with a slash (`/`) character.

The syntax for specifying a resource using the object path name style is `/distinguished/parent/[type:]name`, where `distinguished` is a resource type, such as Regions, `parent` is the start of the object path name, such as a policy region name. `Type` is used to further identify a resource, and name is the particular instance on which you wish to perform some operation.

Some examples of object path names used as arguments for commands are as follows:

```
wls /Regions/rh0255b-region
or
wcd /Regions/rh0255a-region
wmv ../rh0255b-region/ayers-rock ../Servers-region
```

You often use the optional type qualifier when you need to name a particular resource that has the same name as some other resource of a different type. For example, suppose policy region Engineering-region had a profile manager called Servers and a policy subregion also called Servers. To specify the profile manager and not the subregion using an object path name, you could use the following:

```
wls /Regions/Engineering-region/ProfileManager:Servers
```

As mentioned in 2.3.5.3, “Collections” on page 25, you could have a resource naming convention that included the resource type that would remove the need to have to qualify types in this way.

If you specify a resource using an absolute path, its location is not ambiguous between connected TMRs. However, if you use a relative path, both your home and current administrator collection must be located before the resource can be found. Each administrator's home collection is */Administrators/Name*, where *Name* is the administrator's Tivoli name. If you have recently issued a `wcd` command, Tivoli contains a record that specifies the location of the current administrator collection. Otherwise, no such record exists, and in this case, the current administrator collection can be ambiguous if there are multiple connected TMRs. For example, suppose you are an administrator named John (with a login name johnc) in TMR A, and there is another administrator named John (with a login name of jsmith) in TMR B. When you specify an action to be performed on a resource, Tivoli searches for the /Administrators/John collection. The search finds collections belonging to you and jsmith. Because Tivoli cannot determine which home collection you meant to specify, an error message is returned, and the action is not performed. You can execute the `wcd` command to change to the correct administrator collection to prevent this problem from occurring.

If a path begins with a single period (`.`) or a double period (`..`), the object path is a relative path. A relative path is like an absolute path, except it is relative to the current working collection. Every administrator object maintains an object path to the administrator's current position in the hierarchy. This can be likened to the current working directory in a file system. Administrators can use the `wcd` command to change their current working collection or use `wpwd` to check the current working collection. An administrator also has a home collection, which is the object path to the administrator's desktop object (/Administrators/AdminName).

2.4.2 Object IDs

An object ID (OID) is a three-part identifier of the form 1264987995.1.326. The OID is made up of the following:

- **The region number**. Objects in Tivoli 3.0 and above have ten-digit region numbers (1264987995 in the example above). Objects in Versions 2.1 to 2.5 have six-digit region numbers, and objects in releases prior to 2.x had four-digit region numbers. Region numbers are generated at TMR server install time using an algorithm designed to make them unique. Note, however, that due to the random element in the region number generation, there is a small possibility that two TMRs could generate the same region number. If this happens, then one of the regions will need to be reinstalled if it was to be connected to another region with the same number.
- **The object dispatcher (oserv) number**. Sometimes referred to as the host number. For performance reasons, Tivoli recommends up to 200 be

defined in any one TMR. Dispatcher numbers are assigned in the order managed nodes are installed. The TMR server will be dispatcher 1, the next managed node installed will be 2, and so on. If a node is deleted and reinstalled, it will receive a new number, the next in the sequence.

- **The object number**. This is a number for the object itself.

The region number will be the same for all hosts in a TMR. The object number will be unique among objects in a single host but could be the same as object numbers in other hosts. Tivoli never reuses object dispatcher (host) numbers even if a host is removed and added later. This always makes the combination of the three numbers unique.

There is a special OID for an object called the base object. See Section 2.4.5, “The Tivoli Base Object” on page 44 for more information on this object.

During a Tivoli server install, the object database is built by taking a pre-built database from the install media and altering those objects that are specific to the new TMR. A standard server database already consists of around 400 objects before you start adding any applications or customizations. This fact can also be a help when navigating your way around objects. For example, if you see an administrator object with an object number around 190 (x.1.190), then this will be the Tivoli root administrator as this ID already existed in the database when it was first created.

2.4.2.1 Displaying Object Contents with objcall

We have already seen that we can use directory-type commands (`wls`, `wpwd`, `wmv`, and so on) to explore and, to some extent, manipulate resource objects in the object database. To find out more about Tivoli objects, we need to make use of object calls.

Note

If you wish to experiment with the commands presented here, be sure to check any warnings given in this chapter and review Chapter 6, “Commands and Logs for Troubleshooting” on page 131 and the *Tivoli Framework Reference Manual* for more information. Experimentation should always be limited to a stand-alone test system, and be sure to have excellent backups before trying any of this in a live environment.

Tivoli provides the `objcall` command to make direct object calls from the command line. The format of the command is `objcall OID method`, where OID is the OID of the object, as described, above and method is a valid object

method. One commonly used method is *contents*. This simply displays the contents of an object or its attributes and methods.

The following example shows the contents (attributes and methods) of an object (in this example, the object is a managed node). First, we use `wlookup` to list all managed nodes in the region so we can find the OID of the one we wish to know more about:

```
# wlookup -ar ManagedNode                  !find all the Managed Node instances
itso2   1998892590.1.348#TMF_ManagedNode::Managed_Node#
itso3   1295714281.1.348#TMF_ManagedNode::Managed_Node#
rh2900b 1998892590.2.7#TMF_ManagedNode::Managed_Node#
```

Note

Method, attribute, resource-type, and other names used in commands are case sensitive.

Next, we use `objcall` to invoke the contents method on that object:

```
# objcall 1998892590.2.7 contents  !find the attributes/methods of the object rh2900b
ATTRIBUTE:_BOA_id
ATTRIBUTE:class_objid
ATTRIBUTE:collections
ATTRIBUTE:consumers
ATTRIBUTE:databases
ATTRIBUTE:label
ATTRIBUTE:last_failed
ATTRIBUTE:members
ATTRIBUTE:pres_object
ATTRIBUTE:pro
ATTRIBUTE:pro_name
ATTRIBUTE:profile_push_order
ATTRIBUTE:push_trans_commit_behavior
ATTRIBUTE:resource_host
ATTRIBUTE:skeleton
ATTRIBUTE:sort_name
ATTRIBUTE:state
ATTRIBUTE:subscriptions
```

You can see from this list that there are no methods in a managed node object. Since the methods for all managed nodes are the same, the methods for a managed node object are in a *behavior* object. The behavior object OID is in an attribute (called behavior) of the object class (instance manager) of the managed node and not in a managed node instance itself.

2.4.2.2 Finding an Object's Class and Class Contents

The following example shows how to find the class (implemented by the instance manager) of an object and all the contents and the methods of the class (the methods being in the behavior object).

1. First, we find the OID of the original object (in this case, a managed node) using `wlookup`:

```
# wlookup -r ManagedNode rh0255a   !find the Object id of the managed node rh0255a
1264987995.2.7#TMF_ManagedNode::Managed_Node#
```

2. If we want, we can use the getattr method on an `objcall` to get the contents of the label attribute, the label of the object. If there is a problem with the method, we can use another IDL calling mechanism, `idlattr`, to take a look at the label from the OID. This is a double-check to make sure we have the correct OID from the previous step:

   ```
   # objcall 1264987995.2.7 getattr label    !get the label
   rh0255a                                !label of the MN instance object is rh0255a
   ```

   OR:

   ```
   # idlattr -t -g 1264987995.2.7 label TMF_ManagedNode::Managed_Node     !get the label
   rh0255a                            !label of the MN instance object is rh0255a
   ```

   Using `objcall OID getattr attrname` or `idlattr -t -g OID attributename typename` is a little like `objcall OID contents`. However, instead of looking at the contents of an object (which may include attributes and methods), `idlattr` with the `-g` flag gets the contents of the attribute itself.

Important

Unless you are very familiar with object attributes, or you are experimenting on a properly isolated test system, it is recommended that you use `objcall OID getattr` instead of the `idlattr` command. If using `idlattr`, be sure to always use the `-g` (get) flag. The default is otherwise to **set** (`-s`) the attribute, an activity that could alter the execution of a method. Recovery from such a situation may include the restore of a previous database backup. See also Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

3. We can get the OID of the class object (instance manager) as follows:

   ```
   # objcall 1264987995.2.7 getattr class_objid
   1264987995.1.322#TMF_SysAdmin::InstanceManager#
   ```

   Or:

   ```
   # idlattr -t -g 1264987995.2.7 class_objid Object   !get instance manager of MN
   1264987995.1.322#TMF_SysAdmin::InstanceManager#
   ```

   So OID 1264987995.1.322 is the instance manager for the managed node object.

   Or:

   ```
   #wls -od /Library/ManagedNode
   1264987995.1.322#TMF_SysAdmin::InstanceManager#
   ```

4. Here, we take a look at the label of the instance manager to confirm it is the instance manager of the ManagedNode class:

   ```
   # objcall 1264987995.1.322 getattr label
   ```

Or:

```
# idlattr -t -g 1264987995.1.322 label TMF_SysAdmin::InstanceManager  !Get label
ManagedNode    !This is the label of the object class (Instance Manager) of the MN
```

5. Now, we'll get the OID of the behavior object from the behavior attribute in the class. The behavior object has the methods of the class:

```
# objcall 1264987995.1.322 getattr behavior   !Find object id of behavior object
1264987995.1.324
```

Or:

```
# idlattr -t -g 1264987995.1.322 behavior TMF_SysAdmin::InstanceManager
1264987995.1.324
```

6. And now we can `objcall OID contents` to list the contents of the managed node class behavior object, thus listing all the attributes and methods:

```
# objcall 1264987995.1.324 contents  !Attributes & methods of the class ManagedNode
ATTRIBUTE:label
ATTRIBUTE:skeleton
METHOD:_get_label
METHOD:_set_label
METHOD:add_interface
METHOD:add_interface_done
METHOD:add_ip_interface
METHOD:architecture
METHOD:avail_space
METHOD:cancel
METHOD:contact_host
METHOD:deep_read_dir
METHOD:define_tme_ip_interface
METHOD:del_ip_interface
METHOD:delete_things
METHOD:display_view
METHOD:display_xterm
METHOD:edit_interface
METHOD:edit_interface_done
METHOD:execute_task
METHOD:files_transfer
METHOD:get_ip_interface
METHOD:hostid
METHOD:install_directory
METHOD:interpreter
METHOD:list_ip_interfaces
METHOD:make_directory
METHOD:memory_size
METHOD:name
METHOD:ok
METHOD:os_name
METHOD:os_release
METHOD:os_version
METHOD:present_parent
METHOD:privileged_deep_read_dir
METHOD:privileged_files_transfer
METHOD:privileged_make_directory
METHOD:privileged_read_dir
METHOD:privileged_read_from_file
METHOD:privileged_read_link
METHOD:privileged_receive_files
METHOD:privileged_remove_file
METHOD:privileged_set_file
```

```
METHOD:privileged_set_files
METHOD:privileged_stat_file
METHOD:privileged_system
METHOD:privileged_transmit_files
METHOD:privileged_write_to_file
METHOD:read_dir
METHOD:read_from_file
METHOD:read_link
METHOD:receive_files
METHOD:refresh_view
METHOD:reg_notify_ip_change
METHOD:remove
METHOD:remove_file
METHOD:remove_interface
METHOD:reset
METHOD:set_file
METHOD:set_files
METHOD:set_ip_interface
METHOD:stat_file
METHOD:system
METHOD:system_time
METHOD:system_time_zone
METHOD:test_method
METHOD:toggle_state
METHOD:toggle_state_ok
METHOD:toggle_state_switch
METHOD:transmit_files
METHOD:undefine_tme_ip_interface
METHOD:unreg_notify_ip_change
METHOD:write_to_file
METHOD:xterm
```

To summarize the steps we found:

1. Using `wlookup` we found the OID of the object. In the example, the object is an instance of the object managed node.
2. We looked at the label attribute that gave us rh0255a as the label of the instance.
3. Then we found the reference of the class object, the instance manager.
4. We checked the label of this object and saw that it was ManagedNode. This is the Instance Manager for the object type managed node, or in other words, this is the class of the managed node object.
5. To find all the methods of the class Managed node, we found the object that has all the methods of the class, the behavior object.
6. Then we looked at the contents of the behavior object, and it showed all the methods for the class (in the example, the methods are for the managed node class).

In Section 2.4.4.1, “A Quicker Way to Find a Method” on page 42 we will see a slightly quicker way of getting to the OID of the behavior object through the *resolve* method.

Now comes the clever part. Let us suppose we were experiencing some problem with this object, in this case, a managed node. We might want to try

running one of the methods by hand from this managed node object. A simple test here would be to use `objcall` to kick off the managed node object's _get_label method:

```
objcall 1264987995.2.7 _get_label          !Note OID is of a managed node
@@rh0255a                                  !Result is the label
```

Note

When using `objcall` to do something like `objcall OID getattr`, many methods will give you results with additional bits of data that appear as odd ASCII characters before the information you are interested in (as in the previous `objcall OID _get_label` example). When working from the command line and retyping the data, this is not a problem. However, when using scripts or piping information, you may need to try alternatives (such as `idlattr`) or use string manipulation to ensure you have just the piece you need.

As the managed node object inherits the methods of its class, we can call a class method using the managed node instance object.

Important

The intention here is to help you understand how to find methods in use and how to try to run them by hand if the need arises. The example we used is pretty harmless. We used methods that are non-destructive in nature and could actually reassure us that a given object can be contacted and is functioning. However, unless you are sure of what a method will achieve, you risk **irrevocably damaging your object database**. Before taking any such actions, back up your TMR. We recommend testing it first on a test machine that can be reinstalled if a problem occurs. Tivoli support is not going to be able to assist in the reversal of damage caused by an inappropriate use of `objcall`. The only option will be a restore of a previous database backup.

2.4.3 Endpoint Objects

The object ID for TMA objects (dataless endpoints) is constructed using the triplet form (region.object_dispatcher.object_number) and appending a plus sign (+). This results in an object ID of the form R.D.P+, where

R Is the TMR region number.

D Is the number of the object dispatcher for the endpoint.

P Is the object number of the prototype object of the class for which the method is defined.

+ Indicates a TMA object reference.

The object dispatcher number uniquely identifies an endpoint. The TMR server maintains a mapping of endpoint dispatcher numbers to gateway objects; So, it can route a request to the appropriate gateway. TMA objects are not created as instances in the object system and do not have a unique per object state maintained for them by the object system. Therefore, there is no need to create an instance of each object on endpoints as you do in the full framework. Instead, they are born when an appropriate prototype object and an appropriate endpoint dispatcher number are combined in this form. The endpoint method runs on the endpoint without calling the create_instance method.

To make use of TMA objects, you obtain from the registry or profile manager an object in the above form and invoke it. The system locates the appropriate shared state (defined by the prototype object) and services the method request at the appropriate location. Because there is no object data associated with an object by the system, applications have to assume responsibility for maintaining their own persistent store outside the context of the TMA base services.

2.4.4 Object Relationship

We have seen that the database is made up of classes. We have also seen an example that demonstrates that each class has base objects that all of the instances of that class inherit. Most classes have the following objects:

- Behavior Object (BO). Objects that define the functions of the class and methods to implement the class.
- Display Object (DO). Presentation object. All compiled DSLs that make up the dialogs for the class as well as all methods and attributes for the presentation.
- Extended Behavior Object (XBO). All customized behavior methods.
- Default Policy Object (DPO). Default policy methods.
- Validation Policy Object (VPO). Validation policy methods.
- Prototype Object (PO). The object that the new member will resemble. This is where all the attributes of a class are kept.

A list of all the classes is in the library class object.

When a new instance of a class is created, it is cloned from the prototype object. It can have its own display object and extended behavior object for customizations on the instance, but in general, it looks to the extended

behavior object of the class, then to the behavior object for methods, and finally, to the display object for presentation dialogs.

The following figure shows the object's relationship:

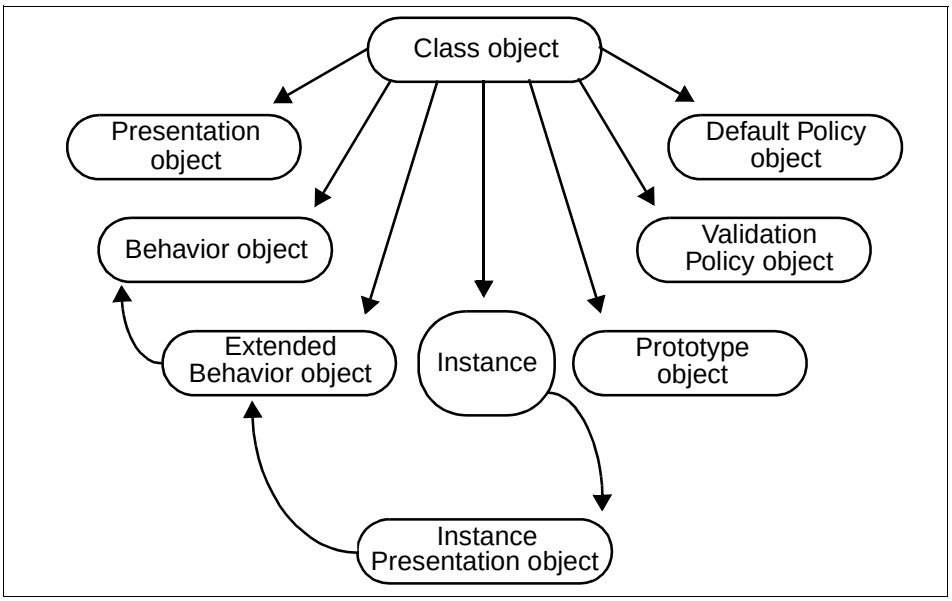


Figure 10. Class and Instance Object Relationship

2.4.4.1 A Quicker Way to Find a Method

The following sequence of steps is designed to show how the object hierarchy works. The objective here is to find the xterm method used both by the `wxterm` command and the xterm option from a managed node icon in the GUI.

As we have seen before, the xterm method should be found on the behavior object for the ManagedNode class. Perhaps by looking at an oservlog or a trace, we know the method name is xterm and that the managed node object can initiate the xterm method. Here is a slightly quicker way of finding the object that contains this method:

Scenario 1:

```
# wlookup -ar ManagedNode   !find all the instances of the Managed Node class to get OID
k124a 1562489759.2.7#TMF_ManagedNode::Managed_Node#
rh0255a 1264987995.2.7#TMF_ManagedNode::Managed_Node#
rh0255b.itsc.austin.ibm.com 1264987995.1.327#TMF_ManagedNode::Managed_Node#
rh0255c.itsc.austin.ibm.com 1562489759.1.327#TMF_ManagedNode::Managed_Node#
rh0255e 1562489759.3.7#TMF_ManagedNode::Managed_Node#
tivdev02 1482082604.1.326#TMF_ManagedNode::Managed_Node#
```

Now, because we know that `xterm` can be called by the ManagedNode object, we can use a method called resolve to find out where the xterm method resides. The resolve method works back up the invocation chain and reports the OID of the object that contains the method:

```
# objcall 1264987995.2.7 resolve xterm    !Find the OID of the Behavior object
1264987995.1.324                              !This is a quick way to do it..!

# objcall 1264987995.1.324 contents        !methods of the class ManagedNode
ATTRIBUTE:label
ATTRIBUTE:skeleton
METHOD:_get_label
METHOD:_set_label
METHOD:add_interface
METHOD:add_interface_done
METHOD:add_ip_interface
METHOD:architecture
METHOD:avail_space
METHOD:cancel
METHOD:contact_host
METHOD:deep_read_dir
METHOD:define_tme_ip_interface
METHOD:del_ip_interface
METHOD:delete_things
METHOD:display_view
METHOD:display_xterm
METHOD:edit_interface
METHOD:edit_interface_done
METHOD:execute_task
METHOD:files_transfer
METHOD:get_ip_interface
METHOD:hostid
METHOD:install_directory
METHOD:interpreter
METHOD:list_ip_interfaces
METHOD:make_directory
METHOD:memory_size
METHOD:name
METHOD:ok
METHOD:os_name
METHOD:os_release
METHOD:os_version
METHOD:present_parent
METHOD:privileged_deep_read_dir
METHOD:privileged_files_transfer
METHOD:privileged_make_directory
METHOD:privileged_read_dir
METHOD:privileged_read_from_file
METHOD:privileged_read_link
METHOD:privileged_receive_files
METHOD:privileged_remove_file
METHOD:privileged_set_file
METHOD:privileged_set_files
METHOD:privileged_stat_file
METHOD:privileged_system
METHOD:privileged_transmit_files
METHOD:privileged_write_to_file
METHOD:read_dir
METHOD:read_from_file
METHOD:read_link
METHOD:receive_files
METHOD:refresh_view
```

```
METHOD:reg_notify_ip_change
METHOD:remove
METHOD:remove_file
METHOD:remove_interface
METHOD:reset
METHOD:set_file
METHOD:set_files
METHOD:set_ip_interface
METHOD:stat_file
METHOD:system
METHOD:system_time
METHOD:system_time_zone
METHOD:test_method
METHOD:toggle_state
METHOD:toggle_state_ok
METHOD:toggle_state_switch
METHOD:transmit_files
METHOD:undefine_tme_ip_interface
METHOD:unreg_notify_ip_change
METHOD:write_to_file
METHOD:xterm                              ! **Here is the xterm method**
```

In this example, you first find the object ID of the object, then you use the command `objcall` with the resolve method to find the behavior object. Then you can use the contents method to see all the attributes and methods. In the list of methods, we confirm that the xterm one is here.

2.4.5 The Tivoli Base Object

All objects inherit from the base object, which is made up of methods and a few objects. Its object ID is 0.0.0, or regionnumber.1.0. This object has a few attributes but is mainly made up of methods. This, for example, is where the resolve method resides:

```
# objcall 0.0.0 contents     !Find all the methods and attributes of the Base Class
ATTRIBUTE:HostLocation
ATTRIBUTE:NameRegistry
ATTRIBUTE:fileioRef
ATTRIBUTE:master_base_oid
ATTRIBUTE:oserv
ATTRIBUTE:security_objid
ATTRIBUTE:skeleton
METHOD:addattr
METHOD:bo_set_acl
METHOD:clone
METHOD:contents
METHOD:corba_setattr
METHOD:echo
METHOD:get_host_location
METHOD:get_master_base
METHOD:get_name_registry
METHOD:get_oserv
METHOD:get_security_objid
METHOD:getattr
METHOD:i_getattr
METHOD:i_setattr
METHOD:is_visible
METHOD:o_add_groups
METHOD:o_addattr
```

```
METHOD:o_backup
METHOD:o_clone
METHOD:o_contents
METHOD:o_get_capabilities
METHOD:o_get_groups
METHOD:o_get_principal
METHOD:o_getattr
METHOD:o_is_visible
METHOD:o_remove_groups
METHOD:o_restore
METHOD:o_rmattr
METHOD:o_rmobj
METHOD:o_self
METHOD:o_set_groups
METHOD:o_setattr
METHOD:o_visible
METHOD:oi_add
METHOD:oi_get_list
METHOD:oi_move
METHOD:oi_remove
METHOD:oi_stat
METHOD:om_add_header
METHOD:om_create
METHOD:om_debug
METHOD:om_define
METHOD:om_enable
METHOD:om_get_acl
METHOD:om_get_definition
METHOD:om_get_implid
METHOD:om_get_roles
METHOD:om_get_sig
METHOD:om_remove
METHOD:om_set_acl
METHOD:om_set_catalog
METHOD:om_set_id
METHOD:om_set_implid
METHOD:om_set_roles
METHOD:om_set_sig
METHOD:om_stat
METHOD:om_undefine
METHOD:resolve
METHOD:rmattr
METHOD:rmobj
METHOD:self
METHOD:setattr
METHOD:visible
```

2.4.6 TMA Endpoints

It is very important to understand the concepts of database and dataless Profile Managers - as well as the distinction between database mode and dataless mode endpoints.

A profile manager operates in one of two modes: Database and dataless.

The *database profile manager*, which operates in database mode, is available in all releases of the Tivoli Framework. The *dataless profile manager,* which operates in dataless mode, was introduced in the Tivoli Framework, Version 3.2. To see an example of the considerations important to applications with

dataless and database profile managers, see 12.7.1, “Local Profile Copies” on page 403.

2.4.7 Database Profile Managers

In database mode, a profile manager distributes profiles to the subscriber’s profile database. If the subscriber is a database endpoint (such as managed node, profile manager, or sentry proxy), the profile data is written to the subscriber’s profile database.

Because TMA endpoints do not have a profile database to write to, database profile managers cannot distribute to TMA endpoints. A database profile manager can only have subscribers of the following types:

- Managed nodes
- PC managed nodes (not applicable for Distributed Monitoring)
- NIS domains
- NetWare managed sites
- Other profile managers - database as well as dataless
- SentryProxies (for Distributed Monitoring only)

Remember, the *database* of database profile manager refers to the *target* of the distribution not the database in which the profile manager keeps its own information. A database profile manager distributes to the profile database of its subscriber.

2.4.7.1 Dataless Profile Managers

In dataless mode, a profile manager writes directly to the system files of its subscribers.

A dataless profile manager can have the following types of subscribers:

- TMA endpoints
- Managed nodes
- PC managed nodes (not applicable for distribution of SentryProfiles)
- NIS domains
- NetWare managed sites
- SentryProxies (for Distributed Monitoring only)

A dataless profile manager distributes to the system files on all endpoints - regardless of the type. If the endpoint does have a profile database, this is bypassed during profile distribution. The effect of this is, that if a managed node is subscribed to a dataless profile manager, there will be no local copies after distribution of the profile. Local profiles are available only when

distributed from database profile managers. Refer to 12.7.1.2, “Local Profile Copies and Dataless Profile Managers” on page 404 for more information.

Since there are no system files attached to a profile manager, dataless profile managers cannot distribute to other profile managers, which require profiles to be written to a profile database.

Even though a dataless profile manager indeed does have a profile database of its own, in which it keeps profiles and subscriber information, it is referred to as *dataless* because it distributes profiles to the dataless endpoint on its subscribers.

2.4.7.2 TMA Endpoint Methods

A dataless endpoint method is a method that runs directly on a TMA endpoint. It is implemented using the special application mini-runtime library, libmrt, which contains a subset of Tivoli operations.

When a TMA endpoint method is invoked, the endpoint uses the method executable stored in its method cache. If the method is not in the cache or is not the most current version, the gateway downloads the correct method to the endpoint, and it is added to the endpoint's cache for future use.

There are two caches:

- The method header cache is located on the gateway. The method header contains the path to the method executable, the ACLs, the user and group IDs, and the path to the dependencies for the method. It does not contain the executable. The master copy of the method header cache is stored on the TMR server; the gateway's method header cache is a copy.
- A method cache is located on the endpoint. The endpoint cache contains executables, scripts, and their dependencies. The master copy is on the gateway; a copy of executables is downloaded to the endpoint as needed.

Endpoint methods are based on Tivoli prototype objects and support many features of that model, including IDL bindings and setuid and setgid methods. But endpoint methods differ from full framework methods in these ways:

- Unlike the full framework, the TMA is single-threaded. Endpoint methods must be single-threaded, and only per-method methods are supported for endpoint methods. That is, each time a method is invoked on an endpoint, a new instance of the method executable loads, executes, and terminates.
- There is no transaction support for endpoints in the TMA environment. Gateways do not provide transaction management services to endpoint methods.

- Besides the executable containing the method, a method may require supporting files, such as shared libraries, message catalogs, or other files. These supporting files are called dependencies and are defined as such in a method's definition. The gateway ensures that all dependencies for a method are downloaded to the endpoint before invoking a method.

In the full framework, only the file path name of the method body or implementation, the binary program, or script that contains the method entry point, is stored in the method header. Any supporting files that the method requires are assumed to be present because all binaries, libraries, message catalogs, and so on, are installed on every managed node and TMR server. However, TMA endpoints do not have any methods or supporting files present on them when endpoints are initially installed. Method bodies are identified in the standard method header and are downloaded when needed. Because the dependencies (or supporting files) must also be downloaded when missing, the dependencies must be called out in the endpoint method so they can be present when needed.

2.4.8 Gateway Methods

A gateway method runs on the gateway servicing the endpoint. It runs as the result of either an upcall request from an endpoint or a request from another managed node. After the gateway method executes, the results are passed back to the endpoint or to the managed node that made the call.

2.4.8.1 Downcalls

A method request that originates on a managed node (or higher) and executes on a TMA endpoint is termed a downcall. This occurs when the gateway invokes a method on an endpoint. In a downcall, the gateway routes a stub call to the endpoint, and an endpoint method runs on the endpoint as a result. The downcall originates from an object call made from any managed node in the TMR or from the TMR server.

The plus sign (+) denotes deferred authorization. That is, the authorization for the object takes place at the gateway. A plus sign also indicates that the object is transient in nature, and that the ID may be short-lived; they have no state and are referred to as abstract objects. In the case of a TMA endpoint, the endpoint object ID will remain the same for the life of the endpoint, the transience applies more to task methods, and so on. The object ID is constructed on the fly using the Class Prototype Object (CPO) as shown in 2.4.4, “Object Relationship” on page 41. These object IDs are used during downcalls. The equivalent for upcalls is the minus sign (-).

The endpoint receives the name of the program to execute, its arguments, and runs the program. After the endpoint method completes, the endpoint returns the results back through the gateway. The gateway returns the results back to the caller. If the requested method executable does not already exist on the endpoint, or does exist but is out of date (relative to the gateway), the method executable is downloaded to the endpoint. If the method has dependencies, such as libraries, they are also downloaded. The following brief sequence summarizes a downcall from a managed node to an endpoint:

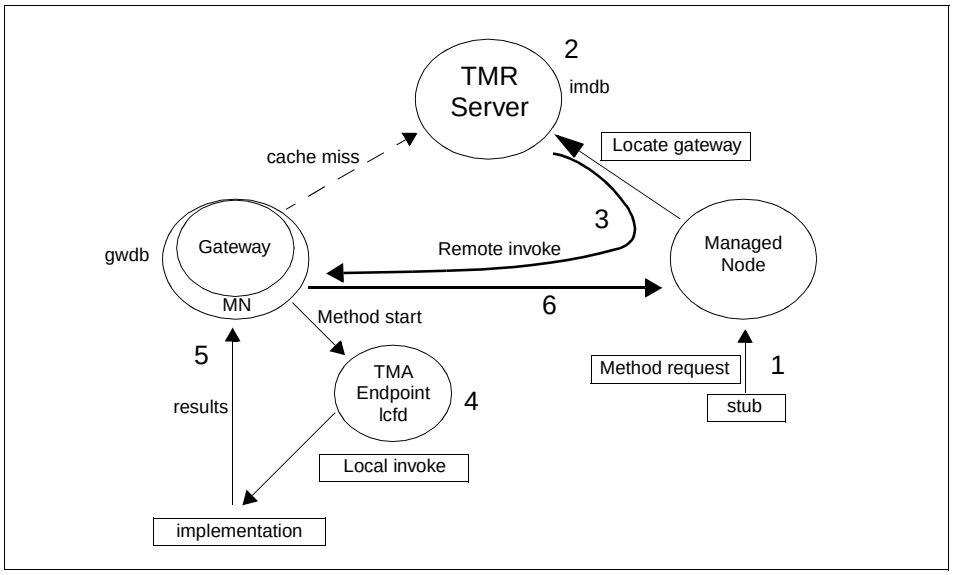


Figure 11. TMA Downcall Architecture

Note

The plus sign (+) in the object reference distinguishes that this is a downcall and, therefore, the object runs on an endpoint.

1. The client makes a request to perform an action on an endpoint - an `objcall` on a managed node, such as a scan operation.
2. The TMR server maps the dispatcher in the object to a gateway object ID. The gateway object ID is used to determine which managed node the gateway is running on.

3. The TMR object dispatcher sends the request to the client that made the request, and the client invokes the method on the gateway's managed node.
4. The gateway resolves the method on the corresponding behavior object and performs authorization of the invocation by the invoking Tivoli principal.
5. The method runs on the endpoint: The parameters are sent, the method runs, the MDist data is transmitted from the gateway if needed, and the results are passed back to the gateway.
6. The gateway then returns the results to the caller.

2.4.8.2 Upcalls

Upcalls occur when endpoint applications initiate Tivoli operations in the TMA environment. This section describes the sequence that occurs when an application makes an upcall, and the following section describes the TMA upcall architecture.

General information about upcalls

Upcalls are method requests that originate at the TMA endpoint. The endpoint can only invoke methods on the managed node associated with its gateway not on any arbitrary object in the TMR. This design maintains scalability: Upcalls are handled at the gateway without going to the TMR server each time because information obtained from the TMR server is cached at the gateway. The gateway can authorize and resolve the method invocation; so, the TMR server does not have to be involved. The upcall consists of a class name, the name of the method to be run, and the arguments for the method. The gateway then resolves the prototype object for the class name and constructs the object call to invoke the method.

The sequence of an upcall

Unlike a method invocation in the full framework, an upcall always runs a gateway method on the host machine of the gateway. The gateway attempts to resolve the method and do the necessary authorization without making a call to the object dispatcher on the TMR server. The method header can be retrieved from the method header cache, as it is for downcalls. The gateway tells the managed node object dispatcher to start a gateway method; the object dispatcher starts the daemon for the method (the upcall collector), if not already started, and the daemon then proceeds. Results are returned through the gateway daemon back to the endpoint.

This is the flow of control when an endpoint does an upcall, invoking a gateway method:

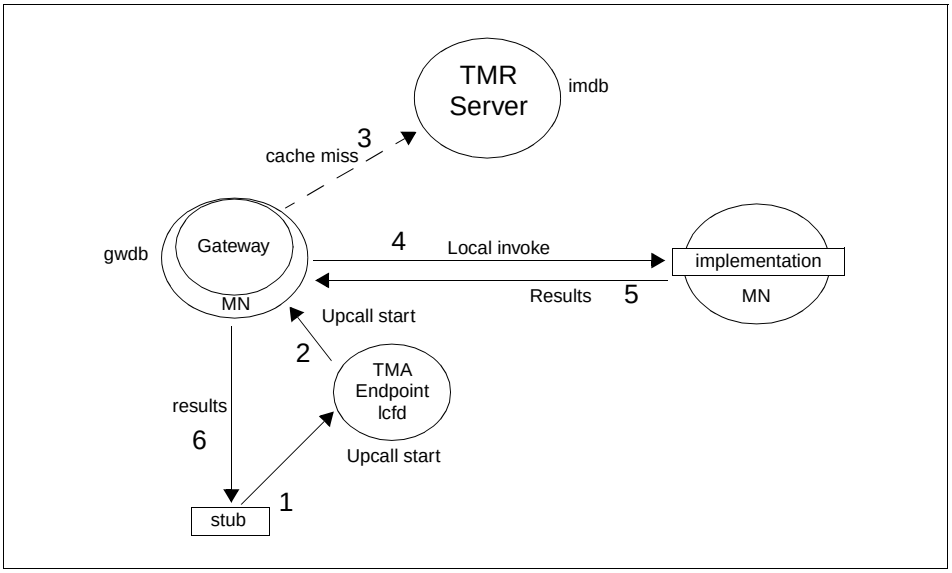


Figure 12. TMA Upcall Architecture

Note

The minus sign (-) on the object reference denotes that the endpoint initiated the upcall. See “Downcalls” on page 48 for a description of the plus and minus signs.

1. An application signals a request to run a method beyond the endpoint and gateway, such as logging a notice group item.
2. The endpoint sends the request to the gateway consisting of a class name, the name of the method to be run, and the arguments for the method.
3. If the gateway can resolve the method header, it tells the object dispatcher on the managed node to invoke the method. If it cannot resolve the method header, it goes to the TMR server to get the header and, when the header is returned to the gateway, the gateway tells the managed node to invoke the method.
4. The method is invoked on the gateway.
5. Results are passed back to the gateway.
6. The gateway passes the results back to the endpoint.

2.5 Troubleshooting Tips Using the Object Database

You should be able to use what you learn from this chapter about the object hierarchy and the tools provided with the Tivoli Framework to use `objcall`, `idlcall`, and `idlattr`.

Important

Unless you are very familiar with object attributes, or you are experimenting on a properly isolated test system, it is recommended that you use `objcall OID getattr` instead of the `idlattr` command. If using `idlattr`, be sure to always use the **get** (**-g**) flag. The default is otherwise to **set** (**-s**) the attribute, an activity that could alter the execution of a method. Recovery from such a situation may include the restore of a previous database backup. See also Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

2.5.1 Finding the Method Executable

If a method fails for some reason, you might try to find out more about it from the object database. You might also want to find the executable and verify, for example, that it is the correct one (easily done if a method works on one system but not another. You can then compare the two). You will probably do some or all of the following:

- Duplicate the problem or action.
- Trace with `odstat` and/or `wtrace`.
- Identify the method from the gathered documentation.
- Find the executable responsible for method:

  1. Execute the following command to find the behavior object (BO) OID.

     ```
     objcall <Object ID> resolve <method>
     ```

  2. Execute the following command to find the type of the method listed in the last line of the result of the command.

     The method type can be:

intrinsic	Method is built into the oserv.
default	Method is looked up for architecture of current system.
<platform type>	Method is custom for the specified platform.

     You can also find some attributes with this command, such as the default user ID, to use:

```
objcall <behavior object id> om_stat <method>
```

3. If you have a default method type, you can find the binary path of the method by executing this command:

```
objcall <behavior object id> om_get_definition <method> default
```

The following is an example of how to find the binary of a method:

```
# objcall 1264987995.2.7 resolve avail_space      !Find the Behavior object id
1264987995.1.324

# objcall 1264987995.1.324 om_stat avail_space    !to get the type of the method.
CATALOG=
SET_USER=         !User ID with which the method execute.
SET_GROUP=        !Group ID with which the method execute.
EXPORT=TRUE
EXECUTE=FALSE
default            !This is the type of the method.

# objcall 1264987995.1.324 om_get_definition avail_space default   !get the path of the
STORAGE=/TAS/MANAGED_NODE/man_node_skel1                          !binary of the method
MODEL=queued-obj-daemon
```

The binary file of the method will be at $BINDIR/TAS/MANAGED_NODE (in UNIX) or %BINDIR%\TAS\MANAGED_NODE (in Windows NT) and the executable in this example is *man_node_skel1.sh* (or .exe as appropriate).

2.5.2 If the Method is Unknown

If you know the class object you can use the following commands to find all the methods of the object:

```
# wls -od /Library/<resource>        !find the class OID (Instance Manager)
# idlattr -tg <class OID> behavior <object type>     !get the Behavior OID
# objcall <behavior OID> contents      !find all the methods of the object
```

If you have a managed node instance name, you can use the following commands to find all the methods and their parameters (note that this will list all the methods for the class specified, not including those available to an instance through inheritance):

```
# wlookup -r ManagedNode <instance>      !Find the name of the class object type
# irview <type name> contents            !Find the methods and their full names
# irview <type name>::<meth_or_at> describe !get the method description using
                                                !the full name of the method
```

A more extensive example of using `irview` is shown next. Here we can find the parameters of the method *avail_space*. The parameters are described below the ParameterDescription label:

```
bash$ wlookup -r ManagedNode itso2   !Get full class name of MN instance itso2
1212391543.1.347#TMF_ManagedNode::Managed_Node#

bash$ irview TMF_ManagedNode::Managed_Node contents   !example of method's full names
```

```
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::os_name
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::os_version
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::os_release
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::system_time
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::system_time_zone
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::memory_size
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::avail_space
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::create_profile
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::update_sub_label
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::set_local_label
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::is_supported_interface
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::set_policy_region_name
1212391543.1.4##4@TMF_ManagedNode::Managed_Node::get_cache_info
.....
..... !(remaining lines deleted)

bash$ irview TMF_ManagedNode::Managed_Node::avail_space describe !Get method description
OperationDescription
 name:           avail_space
 id:             TMF_ManagedNode::Managed_Node::avail_space
 defined in:     TMF_ManagedNode::SysInfo
 TypeCode: ulong
     kind:        tk_ulong
     to_orb_free: 0
     size:        4
     # parms:     0
 mode:           NORMAL
    ParameterDescription
     name:           directory
     id:             TMF_ManagedNode::Managed_Node::avail_space::directory
     defined in:     TMF_ManagedNode::SysInfo::avail_space
     TypeCode: string
         kind:        tk_string
         to_orb_free: 1
         size:        4
         # parms:     1
     mode:           IN
    ExceptionDescription
     name:           ExStdlib
     id:             SysAdminException::ExStdlib
     defined in:     SysAdminException
TypeCode: SysAdminException::ExStdlib
         kind:        tk_struct
         to_orb_free: 1
         size:        44
         # parms:     19
    ExceptionDescription
     name:           ExUsage
     id:             SysAdminException::ExUsage
     defined in:     SysAdminException
     TypeCode: SysAdminException::ExUsage
         kind:        tk_struct
         to_orb_free: 1
         size:        40
         # parms:     17
    ExceptionDescription
     name:           ExNotFound
     id:             SysAdminException::ExNotFound
     defined in:     SysAdminException
     TypeCode: SysAdminException::ExNotFound
         kind:        tk_struct
         to_orb_free: 1
         size:        36
```

```
        # parms:     15
```

Now that we have the method description including parameter and data type, how can we use it?

Well, knowing that the directory name is required as a string. Here is an example of using this non-intrusive method:

```
bash$ idlcall 1212391543.1.347 avail_space '"h:/"'

262440      !There is 262Mb free in drive H:

bash$ idlcall 1212391543.1.347 avail_space '"\tmp"'
{ USER_EXCEPTION SysAdminException::ExSystem { "Exception:UserException:SysAdmin
Exception::ExException:SysAdminException::ExSystem" "TasExCat" 17 "%5$t{%c} (%3$
d): system problem: '%7$s'" 912738032 { 0 } "ntfsinfo: exit 1: Can't cd to
mp: The filename, directory name, or volume label syntax is incorrect." }}
```

Therefore, by using an invalid path parameter (`\tmp`), we discover that the error is returned from ntfsinfo, a tool found in `$BINDIR/tools` for NT. In this example, ntfsinfo was called by the binary `man_node_skel1` of the method avail_space.

2.5.3 Method Errors

There are two common errors that occur with methods that are incorrectly setup: NO_METHOD and NO_IMETH. These errors are generally found on custom methods. Use the following steps to look into these two types of errors:

1. Use `odstat` to identify the method having problems. See Section 6.1, "The odstat Command" on page 132 for more information about this command.
2. Use `objcall <object id> resolve <method>`. If the object cannot find the method, there is something wrong with its inheritance. Check its class Extended Behavior Object whose OID is in the extension attribute of the class object and Behavior Object (OID in behavior attribute) for the listing of methods.
3. Use `objcall <behavior id> om_stat <method>` to find the type of the method. The method type can be:

   intrinsic Method is built into the oserv.

   default Method is looked up for architecture of current system.

   <platform type> Method is custom for the specified platform.
4. So long as the type is not intrinsic, use `objcall <behavior OID> om_get_definition <method> <type>` to list the storage location for the method in relation to $BINDIR.
5. Check the permissions and the name of the file (for custom methods).

There are two identities a method uses when executing:

ACL Identifies which administrative role is needed to run the method.

ID Identifies the UNIX user ID the method uses to run.

To find the ACL for a method, you can use the following command:

```
# objcall <behavior object id> om_get_acl <method>
```

It will give you the valid roles for the method.

To find the UNIX user ID and group ID with which the method will run, you can try the following command:

```
# objcall <behavior object id> om_stat <method>
```

The SET_USER and the SET_GROUP are the values that are used. These will either be set with a name, such as nobody, or they will contain an asterisk (*) meaning they will use the user ID and group ID passed by the caller.

2.6 Object Tools Summary

The following is a list of main object commands:

- `objcall`. Used on objects to execute methods:

  ```
  # objcall <object id> <method> <arg>
  ```

- `idlcall`. Used to execute IDL/TEIDL methods:

  ```
  # idlcall <object id> <method> <args>
  ```

- `idlattr`. Used to get and set attribute values:

  ```
  # idlattr -t [g,s] <object id> <attr_name> <attr_type>
  Note: Set (-S) is the default.
  ```

If the `objcall` method fails, try using `idlcall` or `idlattr` to execute methods or get/set the values of attributes.

The following is a list of commands used to get object IDs and method or attribute names:

`odstat` See Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

`wtrace` See Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

`wlookup` Looks up an instance of a resource from the name registry input:
`wlookup -R`
`wlookup -ar <resource>`

`wlookup -r <resource> <instance>`
The output is: `ObjectID#corba-description# instance`

`wls` Looks up an object in the local TMR database:
`wls [-odl] [Path] wls [-odl] /Library/<resource>`

The ADE manuals provide more information about methods and attribute types. There is also the command called `irview` that can display information about methods and attributes:

```
irview repository-id [contents|describe|describe_contents|check_consistency]
```

For more information about these commands, please see the *Tivoli Framework Reference Manual*, the on line manual pages (`man` *`command`* in UNIX or `START \tivoli\man\w32-ix86\man\MPG.HLP` in Windows NT) or Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

Chapter 3. The Tivoli Core Installation Process

Due to the (mostly hidden) complexity of the Tivoli products, the installation process must accomplish a great deal, such as determining the correct code to load on different interpreter types. With the introduction of the new Software Installation Service (SIS - Chapter 4, “Tivoli Software Installation Service” on page 83), installing clients and applications has become easier. However, it still helps to understand the original install process in some detail for a number of reasons:

1. The TMR server cannot be installed with SIS.
2. SIS still invokes most of the standard install process under the covers.
3. An install of the occasional single machine does not warrant the use of SIS.

We also use this install chapter to describe the layout of the Tivoli CDs.

3.1 Installation Overview

Several components must be installed to create the Tivoli Enterprise environment. Because the Tivoli server’s oserv coordinates communication between the oservs on each Tivoli client, the TMR server must be installed first:

TMR server Either the `wserver` command or `setup.exe` (Windows NT) can be used to install the Tivoli server.

Tivoli clients A remote Tivoli installation can be performed on clients (managed nodes) from the TMR server by using the `wclient` command or the Tivoli Desktop. The Software Installation Service (SIS) can install clients once the TMR server is installed.

Tivoli endpoints Installation of TMA endpoints can be performed using `winstlcf` command, or the Tivoli Desktop, or by local installation on the endpoint.

Tivoli applications Each Tivoli application must be installed on each TMR server and each individual client that will use it. Application installations can be made with either the `winstall` command or the Tivoli Desktop. SIS can also be used to install applications.

Tivoli patches Patches must be installed on each TMR server and each individual client that will use it. Patch

installations can be performed using the `wpatch` command or the Tivoli Desktop. SIS will also install patches.

The interfaces for all install components send information to the same installation engine.

Important

It is extremely important that backups be performed immediately before and immediately after a product installation or before a major maintenance procedure, such as the creation of several managed nodes. These backups should be kept on a separate tape and in a secure place as part of a comprehensive backup and recovery strategy.

3.2 General Pre-Install Checks, Hints, and Tips

There are a number of things you should do before installing a server, client, or application. For all installations you should:

- Read the product release notes.
- Back up your Tivoli database (as well as performing any normal system backups).

General Tips for Installs

- Do not use the c-shell for installing on UNIX systems.
- Do not try and install across TMR boundaries. Always install applications in the local TMR.
- If a previous install has failed, you can specify an exclamation mark (`!`) at the beginning of the path names in the install dialog. This will force an overwrite for any existing directories.
- If you have not created the Tivoli install directories prior to starting the installation, remember to select that directories should be created. When the dialog appears, the check box is not selected by default.

This section provides other hints for things to check when installing either a Tivoli server or a client on UNIX and Windows NT.

3.2.1 UNIX

The install process performs some space checking once the install gets going, but you will save a lot of time if you check for adequate Tivoli code and database file system space in advance. To make your system easier to manage, you may want to define some new file systems for Tivoli. You have to ensure that your file systems are large enough to contain all the Tivoli files (refer to the product release notes and user manuals to determine file space requirements). Tivoli, by default, will install most of its files into /var and /usr. There are a number of reasons why you may want to set up specific Tivoli file systems:

- You avoid problems where other applications may fill up space in /var and /usr file systems.
- You can back up and restore individual file systems defined on your system, although this may still be a little complex for Tivoli products.
- You can control the overall disk structure and layout.

Default directories created are:

/etc/Tivoli	This directory is small at install time and can be left as part of the /etc file system.
/var/spool/Tivoli	Make a new file system for this and specify that it should be mounted at system restart.
/usr/local/Tivoli	This is the largest of the directory trees created by Tivoli. Create the file system and specify that it should be mounted at system restart.

Tivoli will also write install and other log files to /tmp.

The *Tivoli Framework Planning and Installation Guide* also contains details on how to check for sufficient system swap space and process slots for various UNIX types.

3.2.2 Windows NT

The drive where you want to install Tivoli must be formatted with NTFS. You can check this from the My Computer window. Right-click on the desired drive icon and select **properties**. The general page includes the file system type. You can also check this from an NT command prompt with `CHKDSK d:` (where `d:` is the drive where you will install Tivoli). You can convert a FAT file system to NTFS using the `convert` utility. See the Windows NT online help for more information.

Tivoli files for Windows NT are, by default, stored under the \Tivoli directory on the root of the selected drive.Tivoli will also write install and other log files to `%DBDIR%\tmp`.

You should verify through the Control Panel system applet that you have sufficient swap space defined. The *Tivoli Framework Planning and Installation Guide* discusses this requirement.

3.2.2.1 Tivoli Remote Execution Service

You have to locally install the Tivoli Remote Execution Service (also known as the Tivoli Remote Installation Package or TRIP) from the Tivoli Framework CD on one Windows NT system before you can install an NT managed node or TMA endpoint from a UNIX TMR server. This system will then be identified to Tivoli by the variable name CurrentNTRepeat. All subsequent Windows NT installs will copy the TRIP binaries from this machine. Note that it may be necessary to install TRIP manually on another machine or reconfigure CurrentNTRepeat if you wish to install a Windows NT system that may have access problems due to domain security configuration.

3.2.3 NFS Mounts

NFS mounting of binaries can save time in installing with maintenance of patches and system backups. The mounts can be made read-only to clients as long as there is at least one machine for each architecture that has root write access to the binaries. All managed nodes using tasks must have root write access also.

If problems occur with a new installation, make sure the CD-ROM is mounted on the TMR server.

During an installation, the install engine should return the name of the NFS server where the files will be placed. For example, if the client is *spot*, and the NFS server is *decimal*, the install message will look like the following:

```
[Installing product: Tivoli/Sentry 2.5 Monitors

Unless you cancel, the following operations will be executed:
  For the machines in the independent class:
    hosts(spot)
    need to copy the machine independent Binaries to:
         decimal:/decimal/data1/Tivoli/rainbow/25/bin/generic
```

Note

Never share NFS-mounted binaries across machines in different TMRs.

3.2.4 Environment Files and Variables

Tivoli installation can be affected by /etc/wlocalhost**.** This is an executable used to identify a machine when a name is specified that is not returned from `uname -a`. When using the server's full IP name, the .rhosts file must be set up correctly, or the system will perform a remote server install.

The install also uses two environment variables, `EtcTivoli` and `o_dispatch`, which are used for the location of the /etc/Tivoli directory and the TCP/IP port where the oserv will run respectively. Modification of these variables is NOT supported by Tivoli.

3.2.5 Automatic Startup versus Remote Startup

Automatic startup creates or modifies system startup files to automatically start the oserv whenever the system is rebooted.

Remote startup requires the use of the `odadmin` commands to stop and restart the oservs in your TMR from one location. There are two files modified in UNIX: `/etc/services and /etc/inetd.conf`.

3.2.6 Deciding When a Reinstall is Best

When did your last Tivoli database backup occur? Many problems can be solved by restoring just the server's database to a known stable state. Difficult client problems can also be resolved faster through reinstallation. Most customizations are done on the server. Distributions with exact copy options can reduce recovery time.

Reinstall Tip

If you don't have your license key handy during a reinstall, use `odadmin get_platform_license > filename` to make the oserv give you the key before you remove the old installation. You can then cut and paste it from `filename` during the new install process.

3.3 Tivoli Server Installation

The *Tivoli Framework Planning and Installation Guide* provides plenty of information on the actual install process. Here, we list the most important items to remember when performing a TMR server installation.

Before installing the Tivoli Framework on the server, the user must have determined the following:

- Login ID with access as root (for UNIX) or Administrator (for NT).
- A TMR name and server name.
- The TMR installation password (optional).
- The Tivoli license key.
- The path(s) where the files will reside.
- For Windows NT, a Tivoli remote access account (TRAA) if binaries will be shared over the network. See Section 3.4, “Tivoli Client Installation” on page 65 for more information regarding TRAA.

Once the **Install** or **Install and Close** buttons are chosen, the information is verified, and the installation begins.

Notes for Windows NT Installs

- It is more secure if the TRAA used for sharing binaries in a Windows NT domain is a user with less privileges than the local Administrator’s account.
- Do not install the TMR server on a Windows NT Primary Domain Controller (PDC). It can cause problems if you use the domain administrator account and password for installs.
- Use the local administrator account on machines in a Windows NT domain.
- The first NT system installed with TRIP in a TMR is identified by the variable `CurrentNtRepeat`. TRIP is copied from this machine to other NT systems during the install. This variable may need to be changed if there are access problems between this machine and new clients.

3.3.1 Server Install: Behind the Scenes

The server installation performs the following:

- Transfers binaries, libraries, man pages, and message catalogs to the server.

- Installs a template of the object database and modifies it to include the server's name, region number, interpreter type, and so on. We can see this activity in the oservlog of a new server installation as shown in Figure 13:

```
Nov 06 16:08:21: $converting odlist region numbers (2099999999 -> 1360991896)
Nov 06 16:08:21: $changing ALI's secret key to new random value.
Nov 06 16:08:22: $Database mismatch (from stout/146.84.27.11)
Nov 06 16:08:22: $Migrating ALI.
Nov 06 16:08:22: $changing encryption type from none to simple
Nov 06 16:08:22: TME 10 Framework (tmpbuild) #1 Thu Oct 3 08:08:58 CDT 1996
Copyright Tivoli Systems, an IBM Company, 1996.  All Rights Reserved.
TMR 1360991896. ORB 1. TMR server local:94. Port 94.
pid 24052
```

Figure 13. Sample Output from the oservlog After New Install

The following is a list of some of the attributes that changed during the server installation:

- Region number
- Secret key (ALI refers to the TMR server)
- Hostname
- IP address
- Encryption level

The default object database shipped with the install code includes a place-holder machine name. This name (stout in this case) appears in this file. You might see this name when looking directly at objects in the object database. This name is replaced with the user-supplied name in externalized objects. If the server icon on the desktop does not show the correct server name, then the installation did not complete and was, therefore, unsuccessful.

3.4 Tivoli Client Installation

The *Tivoli Framework Planning and Installation Guide* provides plenty of information on the actual install process. Here, we list the most important items to remember when performing a Tivoli managed node installation.

The user provides the following information to begin a client installation:

- A root (UNIX) or Administrator (Windows NT) ID and password or equivalent. Trusted host access is not recommended in UNIX systems because it can compromise general network security.

- The TMR installation password if one was used during the Tivoli server installation.
- The client's name.
- The installation location.
- For Windows NT clients:
  - The first Windows NT machine must have Tivoli Remote Execution Service (also known as TRIP—Tivoli Remote Installation Protocol) installed and running before a client installation can take place. It enables remote operations that would normally use UNIX `rexec` to be performed. See also "Tivoli Remote Execution Service" on page 62.
  - A remote access account must be established to allow the Tivoli client to access the remote Tivoli binaries. The Tivoli Remote Access Account (TRAA) is established either through the GUI installation procedure or by using the `wsettap` command. The `wsettap` registers the TivoliAP.dll in the NT LSA registry key:

    HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa

    Some products (such as Banyan Vines) have been known to add a blank entry under LSA. If TivoliAP is added after that blank line, it will not be found. Removing the blank entry resolves the problem. Refer to Appendix A.2.4, "Understanding the Tivoli Remote Access Account" on page 599 for more information.

Once the **Install** or **Install and Close** buttons are chosen, the information is verified, and the installation begins.

3.4.1 Client Install: Behind the Scenes

While the user waits for the client installation to complete, Tivoli performs the following actions. The pre-installation and installation steps are detailed below.

3.4.1.1 Pre-Installation steps

- On Windows NT clients:
  - TRIP is installed on the client enabling the execution of commands on Windows NT clients from another managed node and `rexec` communication with the TMR server.

    On the NT machine, a drive is mapped to a previously installed managed node, the TRIP binaries are copied, and the service is installed. The first NT system in the Tivoli environment with TRIP

installed is designated the NT repeater. The repeater can then distribute TRIP to any other NT clients in the TMR.

- On all clients:
  - Tivoli determines the client interpreter type.
  - Checks for software dependencies, if any.
  - Checks for adequate disk space to store binary, library, and database components.
  - Checks for previously installed software. A file with a product-related name is stored in various .installed directories after a successful installation. These directories exist for binaries, libraries, message catalogs, and other components. If the installation engine finds previously installed components, it will not install them again. The following shows some of the .installed files. Note that a product will not necessarily have files installed in all areas:

```
#ls $DBDIR/.installed $BINDIR/.installed $LIBDIR/.installed
/usr/local/Tivoli/bin/aix4-r1/.installed:
Inventory_BIN          pa_3.1.2-TMP-0001_BIN  pa_TMF_3.1_SP1_BIN
TMF_BIN                pa_Inventory_3.1_BIN
pa_3.1-TMP-0033_BIN    pa_TMF_3.1.2_BIN

/usr/local/Tivoli/lib/aix4-r1/.installed:
TMF_LIB                pa_TMF_3.1.2_LIB
pa_3.1.2-TMP-0001_LIB  pa_TMF_3.1_SP1_LIB

/var/spool/Tivoli/rh0255c.itsc.austin.ibm.com.db/.installed:
Inventory_ALIDB          pa_3.1.2-TMP-0001_ALIDB  pa_TMF_3.1.2_ALIDB
TMF_ALIDB                pa_Inventory_3.1_ALIDB   pa_TMF_3.1_SP1_ALIDB
```

3.4.1.2 Installation Steps

Once the pre-requisites for the install have been checked, the user sees the Unless you cancel message explaining what will take place during the install. If everything is correct, and the user selects to continue installing, then the following activities take place:

- The NT client asks for the TRAA password.
- Transfers binaries, libraries, man pages, and message catalogs to client. The connection takes place through `rexec` or `rsh`.
- Creates the database directory and creates the database by doing the following:
  - Creates the file_versions directory.
  - Runs `install2.cfg`, which performs the following substeps:

- Starts oserv on the client (creating oservlog):

  `oserv -i -h <TME_host>`

- Puts the client IP address in the server's odlist.
- Creates three client objects in the TMR database.

When completed, it contacts the installation script to continue the installation.

- Configures the database by performing the following substeps:
  - Runs `client.cfg` on the client.
  - Creates database objects. On clients, the database is created from scratch. On the TMR server the database is created from a pre-defined database file.

3.4.2 Reinstalling Clients

Any partially installed clients must be removed before they can be reinstalled.

3.4.2.1 To Remove Tivoli from a Windows NT Client

To remove Tivoli from a Windows NT client, perform the following steps from the NT client:

1. Remove the oserv service from the NT service manager:

   `oinstall -remove`

2. Remove TRIP from the NT service manager:

   `trip -remove`

3. Remove the TAP internal key and unregister the TivoliAP.dll with the local security authority. If you are going to reinstall a client using a different installation password, remove the TivoliAP.dll from the `%SystemRoot%\system32` directory because Tivoli will not overwrite an existing file:

   `wsettap -d`

4. Remove the NT client code from the TMR server. See the next section, 3.4.2.2.

3.4.2.2 To Remove a Partially or Fully-Installed Client from the TMR

To remove a partially or fully-installed client for the TMR, perform the following steps:

1. Determine where, or if, a client is installed with one of the following commands:

- `wlookup -ar clientname`
- `wls /Library/clientname`
- `odadmin odlist`

2. Make sure the oserv is not running on the client.
3. Perform one or more of these steps in order from the server until the client is successfully removed:
   - `wrmnode` *clientname*

     Removes the specified client from the Tivoli database.
   - `wrmnode clientname -d dispatcher-number`

     Shuts down the dispatcher of the specified managed node and removes it from the Tivoli database. The dispatcher number can be obtained with the `odadmin odlist` command.
   - `odadmin odlist objects dispatcher-number`

     Displays the object IDs of the objects owned by the dispatcher. If there are less than three objects, run the following to remove the dispatcher and its objects from the TMR. References to the objects will still remain.
   - `odadmin odlist rm_od dispatcher-number`

     Removes the node.
4. Run `wchkdb -u` to update the Tivoli resource database.
5. Remove the client's database directory.
6. The client can now be reinstalled.

3.4.3 Uninstalling a PC Agent

The removal of the PC Agent involves two actions:

1. Removal of the Tivoli code on the machine
2. Deletion of the PC managed node pointing to the machine

3.4.3.1 Removal of the Tivoli Code on the Machine

This task is very simple. Just delete the directory - along with all subdirectories - in which the PC Agent was installed. Usually, this directory is C:\Tivoli\TmeAgent.

Before removing the TmeAgent directory, references to the programs should be removed. On Windows NT services, definitions will have to be removed. On Windows NT and 95 TME Agent Registry, information should be removed,

and for all TME Agents, automatic start-up of programs may have to be removed.

Removing services

For Windows NT this task is accomplished using the INSTSVC program in C:\Tivoli\tmeagent\win32. To remove a service use the command: `C:\tivoli\tmeagent\win32\instsvc -r`

Removing TME agent registry information

On Windows NT and Windows 95, the TME Agent registry subkey in the `HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli` registry path should be removed.

Removing autostart

Windows xx:

Remove the tmeagent registry key in the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` registry path

OS/2:

Remove references to `c:\tivoli\tmeagent\tivos2.exe` program from either `c:\startup.cmd` or the tcpexit.cmd in `c:\mptn\bin\tcpexit.cmd` or `c:\tcpip\bin\tcpexit.cmd`.

Having removed all references to the TME Agent programs, the TME Agent directory should be deleted.

3.4.3.2 Deletion of the PC Managed Node

Use the `wdel @PCManagedNode:<pc-managed-node-label>` to delete the definition of the PC managed node related to the TME Agent just removed.

3.4.4 Uninstalling a TMA Endpoint

Removal of a TMA endpoint falls into two parts:

1. Remove the TMA code and references to it from the machine on which it is installed.
2. Remove the Endpoint definition from the Endpoint Manger.

3.4.4.1 Removing TMA Code

To remove the installation of a TMA endpoint from a Unix machine, the only actions required are to remove the binaries and directories in which the TMA code is installed. Also, any references to these files should be removed. Before removing the files and directories, the TMA endpoint should be stopped. The necessary steps are:

Stop the endpoint Run the `lcfd.sh stop` command

Remove the code Use the `rm` command. The typical location for the TMA endpoint would be /usr/local/Tivoli/lcf

To remove a TMA endpoint from an Intel machine, the same actions apply with the additional step of stopping and removing the `lcfep` program running on Windows 95 and NT. During installation, an uninstall command-file is build somewhere in the installation path:

Windows `%LCF_ROOT%\uninst.bat`

OS/2 `%LCF_ROOT%\bin\%interp%\mrt\uninstal.cmd`

Before uninstalling, all Tivoli application should be stopped. For all applications, except Distributed Monitoring, this is handled in the uninstall script by stopping the `lcdf` program that most applications rely on. To explicitly stop the Distributed Monitoring before uninstalling, use the `wstopeng` command.

For Windows NT and Windows 95 TMAs, it is also recommended that the registry key - holding the name of the subdirectory of %LCF_ROOT%\DAT in which the current configuration files are held - is deleted. The following key, and all subkeys, should be removed from the registry:

`HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\lcfd`

3.4.4.2 Remove the Endpoint Definition

To delete all information in the TMR regarding the endpoint just removed, it should be removed form the Endpoint Manager. This can be obtained using the GUI or the `wdelep <endpoint>` command.

3.5 Finding Out What's Installed

The following commands can be used to determine which files have been installed in the TMR:

- `wlookup -ar` with a parameter of either `ProductInfo` or `PatchInfo` lists the products and patches installed in the TMR server:

```
#wlookup -ar ProductInfo
Inventory       1360991896.1.500#TMF_Install::ProductInfo#
Inventory_3.1   1360991896.1.551#TMF_Install::ProductInfo#
Tivoli_Inventory_PC_Scanning_Program    1360991896.1.561#TMF_Install::ProductInfo#

#wlookup -ar PatchInfo
3.1-TMP-0033    1360991896.1.493#TMF_Install::PatchInfo#
3.1.2-TMP-0001  1360991896.1.573#TMF_Install::PatchInfo#
Inventory_3.1   1360991896.1.538#TMF_Install::PatchInfo#
TMF_3.1.2       1360991896.1.486#TMF_Install::PatchInfo#
TMF_3.1_SP1     1360991896.1.479#TMF_Install::PatchInfo#
```

The list is sorted alphabetically. Note that because Tivoli never reuses object numbers, the last part of the OID can tell you the order in which the products and patches were installed. In this list, TMF_3.1_SP1 (479) was installed first, and 3.1.2-TMP-0001 was installed last (573).

- The `wlsinst -ah` command lists the installed products and patches and the host names and interpreter types of the machines on which they were installed (this data is parsed from that you can also see from `idlcall $OID _get_locations`):

```
                          Product List
*-----------------------------------------------------------------*
TME 10 Framework
    rh0255c.itsc.austin.ibm.com aix4-r1
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    rh0255c.itsc.austin.ibm.com aix4-r1
*-----------------------------------------------------------------*
                           Patch List
*-----------------------------------------------------------------*
TME 10 Framework Patch 3.1-TMP-0033
    k124a          aix4-r1
    rh0255c.itsc.austin.ibm.com aix4-r1
TME 10 Framework Patch 3.1.2-TMP-0001
    rh0255c.itsc.austin.ibm.com aix4-r1
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255c.itsc.austin.ibm.com aix4-r1
TME 10 Framework Maintenance Release 3.1.2
    rh0255c.itsc.austin.ibm.com aix4-r1
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    rh0255c.itsc.austin.ibm.com aix4-r1
TME 10 Framework Version 3.1 Service Pack 1
    rh0255c.itsc.austin.ibm.com aix4-r1
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    k124a          aix4-r1
    rh0255e        w32-ix86
    rh0255c.itsc.austin.ibm.com aix4-r1
```

- The `image_report` program in $BINDIR/TAS/INSTALL lists the Tivoli products or patches, on the CD-ROM, to be installed in the TMR. Specify the name of the index file in the CD-ROM directory when using this command:

```
#$BINDIR/TAS/INSTALL/image_report
Usage: image_report [-a regex] [-c cdrom-dir] [-f] [product [product-2] ...]
    -a   only report on media packets architectures that match the given regex.
         for example, to select the sunos4 specific packets, pass:
               -a '(sunos4)|(generic)'
    -c   cdrom image directory [defaults to the current working directory]
    -f   give full listing of the contents of each media packet

    the product list is a list of the install index names and does not need
    the .IND extention.  If no products are given on the command line, all
    install indexes found in the cdrom image directory are listed.

#ls
ade_31_s.ind  file12.pkt    file17.pkt    file21.pkt    file26.pkt    file4.pkt
file9.pkt     cfg           file13.pkt    file18.pkt    file22.pkt    file27.pkt
file5.pkt     patches.lst   file1.pkt     file14.pkt    file19.pkt    file23.pkt
file28.pkt    file6.pkt     tmf_31_s.ind  file10.pkt    file15.pkt    file2.pkt
file24.pkt    file29.pkt    file7.pkt     file11.pkt    file16.pkt    file20.pkt
file25.pkt    file3.pkt     file8.pkt

#$BINDIR/TAS/INSTALL/image_report -a solaris2 tmf_31_s
/mnt14478
                          Installation Media Report
                 TME 10 Framework Version 3.1 Service Pack 1
     TMF_3.1_SP1 Package   Platform    Size   Compress Ratio    Date Built
     File (Install check file)         (MB)    Size
    ==========================================================================
    Libraries             solaris2     1.75    0.82    2.14    2/18  4:10
    FILE1.PKT ()
    Server Database       solaris2     0.30    0.09    3.33    2/18  4:11
    FILE15.PKT ()
    Client Database       solaris2     0.00    0.00    1.03    2/18  4:11
    FILE22.PKT ()
    Binaries              solaris2     3.04    1.75    1.74    2/18  4:11
    FILE8.PKT ()
    --------------------------------------------------------------------------
    Product total                      5.09    2.66    1.9

#$BINDIR/TAS/INSTALL/image_report -a aix4-r1 ade_31_s
/mnt14478
                          Installation Media Report
                    TME 10 ADE Version 3.1 Service Pack 1
      ADE_3.1_SP1 Package   Platform    Size   Compress Ratio    Date Built
      File (Install check file)         (MB)    Size
    ==========================================================================
    Binaries              aix4-r1      0.09    1.87    0.05    2/18  4:11
    FILE13.PKT ()
    Header Files          aix4-r1      0.15    0.09    1.73    2/18  4:11
    FILE20.PKT ()
    Libraries             aix4-r1      1.96    1.05    1.87    2/18  4:11
    FILE6.PKT ()
    --------------------------------------------------------------------------
    Product total                      2.20    3.00    0.7
#
```

3.6 General Troubleshooting Tips for Installation Problems

This section details items to check when problems occur during the installation process.

3.6.1 Common Errors

Check for the following possibilities common to most types of install:

- Error `e=5` - permissions

  The most common cause of this error is that you have created the Tivoli Install directories manually, and some or all of the directory permissions in the path are incorrect. This comes from the underlying operating system. System error code `5` is usually an access denied message such as could not run a method as nobody.

- Error `e=1` - wrong usage:

  A system call was incorrectly initiated, or an invalid call was made. Check tivoli.cinstall file for messages.

- Error `s=9` (or `6` on HP) - library path problem:

  The exit errors (`e=`) are usually errors Tivoli received from the system or some other application. You may be able to use system documentation to obtain more information about the cause of the errors. In Windows NT, try `NET HELPMSG n` where `n` is the number following `e=`. In many cases, the error can reflect a problem that happened on another system. When this happens, the system error you receive locally may not be as meaningful.

- Directory permission:

  For example, in UNIX /dev/null, /usr, /usr/local and /usr/local/Tivoli (if /usr/local or /usr/local/Tivoli already exist) must all be `rwxr-xr-x`. Other products may change the permissions as Oracle does for /dev/null.

- Lack of space:

  Check space and permissions on `$DBDIR` and `/tmp` for UNIX and `%DBDIR%` for Windows NT.

3.6.2 Windows NT Specifics

Refer also to Appendix A., “Tivoli’s Use of Windows NT” on page 595.

If you have installed TRIP, and you still have problems installing the Windows NT system as a managed node, use the following checklist:

- Is the Tivoli Remote Execution Service running?

  This can be checked by looking at the Services dialog in NT accessed through the Control Panel. The following diagram shows that the service is automatically started, and that it is currently running:

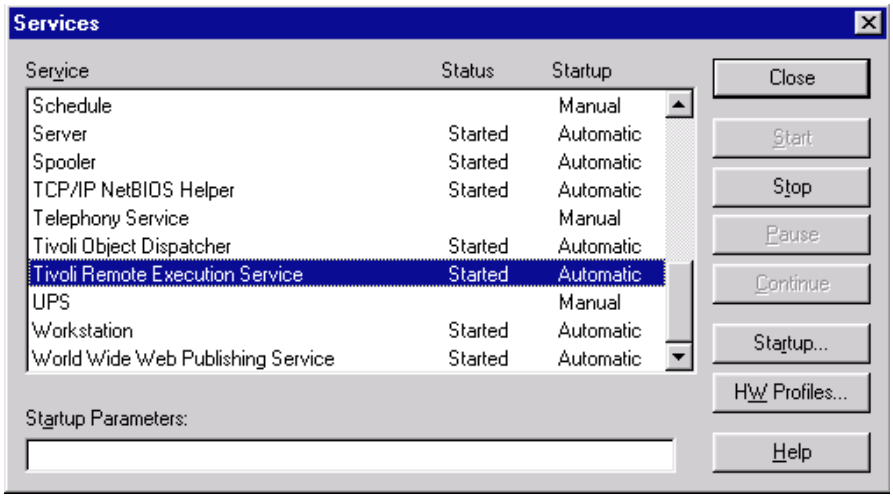


Figure 14. Windows NT Services Dialog

You can also check from a command prompt with the command `net start`. `Tivoli Remote Execution Service` needs to be listed in the output. If it is not, you can type `net start trip` to start it.

TRIP can be run in a debug mode. Go to the services dialog and stop the Tivoli Remote Execution Service or type `net stop trip` at a command prompt. Start the TRIP executable from a command prompt using `trip -debug`.

Note that if you have a different application running an `rexecd`, such as Exceed, TRIP will not start, and the Tivoli install process will fail.

- Is the tmersrvd ID defined to the NT machine?

  Use the Start>Programs>Administrative Tools>User Manager for Domains dialog or use the `net user` command to check if the ID exists. If not, then add the ID with the following attributes: ID is tmersrvd, PW as no password required, password never expires and do not prompt to change PW. Add it to the global **Domain Users** group. You can also create this account by cloning the standard Windows NT guest user ID. This ID must be able to log on locally.

- Check that the Tivoli_Admin_Privileges local group was added to NT through the same dialog as above or by using the `net localgroup` command.

3.7 Additional Troubleshooting for a TMR Server Installation

The following table highlights the log files written during a TMR server installation:

Table 1. Files Written During Server Installation

File	UNIX Directory	NT Directory	Location
tivoli.sinstall	/tmp	%DBDIR%\tmp	Server
oservlog	$DBDIR$	%DBDIR%	Server
install.cfg.error install.cfg.output	/tmp	%DBDIR%	Server

3.7.1 Common Server Install Problems

The following is a list of common server installation problems:

- Copying CD-ROM

  Use `wcpcdrom` in the original install directory where `wpreinst.sh` was executed:

  ```
  wcpcdrom /cdrom /cdrom.shadow
  ```

  This creates a directory tree of soft links pointing to /cdrom. Small files (.cfg, .ind) are copied, so they can be edited if required.

  There have been cases where `wcpcdrom` alone does not create the right letter case for files causing a problem when the link is used during installation. If you receive file-not-found messages after using `wcpcdrom`, you need to check for this problem.

  If you want to create a disk-based image to install from, the recommended process is to use the Software Installation Service (SIS). The steps below provide another way to create a CD-ROM image of the Framework on your disk if you still need to do so. This is not supported by SIS:

  ```
  # cd /
  # mkdir temp.dir
  # wcpcdrom /cdrom /temp.dir
  ```

  Note: This `wcpcdrom` command just creates links.

  ```
  # mkdir /TME3
  # cd /temp.dir
  # tar -chf - . * (cd /TME3; tar -xvf -)
  ```

  Note: This copies the image to disk.

  ```
  # cd /
  # rm -r temp.dir
  ```

Where /temp.dir is a temporary directory, /cdrom is the path to your CD-ROM device, and /TME3 is the final destination of the image.

There is a file called file0.tar created here, which you can use in place of the `WPREINST.SH` command.

Copy file0.tar to a temporary directory and then untar it. This basically creates the same structure as `WPREINST.SH` would, and all you have to do to install a TMR server is run `./wserver -c /<temp dir>` from the temporary directory.

- Host name:

  If you install the server using the fully qualified domain name, you will have to set up a `.rhosts` entry for the server; otherwise, Tivoli thinks that a remote install is being performed and uses `rsh`.

3.8 Additional Troubleshooting for a Client Installation

The following table highlights the log files written during a TMR client installation:

Table 2. Files Written at Installation Time

File	UNIX Directory	NT Directory	Location
tivoli.cinstall	/tmp	%DBDIR%\tmp	Server
oservlog	$DBDIR$	%DBDIR%	Client & Server
install2.cfg.error install2.cfg.output	/tmp	%DBDIR%\tmp	Client
client.cfg.error client.cfg.output	/tmp	%DBDIR%\tmp	Client

3.8.1 Common Client Install Problems

The following is a list of common client installation problems:

- Resolving names:

  Try to `rlogin` back and forth. Check the IP address and hostname and check for `/etc/wlocalhost`.

- Multiple network interfaces on the client:

  Most platforms include a `netstat` command that can be quickly used to check the local configuration.

  Look for a cannot map hostname error in oservlog.

- CD not mounted to the TMR server:

  The installation of a client is driven by the server and requires numerous server updates. It is possible to mount the CD at the client and start the install, but due to the interaction with the server, this could fail. The most likely reason would be a slow connection between the client and the server, thus causing media packet failures.

- Encryption key:

  Indicated by a `could not encrypt/decrypt data` error in oservlog**.** The user was denied authorization due to missing or wrong password. Retype the password, fix the `odadmin set_install_pw` command or start the oserv specifying the install key using `oserv -s install_key`.

3.9 Installation CD-ROM Contents

A Tivoli product CD-ROM contains the following types of files:

.IND Index files. List what components are on the CD for each interpreter type.

.PKT File packets. Tivoli Software Distribution-type packages containing the binaries. These can be expanded with `sapack`. See 3.9.1, “CD-ROM Installation Tools” on page 80.

.CFG Configuration files. Descriptions of the .PKT files and location of any before and after scripts.

.LST Contents or patch list. Used to build list of items available for install for the GUI.

The following shows what index, configuration, and contents files look like. (This example is from a 3.1 release CD. The format is the same for later releases). You can use this information to find out what components are installable on what interpreter types or whether libraries are included in the install:

```
#grep aix tmf_31_s.ind
 TMF_3.1_SP1:fp:LIB:aix3-r2::2500:5
 TMF_3.1_SP1:fp:LIB:aix4-r1::2534:6
 TMF_3.1_SP1:fp:BIN:aix3-r2::3347:12
 TMF_3.1_SP1:fp:BIN:aix4-r1::3423:13
 TMF_3.1_SP1:fp:ALIDB:aix3-r2::298:19
 TMF_3.1_SP1:fp:ALIDB:aix4-r1::298:20
 TMF_3.1_SP1:fp:DB:aix3-r2::4:26
 TMF_3.1_SP1:fp:DB:aix4-r1::4:27
(Format of IND file is prodname:fp:Target DIR:Interpreter::Size:Packet File Number)

#more ./cfg/file1.cfg
 #*TFP-v2.01
 #version=TMF_3.1_SP1_LIB_solaris2
 do_compress=y
 do_checksum=y
 stop_on_error=n
 create_dirs=y
 keep_paths=y
 file_cksums=y
 unix_default_dir_uid=0
 unix_default_file_uid=0
 unix_default_dir_gid=0
 unix_default_file_gid=0
 post_notice=n
 default_file_mode=0755
 default_dir_mode=0755
 unix_before_prog_from_src=y                              ! Yes use on UNIX
 unix_before_prog_path=/a/tivoli/builds/3/TMP_3.1/prod \  ! path of script
  /solaris2/patches/scripts/TMF_31_SP1/LIB_before_unix.sh
 nt_before_prog_from_src=y                                ! Yes use on NT
 nt_before_prog_path=/a/tivoli/builds/3/TMP_3.1/prod/  \  ! path of script
  /solaris2/patches/scripts/TMF_31_SP1/LIB_before_nt.sh
 #
 # File List:
 #
 %
 .placeholder
 libas.so
 libas_imp.so
 librim.so
 libtable.so
 libtas.so
 %

#more patches.lst
TMF_31_S:TME 10 Framework Version 3.1 Service Pack 1
ADE_31_S:TME 10 ADE Version 3.1 Service Pack 1
```

3.9.1 CD-ROM Installation Tools

The sapack command is used to create and expand file packages. During a client installation, sapack is transferred to the client and used to extract the binaries and other files until the oserv is started (after which unpack is performed through the fps_unpack method).

- Prior to Release 2.0.2, problems occurred if the install bundle was deleted (extracted with wpreinst.sh) and clients could not be installed. To solve

this problem, the install bundle is now placed in the binary tree and should not be removed.

- The sapack command is not documented, and direct use is therefore not supported by Tivoli. If you want to use it, it does have helpful usage statement, just run sapack with no arguments to see it.
- To extract the before or after scripts using sapack, perform the following:

  ```
  o_dispatch=100 <path_to_sapack> -Dinstall_progs=y -u <path_to_pkt>
  ```

  We set o_dispatch to 100 to ensure we do not execute the script on a currently-running TMR. The afterscript always runs; so, we must prevent it from modifying the current TMR.
- For architecture types that are released on separate media, run wpreinst.sh and extract their sapack and copy it into the install bundle directory.

```
#cd /usr/local/Tivoli/bin/
#./client_bundle/bin/aix4-r1/sapack
Usage:  ./client_bundle/bin/aix4-r1/sapack [-C dir ] [-Dkey=value] [-pPl] [-N name] fp-desc > fpblk
        ./client_bundle/bin/aix4-r1/sapack -u [-C dir ] [-Dkey=value] [-pPl] [-N name] fpblk
        ./client_bundle/bin/aix4-r1/sapack -ur [-C dir ] [-Dkey=value] [-N name] fpblk
```

The sapack command expands files into the current directory. The recommended steps for usage (on a test system only) are:

1. Create a new directory
2. Copy the PKT file to be unpacked into the directory
3. Copy the $BINDIR/TAS/Install/sapack utility
4. Export o_dispatch to a port value that is not in use
5. Run sapack -up PKTname to preview the unpack
6. Run sapack - u PKTname (or sapack -Dinstall_progrs=y -u PKTname if you wish to save the script that is run)

Chapter 4. Tivoli Software Installation Service

The Tivoli Software Installation Service (SIS) is an application designed for faster and easier installation of Tivoli products in a Tivoli Management Region (TMR). SIS can push products to Tivoli clients and is intended to provide increased functionality over the standard Tivoli installation process used in previous releases of the Tivoli Framework. Using SIS, you can create an install repository (IR) that contains the installation images of the products, determine a product configuration for some or all of the machines in your Tivoli Management Region, and install that configuration on the machines you choose.

4.1 SIS Component Overview

This session gives a high level overview of the concepts of SIS.

There are three components to SIS. They are:

- The Tivoli Software Installation Service Binaries:

  The Tivoli SIS server is any managed node (including the TMR server) that has the SIS binaries installed on it. Using either the graphical user interface (GUI) or the command line interface, you can invoke the Tivoli Software Installation Service that runs on the SIS server.

- The Install Repository:

  SIS introduces the concept of an Install Repository (IR). The IR holds the images of all the products or patches that are to be installed using SIS. Products are imported into the IR either through the GUI or through the command line interface(CLI). You control which products are to be installed on which targets. During the installation of SIS, you specify the location of the IR. You may find it useful to set a variable, `$IR`. The Install Repository location will be referenced in scripts and commands in this chapter by the variable `$IR`. Note that this variable is not defined or used by SIS itself.

- A Response File:

  Response files are text files that contain product and machine attributes that are required by SIS for an installation, such as:

  - Product install directory paths
  - Machine name
  - Machine access methods
  - Operating system type
  - Password settings

- Login account information

You can define specific attributes for a product and machine, or use a global set of attributes for all machines, and pass these values to SIS in a response file.

Figure 15 shows the high-level design of SIS:

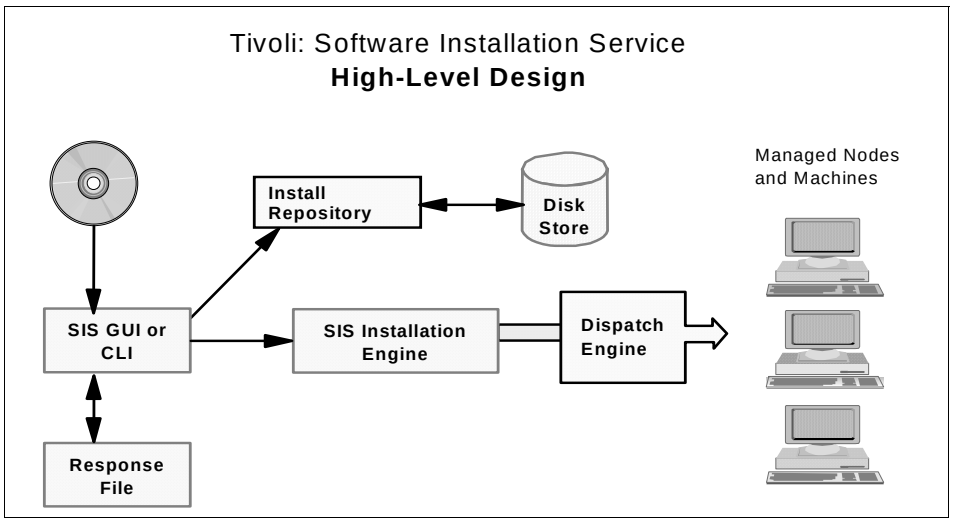


Figure 15. SIS High-level Design

The steps for implementing SIS are:

1. Build the IR, and import the product images you need into the Install Repository by SIS GUI or using the CLI.
2. Once the IR has been built, you can configure what products you want to install on which nodes. This is done through the SIS GUI or through a Response File.
3. Invoke the SIS Installation Engine to dispatch and install products to the machines you specified by either GUI or CLI.

4.1.1 SIS Considerations

You should note the following about SIS:

- SIS does not support installing to nodes located in interconnected TMRs. However, to conserve disk space, the Install Repository can be shared between different TMRs.

- You can not use SIS to install any products or patches on an Endpoint. However, you can create an Endpoint itself using SIS on Windows NT or UNIX. This is discussed in the next paragraph.
- SIS can not create Endpoints on any other machines without the PC agent running on them. This means if you want to create a Windows 95, Windows 98, Windows 3.x, OS/2, or NetWare Endpoint using SIS; it must already be a PC managed node.

Note

For Windows NT systems:

1. You must run the bash command shell for the SIS command line commands contained in this document.
2. The correct directory for all SIS command line programs is:

   ```
   $BINDIR/../generic_unix/SIS
   ```

4.2 Installation of SIS

The SIS server 3.6 can run on AIX, HP-UX, Solaris, and Windows NT platforms, and is installed from either the Tivoli desktop or command line using the traditional framework install methods. Refer to the user manual and release notes for details of OS versions and levels supported.

4.2.1 Installation Procedure

Reminder

Do not forget to perform a Tivoli backup before you install any product on your TMR.

Use the following guidelines to help you plan your SIS installation:

- SIS requires sufficient disk capacity to hold the images of the software SIS will be pushing and installing. To help reduce disk space requirements, SIS 3.6 introduces the concept of a shared IR. A shared IR uses either UNIX NFS mounts or Windows NT sharing to share a single IR among several SIS servers.
- Software prerequisites (Operating System fixes and Tivoli Patches. Note, always check the release notes).
- Network topology. Installation from a SIS server will involve a great deal of network traffic.

- The Tivoli Management environment. The products in use, the distribution of managed nodes and other systems, and so on.

In order to install and use SIS, you need to have the following installed:

- The TMR server.
- The managed node on which you want to install SIS (this could be the TMR server if desired).
- For Windows NT managed nodes, `bash.exe`, version 11 or later (included in Tivoli Management Framework 3.6, 3.2 and 3.2.1 and on the SIS CD ROM).

Note

SIS can be installed on any managed node including the TMR server. However, for performance reasons, it is recommended that you install the SIS software on a managed node that is not the TMR server.

Note also that SIS requires a Java Virtual Machine to run, which can limit the versions of platforms on which it can run.

For more information about Planning activities, please refer to *Mass Installation Using SIS* - SG24-5109.

After preparing for the installation, you can start the Tivoli desktop and use the traditional installation method to install SIS on the machine you choose as the SIS server. You can also use the `winstall` command to install the SIS server. The install for SIS has an install options dialog as shown in Figure 16.

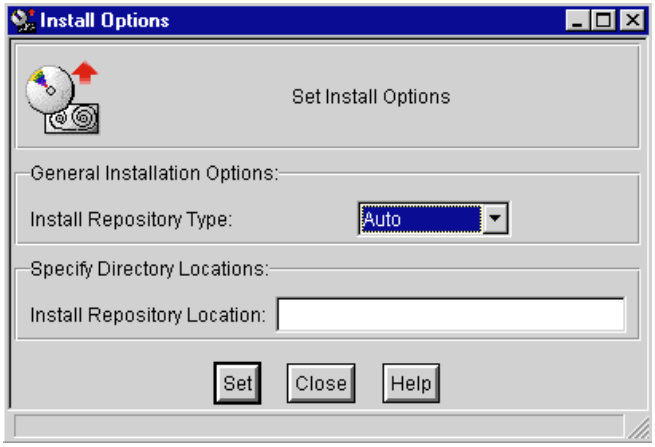


Figure 16. SIS Install Options Dialog

In the install options dialog, you need to perform the following actions:

1. Specify the type of IR you will be using. There are four types of IR you can have:

   Non-Shared This option does not allow you to share this repository with SIS running in another TMR.

   Read-Only Read-Only allows SIS to read the products from this repository but will not allow users to add or remove products from the repository.

   Shared This option allows the repository to be shared by SIS running in other TMRs for both reading as well as importing and removing Tivoli products for this repository.

   Auto Auto will set the mode of the IR for the user. If the IR specified in the installation options appears to have an existing Install Repository (that is $IR/TMR/Defaults/interp.sis-36 exists), then the IR mode is set to **Shared**. If this file does not exist, the IR mode is set to **Non-Shared**.

   You should set this value and not rely on SIS to provide a default.

2. Specify the path where your IR is to be located in the Install Repository Location field. If necessary, the directory specified will be created during the installation process. Remember to allow for sufficient disk space for product images that will be imported.

3. After finishing the installation, you must completely restart the Tivoli desktop. This means you must shut down all Tivoli desktops started in that TMR. Now when you open the Desktop action bar pull down and select **Install** from the pull down list, the Software Installation Services... option is now available

   For more information about installing SIS, please refer to the *Tivoli Software Installation Service User's Guide.*

4.3 Using SIS

This section provides details on the components you will work with to use the Software Installation Service.

4.3.1 Starting the SIS Graphical User Interface

You can start SIS from the Tivoli desktop or the command line. To start from the desktop perform the following steps:

1. From the Tivoli desktop menu select Desktop --> Install --> Software Installation Service... to launch the SIS desktop.
2. When you are prompted for the Tivoli Installation password by the Get Installation password dialog, enter the framework installation password, if you had specified one, and select **OK**. If you do not have an installation password, select **OK**. At this point, the SIS desktop, as shown in Figure 17, should be displayed.

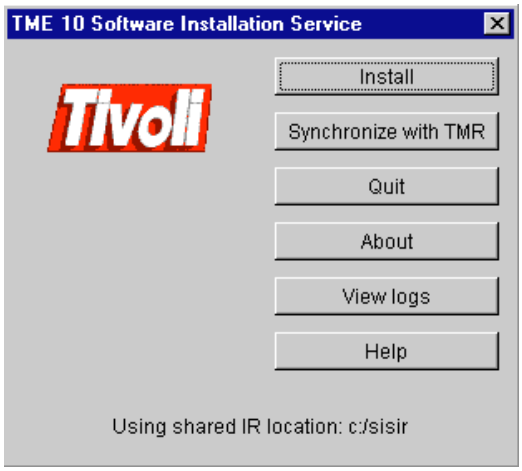


Figure 17. SIS Desktop Dialog

The desktop start method is the supported method for starting the SIS GUI. If it is necessary to start SIS from the command line (for example, if directed to do so by Tivoli support), you can enter, in a UNIX environment:

```
$BINDIR/../generic_unix/SIS/sisgui
```

In Windows NT environment, enter:

```
sh %BINDIR%/../generic_unix/SIS/sisgui
```

From the SIS dialog, you can choose:

Install To startup the software installation procedure.

Synchronize with TMR

To synchronize the IR with the TMR server database and update product and managed node installation information in the IR.

Quit To close Tivoli Installation Service and remove TMR, IR, and usage locks.

About To view SIS product information.

View Logs To view the HTML log files generated by SIS.

The following occurs when SIS starts:

1. SIS writes to the `$IR/sis-<node>.out` file for both shared and non-shared IRs.
2. SIS creates the necessary locks. There are four locks created during SIS startup:

   TMR lock When the SIS starts, it will create the TMR lock in the object database using `wregister -i SIS <SIS nodename>`. The TMR lock prevents other users within the TMR from using SIS at the same time. That is, this prevents two users from attempting to distribute to the same machine at the same time. Instead, the second user would see a warning message describing the TMR lock and displaying the host name where SIS is being executed within the TMR.

   IR lock The IR lock prevents multiple users in different TMRs from using the IR directory in write mode concurrently. A warning message will be displayed to all users after the initial launch of SIS. This warning lists the hostname and region name of the machine running the initial SIS. If the IR is used in shared mode, the user has the option of continuing in read-only mode. If the IR is used in read-only mode, the IR lock is not created. The IR lock is created as a file `ir.lck` when SIS starts.

Note

The Install Repository can be shared between different versions of SIS, and in order to maintain compatibility with SIS 1.0, SIS 3.6 will create a lock file in two directories:

```
$IR/ir.lck
$IR/TMR/Defaults/ir.lck
```

`$IR/ir.lck` is used by SIS 1.0, `$IR/TMR/Defaults/ir.lck` is for SIS 3.6.

Usage lock The IR can be shared between different TMRs at the same time. The first node that starts SIS has the write authority. Other nodes in different TMRs can still use the IR and will

create the usage lock when they start SIS. The usage lock file is $IR/TMR/Defaults/ULOCK/<nodename>.lck.

CLOSEDIR.lck Used by `wimport -remove` in the ULOCK directory.

3. SIS reads the products in the IR.

   SIS first checks for the existence of the $IR/TMR/Defaults/miniprod.sav (non-shared IR) or $IR/TMR/<region number/miniprod.sav (shared IR) file. If this file exists, SIS will read the product information. The miniprod.sav file is the product index file that indicates where to find the product detail information. If SIS can not find the miniprod.sav, it will scan the $IR/Framework, $IR/Products, $IR/Patches, and $IR/Upgrades directories to build miniprod.sav.

   There is a product.sav in separate directories for each product. The directory depends on the type of product, Framework, Patches, Products, or Upgrades. The directory structure for both the shared and non-shared IR is: $IR/<product type>/<product name>. SIS reads the product.sav for detailed product information. If the product.sav does not exist, SIS can create it according to the .IND file in each product directory.

   Note

   If you find the products information is not correct, just delete miniprod.sav, product.sav and restart SIS.

4. Synchronize with the TMR.

   SIS checks for the existence of $IR/TMR/Defaults/minitmr.sav (non-shared IR) or $IR/TMR/<region number>/minitmr file. If this file exists, SIS will read the pointer information about the managed nodes within a TMR and the Tivoli products currently installed in the TMR from this file. The actual information is stored in the tmr.sav file. If the minitmr.sav doesn't exist, SIS looks for the TMRSync.out and if it does not find it then SIS runs the `$BINDIR/../generic_unix/TMRSync.sh` script to build the TMRSynch.out to synchronize the IR with the TMR. This script queries the TMR for the information.

   Note

   If you need to re-synchronize with TMR server when SIS starts, just delete the minitmr.sav, then start SIS

   SIS 1.0 checks $IR/tmr.sav when it starts

   For each TMR using the Install Repository, there is a tmr.sav in the $IR/TMR/Defaults (non-shared IR) or $IR/TMR/<region number> (shared

IR) directory. SIS reads the tmr.sav for detailed TMR information. If the `tmr.sav` does not exist, SIS will create it when Software Installation Service is launched.

If there are some problems during SIS startup, refer to 4.4.3, “Troubleshooting SIS Startup” on page 107 for more information.

4.3.2 Building the Install Repository

The IR contains all products and patches available for installation on to machines by SIS. In the IR directory you specified during SIS installation are the following directories:

- Framework/
- Products/
- Patches/
- Upgrades/

These directories contain the .IND files and file packages for Tivoli products. Each version of Framework, Products, Patches, or Upgrades is stored in a separate directory. There is also a product.sav file in each directory. This file is created according to the .IND file when you initially build the IR or when you add products/patches to the IR. The product.sav also stores product defaults. Refer to 3.9, “Installation CD-ROM Contents” on page 79 for more information about .IND files. An example of the top level IR directory is shown in Figure 18.

```
[root]ls -l
total 72
drwxrwsrwx   4 root     sys          512 Nov 15 09:54 Framework
drwxrwsrwx   2 root     sys          512 Nov 15 09:47 GIF
drwxrwsrwx   3 root     sys          512 Nov 15 09:48 Patches
drwxrwsrwx  32 root     sys         2560 Nov 19 11:22 Products
drwxrwsrwx   3 root     sys          512 Nov 15 09:47 TMR
drwxrwsrwx   3 root     sys          512 Nov 15 09:48 Upgrades
drwxrwsrwx  24 root     sys          512 Nov 15 09:47 bin
drwxr-sr-x  12 root     sys          512 Nov 19 16:41 log
drwxrwx---   2 root     system       512 Nov 15 08:57 lost+found

[root]ls -l
total 16
drwxrwsrwx   3 root     sys         1024 Nov 15 09:52 TME10_ENDPOINT
S_4.1
drwxrwsrwx   3 root     sys          512 Nov 15 09:56 TMF-3.6
```

Figure 18. Contents of IR Directory

When SIS is installed, the IR contains several prepackaged product image directories. Other directories contain only a .IND file. These prepackaged .IND files contain tags that provide SIS with additional information not provided in the product images of the product CD-ROMs. When the product that corresponds to the .IND file is imported, the prepackaged tags are merged with the imported product .IND file.

An example of what the provided product directories looks like is shown in Figure 19.

```
[root]ls -l |more
total 248
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 ACF-3.6-TME_10_Enterprise_
Console_3.6_Adapter_Configuration_Facility
drwxrwsrwx   3 root     sys          512 Nov 15 10:28 ARMEP_36-TME_10_Distribute
d_Monitoring_ARM_EndPoint_version_3.6
drwxrwsrwx   3 root     sys          512 Nov 15 10:28 ARMMON_36-TME_10_Distribut
ed_Monitoring_ARM_Monitors_version_3.6
drwxrwsrwx   3 root     sys          512 Nov 15 10:25 ARM_36-TME_10_Distributed_
Monitoring_ARM_Agent_version_3.6
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Admin-3.0-Tivoli_Admin_3.0
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Admin-3.1-TME_10_User_Admi
nistration_3.1_
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Admin-PCFPGEN_3.0-Tivoli_A
dmin_3.0_PC_Filepack_Utilities
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 AdminPcFp-3.1-TME_10_User_
Administration_3.1_PC_Filepack_Utilities
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Courier-3.0-Tivoli_Courier
_3.0
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Courier-3.1-TME_10_Softwar
e_Distribution_Release_3.1
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Courier-3.6-TME_10_Softwar
e_Distribution,_Version_3.6
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Inventory-3.6-TME_10_Inven
tory,_Version_3.6
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Inventory-Tivoli_Inventory
_Application
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 Inventory_3.1-TME_10_Inven
tory_Version_3.1,_for_NT_only
drwxrwsrwx   3 root     sys          512 Nov 15 10:26 NTMonitors35-3.6-TME_10_Di
stributed_Monitoring_NT_Monitors
drwxrwsrwx   2 root     sys          512 Nov 15 10:26 Sentry2.0.2-3.0-Tivoli_Sen
try_Version_3.0
drwxrwsrwx   3 root     sys         1024 Nov 15 10:39 Sentry2.0.2-3.6-TME_10_Dis
tributed_Monitoring_3.6
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 TACFPLUS-TACF_Tasks,_Monit
ors,_and_Event_Integration_v3.6
drwxrwsrwx   2 root     sys          512 Nov 15 09:48 TEC_CONSOLE-3.6-TME_10_Ent

[root]ls -l
total 16
-rwxrwxrwx   1 root     sys           27 Nov 15 09:48 CONTENTS.LST
-rwxrwxrwx   1 root     sys          595 Nov 15 09:48 COURIER.IND
```

Figure 19. Example of Provided Product Directories and Files

Tivoli products and patches must be imported into the IR before they can be installed on the machines in the TMR using SIS. For more information of how to import Tivoli products images into IR, please refer to *Mass Installation Using SIS* - SG24-5109 or the *Tivoli Software Installation Service User's Guide*.

Note

Before SIS copies the product images into the IR, it will check the disk space. If there is not enough space, it will display a warning window to tell you how much space is available and how much is needed for the products you want to import. You can either expand your IR space and continue or exit the import.

4.3.3 Select Target for Install

SIS allows you to install products on selected machines. For each selected machine, SIS will communicate with it to determine the operating system type.

For a new UNIX or NT machine, you will need to have either the root/Administrator password or use the trusted host option. NT is supported by rexec through TRIP only, no trusted host access. There is also one new option FTP/Account that is only for OS/400. For the machines created by the traditional Tivoli installation method, you can synchronize SIS with TMR server using the `TMRSync.sh` command.

From SIS, the Windows 3.x, 95 and 98, NetWare, and OS/2 platforms can only be used if they are PC managed nodes. You will not be able to use SIS to load the Tivoli Management Agent if the PC Agent is not already installed but rather should use the traditional Tivoli installation methods and then synchronize the TMR.

For new Windows NT managed nodes, if TRIP is not present, there will be a window to prompt you to remotely install TRIP. If you want to install TRIP remotely, there must be at least one Windows NT managed node that has already installed TRIP, and it must be in a domain with a trust relationship with the domain for the Windows NT machine you are trying to install. SIS will auto install TRIP using this managed node as a repeater to distribute TRIP to other Windows NT machines as you select. See also A.5.1, “Installation of the Tivoli Remote Installation Package” on page 616, and 3.2.2.1, “Tivoli Remote Execution Service” on page 62.

Note

When you add a new machine, SIS will check the network connection to it. You can only add a new machine if it is available. This can help reduce installation errors caused by network problems.

4.3.4 SIS Response Files

Response Files are text files that allow you to automate product installations. When using the CLI, you need to provide a response file to SIS to tell the install engine what product to install, to which machine, and the method to use for installation.

By using the GUI Install Details screen, you can define a target machine you want to create (or add) and what products to push to that target machine. After this, you can export the information to a response file. This file can then be edited and/or re-imported into the GUI for use or to be used in a command line installation with the command `wsis <response file>`.

You can see more Response File examples in *Mass Installation Using SIS* - SG24-5109.

4.3.4.1 Response File Considerations

The passwords for the root/Administrator of the target machine is in clear text format in the response file. You can edit it directly. If you don't want to put the password into the response file, set the keyword `promptForPassword=yes` in the response file. If this keyword and option are used, SIS will prompt you for the password during installation. If you put the password into the response files, you will need to take the proper precautions to secure these files.

When you use SIS to install products on a large quantity of machines, you can export the installation information of one machine and then edit it through your favorite text editor to add the information about other machines. At this point, you could either install using CLI or, after re-importing the file, through the GUI.

You can get the keywords about the install directories from the .IND file in the product packages. The .IND file specifies installation options including install directories, port specifications, database systems, and so on. For example, install directory information is located in the id line of the file. It is recommended that you use the GUI to export a response file for a sample machine and use this as a template to add new machines in your response

file. By using this method, it will be easier to build the response file, and it will be more accurate.

Figure 20 shows the ep.IND file from the $IR/Framework/TME10_FRAMEWORK_4.1 directory. The highlighted lines were used in the response file that is displayed in Figure 21 on page 96.

```
TME10_ENDPOINTS_4.1:description:TME 10 Endpoints (v4.1):TME10_ENDPOINTS_4.1
TME10_ENDPOINTS_4.1:lcf_install:
TME10_ENDPOINTS_4.1:patch_id:TME10_ENDPOINTS_4.1
TME10_ENDPOINTS_4.1:id:BIN2:binaries:both::default=@lcf_stage@:ThisDir=@BIN2@;ThisHo
st=@HostName@;ThisPkg=BIN2;BINDIR=@BINDIR@;INTERP=@Arch@;:
TME10_ENDPOINTS_4.1:fp:BIN2:solaris2::2589:1
TME10_ENDPOINTS_4.1:fp:BIN2:aix4-r1::1891:2
TME10_ENDPOINTS_4.1:fp:BIN2:hpux10::2006:3
TME10_ENDPOINTS_4.1:fp:BIN2:w32-ix86::1916:4
TME10_ENDPOINTS_4.1:fp:BIN2:sunos4::372:5
TME10_ENDPOINTS_4.1:fp:BIN2:windows::856:6
TME10_ENDPOINTS_4.1:fp:BIN2:win95::839:7
TME10_ENDPOINTS_4.1:fp:BIN2:nt::1000913:8
TME10_ENDPOINTS_4.1:fp:BIN2:nw3::3171:9
TME10_ENDPOINTS_4.1:fp:BIN2:nw4::3223:10
TME10_ENDPOINTS_4.1:fp:BIN2:os2::738:11
TME10_ENDPOINTS_4.1:fp:BIN2:netware::738:12
TME10_ENDPOINTS_4.1:id:@EndpointLabel@:Endpoint
Label:product::default=@HostName@::
TME10_ENDPOINTS_4.1:gui:L:EndpointLabel:product:
TME10_ENDPOINTS_4.1:id:@InstallBaseDirectory@:Path to install
endpoint:product::default=@lcf_default@::
TME10_ENDPOINTS_4.1:gui:L:InstallBaseDirectory:product:
TME10_ENDPOINTS_4.1:id:@GatewayIP@:Gateway IP address:product::::
TME10_ENDPOINTS_4.1:gui:L:GatewayIP:product:
TME10_ENDPOINTS_4.1:id:@GatewayPort@:Gateway Port:product::default=9494::
TME10_ENDPOINTS_4.1:gui:L:GatewayPort:product:
TME10_ENDPOINTS_4.1:id:@EndpointPort@:Optional port for
endpoint:product::default=9494::
TME10_ENDPOINTS_4.1:gui:L:EndpointPort:product:
TME10_ENDPOINTS_4.1:id:@EndpointStartupOpts@:Additional Options for
endpoint:product::::
TME10_ENDPOINTS_4.1:gui:L:EndpointStartupOpts:product:
TME10_ENDPOINTS_4.1:id:@EndpointStartupTimeout@:Time to wait for the endpoint
to connect:product::default=300::
TME10_ENDPOINTS_4.1:gui:L:EndpointStartupTimeout:product:
```

Figure 20. Example of the ID Line in a .IND File

Be careful when using both byProduct and byNode definition in one response file. SIS will attempt to parse the byNode section first, then the byProduct section. If there are any conflict in definitions, the byProduct selections will generally be overwritten by the byNode selections. Figure 21 on page 96 contains a response file with conflicting definitions.

```
[machine pctmp109]
access=rexec
userid=Administrator
password=password
interp=w32-ix86

[byProduct]
aliasname2=pctmp109

[byNode]
pctmp109=aliasname1

[alias aliasname1 TME10_ENDPOINTS_4.1-client-]
@EndpointLabel@=pctmp109
@InstallBaseDirectory@=c:/Tivoli/lcf
@EndpointPort@=9494
@EndpointStartupOpts@=
@EndpointStartupTimeout@=300
GatewayName=Broadcast to Gateways
PolicyRegionName=None
tapUser=
TapPassword=
EndpointEnableRestart=No
Overwrite=

[alias aliasname2 TME10_ENDPOINTS_4.1-client-]
@InstallBaseDirectory@=c:/Tivoli/lcf1

[globals]
InstallPassword=
AskBeforeReboot=false
InstallAlgorithm=Install to Machines Maximizing Network Bandwidth
```

Figure 21. Example of a SIS Response File with Conflicting Definitions

In the above example, the target machine pctmp109 may use an `InstallationBaseDirectory` of c:/Tivoli/lcf1 since this keyword conflicted with the definitions in the byNode section. This type of conflict can cause unpredictable results and should be avoided.

After the response file is created, you can then use the `wsis <response file>` command to install to target machines, or you can import the response file in the GUI and install to target machines. The resulting log file for SIS importing response file is shown in Figure 22 on page 97. You can see that SIS first parsed the byNode section then the byProduct section. The result of the installation is determined by the byProduct section.

```
About to parse the "by node" section

Parsing node section.
  --  Seaching InstallInfo for (pctmp109, TME10_ENDPOINTS_4.1-client-)
InstallInfo found.
Beginning import of response file attributes into InstallInfo object.
@EndpointLabel@ == pctmp109
@InstallBaseDirectory@ == c:/Tivoli/lcf
@EndpointPort@ == 9494
@EndpointStartupOpts@ ==
@EndpointStartupTimeout@ == 300
GatewayName == Broadcast to Gateways
PolicyRegionName == None
tapUser ==
TapPassword ==
EndpointEnableRestart == No
Attribute import successful.
bout to parse the "by product" section

Parsing product section.
  --  Seaching InstallInfo for (pctmp109, TME10_ENDPOINTS_4.1-client-)
InstallInfo found.
Beginning import of response file attributes into InstallInfo object.
@InstallBaseDirectory@ == c:/Tivoli/lcf1
Attribute import successful.


About to start the massive UD check.....
massive UD check done!!!
```

Figure 22. Example of SIS Log File for Imported Response File

4.3.5 Using SIS to Install Tivoli Products

You can use SIS to install the Tivoli Framework, products, and patches.

Note

SIS can be used to create TMA endpoints, but at the time of writing, you can not use SIS to install products that require installation to the endpoints. For some Tivoli products, such as Tivoli Security Management and Tivoli Remote Control, you need to install some extra code on the endpoints. If you want to use these products, you will still need to use the Tivoli traditional installation method or Tivoli Software Distribution.

The following sections detail the flow when SIS installs new endpoints or managed nodes and products.

4.3.5.1 Run Prerequisite Checks

There are four types of prerequisites: Client, Product, Patch, and Endpoint. The type of prerequisite that is run depends on what is being installed.

Client Prerequisites performs the following tasks:

- Connection test - Tests ability for SIS Server to talk with the client. It attempts connectivity by running this prerequisite on the remote client. If the connectivity attempt times out, then this prerequisite fails.
- Bash prerequisite - Windows NT only. Verifies that the `bash.exe` Version 11 or later is installed on the Windows NT machines.
- Shell prerequisite - UNIX only. Verifies that root shell is either Bourne or Korn shell.
- /dev/null permission - UNIX only. Verifies that /dev/null has world write access.
- NT kbdus.dll - Windows NT only. Verifies that the Windows NT node has the US keyboard DLL installed in the system directory. This DLL is necessary for SIS to create Windows NT managed nodes.
- NTFS - Windows NT Only. Verifies that the client database directory is a local NTFS directory.
- Existing DBDIR - Checks for the existence of a managed node database in the $DBDIR. If the $DBDIR on that node contains an existing managed node database, a new managed Node will not be created unless the overwrite flag is checked for the database directory.
- Two way communication - Checks to make sure the client can ping the TMR server. It does this by running the `ping` command on the client.

Endpoint Prerequisites (except AS/400) performs the following tasks:

- Connection test.
- Bash prerequisite - Windows NT Only.
- Shell prerequisite - UNIX only.
- /dev/null permission - UNIX only.
- NT kdbus.dll - Windows NT only.
- Two way communication with TMR server.

The actions taken by these prerequisite checks are the same as the Client Prerequisites. Endpoint prerequisites also adds the following check:

- Two way communication with gateway check - Checks to make sure the client can ping the gateway it will use to login. It does this by running the `ping` command on the client.

AS/400 Endpoint Prerequisites performs the following tasks:

- Disk Space - Verifies that a minimum of 20MB of free disk space is available.
- Authority - Verifies that the AS/400 account specified in either the ADD MACHINE dialog of the user ID in the response file has the authority to issue the `RSTLICPGM` command and has *SAVSYS special authority.
- Allow object restore - the QALWOBJRST system value must specify *ALL or *ALWPGMADP. This value specifies whether or not objects with security sensitive attributes can be restored.
- Slip Installation - Determines if the 1TMELCF product is already installed.
- Connection with the gateway - Verifies that the endpoint can open a TCP/IP socket to the gateway and port specified.

> **Note**
>
> For SIS to be able to perform the prerequisite checks, the AS/400 account (specified in the ADD MACHINE dialog for the GUI or userid in the response file) must have authority to issue the `RSTOBJ` command.

Product and Product Prerequisites

- Connection to target - Verifies that the machine from which you are installing and the target machine can successfully communicate with each other.

> **Note**
>
> You can add user defined prerequisites as you need through the SIS GUI.

4.3.5.2 Disk Space Probe

The disk space probe is run to ensure that there is enough disk space for the file package.

When checking disk space, SIS first determines the disk space of each directory SIS will use and compares this to the file pack size. The file size is the next to last field of the `fp` line shown in the .IND file for the product/patch. From this file, we use only the disk space for any file package marked generic and the interpreter type of the target machine. As SIS determines size requirements, the size of the file package is subtracted from the free space available at each target directory. This is how SIS determines if enough disk space is available. If using NFS mounts or Windows NT shares to share man pages, and so on, SIS will track all file packs going to the same physical disk.

The amount of necessary disk space will be calculated based on the size of the file pack not the file pack multiplied by the number of machines sharing that physical drive.

For example, to install an endpoint for a AIX, locate the line containing the correct interpreter type `aix4-r1` in Figure 23. The correct line is `TME10_ENDPOINTS_4.1:fp:BIN2:aix4-r1::1891:2.` The next to last field for this entry is `1891`. This number is in KB. So, the amount of disk space necessary for an AIX endpoint is roughly 1.84MB.

```
TME10_ENDPOINTS_4.1:fp:BIN2:solaris2::2589:1
TME10_ENDPOINTS_4.1:fp:BIN2:aix4-r1::1891:2
TME10_ENDPOINTS_4.1:fp:BIN2:hpux10::2006:3
TME10_ENDPOINTS_4.1:fp:BIN2:w32-ix86::1916:4
TME10_ENDPOINTS_4.1:fp:BIN2:sunos4::372:5
TME10_ENDPOINTS_4.1:fp:BIN2:windows::856:6
TME10_ENDPOINTS_4.1:fp:BIN2:win95::839:7
TME10_ENDPOINTS_4.1:fp:BIN2:nt::1913:8
TME10_ENDPOINTS_4.1:fp:BIN2:nw3::3171:9
TME10_ENDPOINTS_4.1:fp:BIN2:nw4::3223:10
TME10_ENDPOINTS_4.1:fp:BIN2:os2::738:11
TME10_ENDPOINTS_4.1:fp:BIN2:netware::738:12
```

Figure 23. Example of "fp" Line in .IND File

Another way to obtain the amount of disk space is to run an `image_report`. You will need to have the path to the directory that contains the .IND file, the name of the .IND file, and either the interpreter type or you can product a report with all interpreter types.

The syntax of `image_report` is shown in Figure 24:

```
Usage: image_report [-a regex] [-c cdrom-dir] [-f] [product [product-2] ...]
    -a   only report on media packets architectures that match the given regex.
         for example, to select the sunos4 specific packets, pass:
                -a '(sunos4)|(generic)'
    -c   cdrom image directory [defaults to the current working directory]
    -f   give full listing of the contents of each media packet

    the product list is a list of the install index names and does not need
    the .IND extention.  If no products are given on the command line, all
    install indexes found in the cdrom image directory are listed.
```

Figure 24. Usage of the image_report Command

Figure 25 on page 101 shows one example of the `image_report` command.

```
$BINDIR/TAS/INSTALL/image_report -c
/var/spool/Tivoli/IR/Frameork/TME10_ENDPOINTS_4.1 ep

                          Installation Media Report
                           TME 10 Endpoints (v4.1)
    /var/spool/Tivoli/IR/Framework/TME10_ENDPOINTS_4.1
    TME10_ENDPOINTS_4.1 Pa  Platform    Size   Compress Ratio    Date Built
     File (Install check file)          (MB)    Size
    ================================================================================
    binaries                solaris2      2.59    0.22   11.85   11/17 17:02
    FILE1.PKT ()
    binaries                nw4           3.22    0.55    5.83   11/17 17:02
    FILE10.PKT ()
    binaries                os2           0.74    0.24    3.12   11/17 17:02
    FILE11.PKT ()
    binaries                netware       0.74    0.00  738000   11/17 17:02
    FILE12.PKT ()
    binaries                aix4-r1       1.89    0.32    5.87   11/17 17:02
    FILE2.PKT ()
    binaries                hpux10        2.01    0.29    6.90   11/17 17:02
    FILE3.PKT ()
    binaries                w32-ix86      1.92    0.57    3.34   11/17 17:02
    FILE4.PKT ()
    binaries                sunos4        0.37    0.22    1.73   11/17 17:02
    FILE5.PKT ()
    binaries                windows       0.86    0.49    1.74   11/17 17:02
    FILE6.PKT ()
    binaries                win95         0.84    0.52    1.62   11/17 17:02
    FILE7.PKT ()
    binaries                nt            1.91    0.57    3.35   11/17 17:02
    FILE8.PKT ()
    binaries                nw3           3.17    0.54    5.86   11/17 17:02
    FILE9.PKT ()
    --------------------------------------------------------------------------------
    Product total                        20.25    4.53    4.5
```

Figure 25. Sample image_report - ep.IND

The sample of the `image_report` in Figure 25 shows all operating system/interpreter sizes for the ep.IND file. This method can be used if you are unsure of the correct interpreter. However, in long listings, it may be more preferable to specify the interpreter type. In the example in Figure 26 on page 102, the `image_report` was specifically requested for generic and `aix4-r1`. The generic interpreter type files will always be delivered regardless of the operating system type. This is why you cannot deselect this operating system type when importing through the GUI.

```
$BINDIR/TAS/INSTALL/image_report -c /var/spool/Tivoli/IR/Framework/TME_ENDPOINTS_4.1
-a '(aix4-r1)|(generic)' ep

                          Installation Media Report
                           TME 10 Endpoints (v4.1)
    /var/spool/Tivoli/IR/Framework/TME10_ENDPOINTS_4.1
    TME10_ENDPOINTS_4.1 Pa  Platform    Size   Compress Ratio    Date Built
     File (Install check file)          (MB)    Size
    ================================================================================
    binaries               aix4-r1       1.89    0.32    5.87   11/17 17:02
    FILE2.PKT ()
    --------------------------------------------------------------------------------
    Product total                        1.89    0.32    5.9
```

Figure 26. Sample image_report - Specific Interpreter Types

4.3.5.3 File Package pushed

This is when the products are actually distributed to the node. All the file packages and after scripts are performed in this portion.

Note

The two main differences between SIS and the traditional installation method are:

- SIS adds Prerequisite and disk space checks.
- SIS can install products in parallel.

4.3.5.4 Managed Node and Endpoint Install Considerations

Related to the previous discussion are the following notes about using SIS to create managed nodes and TMA endpoints.

SIS install of a managed node

The first NT managed node in the TMR must have TRIP installed. SIS can not automatically install TRIP to the first NT managed node.

There are three installation options in SIS:

- Run Prerequisites & Probe Only - This will run the prerequisites and disk space checks, no installation occurs.
- Install to Machines independently - This means that installation will not wait to optimize a file package push. This will maximize network traffic since this algorithm does not attempt to combine MDist pushes.
- Install to Machines Maximizing Network Bandwidth - This will take full advantage of MDist repeaters.

> **Note**
>
> When you export the Response File, you will need to select one of these install options. If you choose Install, it will not check the disk space again. So, please make sure there is enough disk space for products.

It's recommended that you run a prerequisite and probe check before the installation to make sure that all prerequisites are met.

SIS install of a TMA endpoint

When using SIS to create a TMA endpoint, you can specify the gateway where this endpoint will log in. During installation, it will attempt to log into the gateway you defined. If the gateway is not available, you will get an installation failure message.

In this instance, even though you got an error message, the TMA endpoint installation is already finished. The next time you restart the endpoint, and if the gateway you defined is available, it will then log into the gateway successfully.

If you Synchronize with the TMR, all machines that are TMA endpoints will be deleted from the IR. You have to check your endpoint information from the Tivoli Desktop, gateway Manager.

4.3.6 Tuning SIS

SIS can be used to install Tivoli products or patches for hundreds of machines at the same time by both GUI or CLI. But in a production environment, be careful of the number of machines that you want to install at any one time for the following reasons:

- The number of machines that can have files installed on them at the same time is defined in the `$DBDIR/sis.ini`, as DISPATCHTHREADS=20. 20 is the default number. It's recommended that you do not change this value unless the application is encountering resource problems. If SIS is consuming too much system resources, adjust the number of DISPATCHTHREADS downward. For more information about `sis.ini`, please refer to the *Tivoli Software Installation Service User's Guide*.
- SIS will open a thread for each machine that you install for parallel installation; so, it will need a lot of system resource if you want to install products on many machines at the same time. It may be necessary to modify the system-specific process configuration.

- SIS will keep detailed log files for everything it does. So you will get many log files if you install products on many machines. Performing installations in smaller batches helps to manage the task of reviewing log files in the case of a problem.

4.3.6.1 Unsuccessful Install

If something happens during an installation, such as the network going down, SIS will not register the products unless the installation completed before the connection failed. If the installation is broken during the transfer of files, such as BIN or LIB files, you can just reinstall and select to overwrite the files. If it is broken at the update to Tivoli database files, it is likely that you will need to restore Tivoli backup files to get back the environment before your installation started and then reinstall.

4.3.7 Synchronize SIS with TMR

SIS is separate from the Tivoli Framework; it needs to synchronize with the TMR to make sure that all the information in the IR is up to date. If you are using the Tivoli traditional installation method to install some products after installing SIS, synchronize with the TMR to update your IR. You will also need to synchronize with the TMR if you delete a managed node.

There are two options for synchronizing:

- Auto Synchronize with TMR

  SIS will auto-synchronize with the TMR at startup if `$IR/TMR/Defaults/miniprod.sav` or `minitmr.sav` does not exist. If `miniprod.sav` does not exist, SIS will first rebuild the products information, then synchronize with the TMR, even if the `minitmr.sav` file exists. If only `minitmr.sav` does not exist, SIS will first load the products information according `miniprod.sav`, then will synchronize with the TMR to get node information.

- Manual Synchronize with TMR

  You can select the **Synch with TMR** button in SIS desktop, as shown in “SIS Desktop Dialog” on page 88, whenever you want to synchronize with the TMR.

  From the command line you can run `$BINDIR/../generic_unix/SIS/TMRSync.sh` to synchronize the IR and the TMR.

4.3.7.1 What Happens during Synchronize with TMR

SIS will run `TMRSync.sh` to synchronize with the TMR, and will execute `wlsinst -av` for information about what products are installed on what nodes, `wlookup` for ProductInfo, PatchInfo, and the Installation objects.

`TMRSync.sh` will put the information in a temporary file $TMPDIR/TMRSync.out, and SIS will parse this file and build the `minitmr.sav` and `tmr.sav`.

SIS will then copy the TMRSync.out to IR/log/iu-<date> directory and remove the `tmp` file.

When SIS synchronizes with the TMR, all of the endpoints in your SIS machine list will be lost since SIS will not keep any endpoint information.

Note

If there are hundreds or thousands of machines in your TMR, synchronizing SIS with the TMR may take half an hour or more to complete. In order to not execute `TMRSync.sh` during SIS startup, it's recommended that you backup `miniprod.sav`, `minitmr.sav`, `tmr.sav` files and check them before you start SIS. SIS will not execute the `TMRSync.sh` to synchronize with the TMR if it can find these files when it starts.

It is better to use one install method in your TMR. If you use SIS for Tivoli installs, try to keep using SIS. Otherwise, if you change back to traditional install, and then want to use SIS again, you will have to synchronize SIS with the TMR.

4.4 Troubleshooting SIS

This section describes actions you can take if you encounter problems while using SIS.

4.4.1 SIS Log Files

SIS creates log files for each SIS session. The following table documents the log files available from SIS:

Source	File Name and Location
SIS product installation	'wtemp'/tivoli.cinstall
Uninstall of SIS	'wtemp'/wuninst.log

Source	File Name and Location
Synchronization of TMR *	$IR/TMRSync.out-<date>-<time>
`sisclean`	$DBDIR/tmp/sisclean.log
Importing products into the IR *	$IR/iu-<date>-<time>/<product name>.IND-<date2>-<time2>
Installing products on targets *	$IR/iu-<date>-<time>/<target name>-<date2>-<target2>.html
Stdout and stderr information from SIS	sis-<hostname>.out Hostname is the hostname from which SIS is executed.
Stdout and stderr information from the `wimport` command	wimport-<hostname>.out Hostname is the node from which the `wimport` command is issued.
Stdout and stderr information from the `wsis` command.	wsis-<hostname>.out Hostname is the node from which `wsis` command is issued.
Import response file information *	$IR/iu-<date>-<time>/rf-<date2>-<target2>.html
Initialization of SIS *	$IR/iu-<date>-<time>/tmrInitLog.html
Detailed results of pushing each file package per product per target machine *	$IR/iu-<date>-<time>/<target>/<product>/FILEXX.PKT.log

* When SIS uses the IR in non-shared mode, these log files reside in $IR/log. In shared mode, these files reside in $IR/TMR/<region number>/log.

The date and time stamp on the directory structure iu-<date>-<time> is the date and time the SIS session was started.

The date2 and time2 stamps are the date and time stamp of when the action occurred, that is, when the product was installed.

Most of the log formats are in HTML format. To view these logs, you can select the **View Log** button or use any Web browser.

4.4.2 Troubleshooting SIS Desktop Launches

If there are some errors starting the SIS desktop, check the following:

- If you can not get the Software Installation Service... menu in the Tivoli desktop after SIS install, make sure to exit all desktops and re-start the

Tivoli desktop. In some cases, it may be necessary to re-cycle the oserv for the menu item to appear to ensure the desktop cache is cleared.

- Make sure your X server grants permission for the SIS X client to display. `xhost + <hostname>` must be performed even if SIS X client is running local to the X server. Figure 27 is an example of an `sis-<hostname>.out` in this situation.

```
Xlib: connection to "hptmp9:0.0" refused by server

Xlib: Client is not authorized to connect to Server

SIGSEGV   11*  segmentation violation
location=7B03D530.
stackbase=7B03AE84, stackpointer=7B03D438
.
.
.
```

Figure 27. Example of sis-<hostname>.out - xhost Error

- Make sure the /etc/Tivoli directory on SIS server contains the correct Tivoli environment.
- If there still are problems when launching from a Tivoli desktop, start SIS directly from the command line using `sisgui`. If SIS still does not start, try re-booting the machine or re-install SIS.
- If the SIS starts correctly from the command line, but not from the Tivoli desktop, use `launch_sis`, `sisgui` and `sisguisub.sh` with the "`-x`" debug flag and re-direct stderr to a file. The debug output should help determine the cause of the problem.

4.4.3 Troubleshooting SIS Startup

If you have problems during the startup of SIS, check the following:

- Check the $IR/sis-<node>.out file and $IR/log/iu-<date>/sis-<date>.html files for error messages.
- If you are using a shared repository, make sure you can create a file in the IR directory. If SIS does not have write access to the IR directory, SIS will not start.
- If the IR load fails, delete the IR/TMR/miniprod.sav and restart SIS. This will rebuild the `miniprod.sav`. If the repository load continues to fail, the IR

may be corrupted. If this is the case, you should re-install SIS and create a new IR.

> **Note**
>
> For SIS 1.0, it is the $IR/product.sav

- If the TMR Synchronization fails or does not contain the correct information, delete the $IR/TMR/minitmr.sav files. This will cause the `minitmr.sav` to be rebuilt during when SIS starts. If the TMR Synchronization fails, run the `$BINDIR/../generic_unix/SIS/TMRSync.sh` script manually and check $TMP/TMRSync.out file.

> **Note**
>
> For SIS 1.0, it is IR/tmr.sav and IR/TMRSync.out

- As a general rule, if SIS does not start correctly, delete the $IR/TMR/miniprod.sav and $IR/TMR/minitmr.sav and re-start SIS.

4.4.4 Troubleshooting SIS Locks

As discussed in 4.3.1, “Starting the SIS Graphical User Interface” on page 87, locks are created during the SIS initialization. When SIS is closed, these locks are deleted. The following will detail information concerning troubleshooting SIS locks and lock messages.

If you are sharing an IR, you will see the warning message that SIS had an error loading repository products and is rebuilding, as in Figure 28. You will also see this screen if you deleted the `miniprod.sav` file. For shared IRs, you will not be destroying any existing information for any other TMRs or managed nodes using the shared IR.

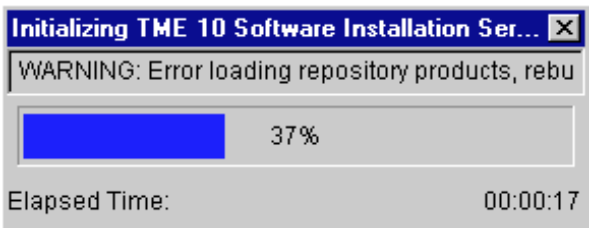


Figure 28. SIS Warning on First Initialization of a Shared IR

The warning in Figure 29 on page 109 is displayed if another TMR or managed node is already using the IR. Remember, the first TMR or managed

node that has the Shared IR type will establish the IR lock. From here, you can use the IR in ReadOnly mode, that is, you will not be able to import products, change global defaults of products, or change shared prerequisites.

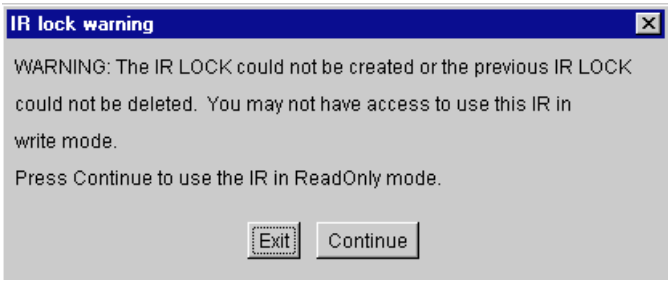


Figure 29. SIS IR Read-Only Mode Warning

If you do not have write access to the IR, you will receive the warning message in Figure 30. If this warning is in error, check the type of IR you are using and make sure you can create a file in the IR directory.

Figure 30. SIS Shared IR - No Write Access

When you are using a shared IR, an IR lock is created by the first TMR or managed node that accesses the IR with a Shared IR type. Subsequent TMRs or managed nodes that access that IR in Shared IR type will see the warning message as in Figure 31 on page 110. If you are sure that the node is not running SIS, you can select the **Delete Lock** button and continue. If not, you can select the **Exit** button. At this point, you can either wait for the node to complete the SIS installation, or you may place the IR type into ReadOnly.

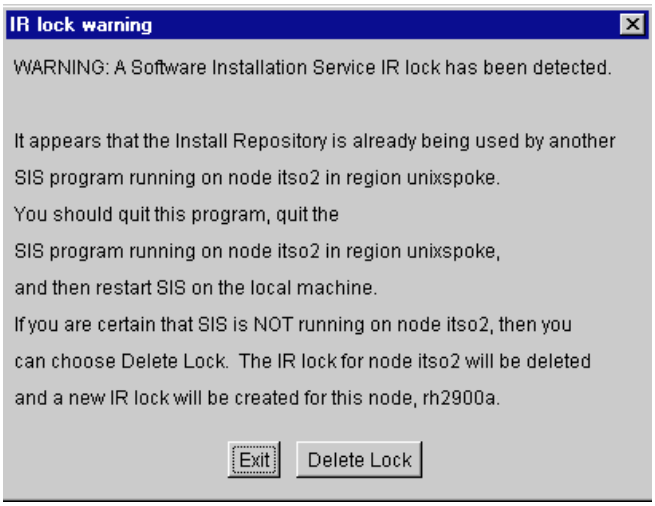


Figure 31. SIS Shared IR Type Warning

If there is a problem with the IR lock, you can delete the lock file `ir.lck` from the $IR and $IR/TMR/Defaults directories. If there is a problem with the usage lock, you can delete the usage lock ULOCK from the $IR/TMR/Defaults directory. And if there is a problem with the TMR lock, you will need to remove it with the `wregister -u SIS <nodename>` command.

4.4.5 Troubleshooting SIS Usage

SIS keeps the detailed information in the log files. These log files are extremely useful when trying to troubleshoot SIS.

If there are problems when installing products, check the disk space of your file systems. SIS has an installation progress window that will show the error message received if there is not enough disk space available. The amount of disk space necessary is located in the appropriate file packet for the product. See 4.3.5.2, “Disk Space Probe” on page 99 for how to determine the amount of necessary disk space.

When you are using SIS, especially on a Windows NT machine, you may run into an Mdist problem. This error will occur if there is not enough swap space. Extend the swap space and try again. Below is one example of Mdist error:

```
STDOUT:-----
{ 1 { 2010427741.3.7 "wgres1" } } -1
---------------------------
STDERR:-----
```

```
IdlSystemExec:idlParse() Token does not start with a single quote: {
Token number 2
----------------------------
FATAL:  Appending MDist error file ...
File Package:
"/IR/Products/RC_CTL_M-3.6-TME_10_Remote_Control_Controller_3.6__
Managed_Nodes_/FILE61.PKT"
Operation:     install  (m=5)
Finished:      Tue Nov 16 17:38:22 1998
```

When you delete one managed Node from your TMR server and synchronize your SIS node with TMR database, if you find the information about the node is still there, it is because the Tivoli object database on the TMR server is not correct. Execute the `wchkdb -u` command to clean up your TMR database.

4.4.6 Important SIS Files and Executables

Listed below is a summary of the some important SIS directories and files:

$DBDIR/ir.loc — This file specifies the location of the Install Repository. This is a text file and can be edited. You can edit this file *BEFORE THE FIRST INVOCATION* of SIS, and SIS will read this file and establish the location of the IR. However, if the location of the IR needs to be changed after the initial invocation of SIS, you will need to edit this file and then execute `$BINDIR/../generic_unix/SIS/PointIR`.

Note

If the `$DBDIR/ir.loc` file has been deleted, this file can be recovered with:

```
$BINDIR/../generic_unix/SIS/FindIR > $DBDIR/ir.loc
```

IR directory — All of the IR files are located in the directory you specified for IR during SIS Installation.

`$BINDIR/../generic_unix/SIS/` contains the following files:

- `BUILD.TXT` — This file contains the build version of SIS. It can be used to determine what version of SIS you are using.
- `FindIR` — This script is used to determine the location of the Install Repository.
- `PointIR` — This script is used to update the location of the Install Repository after the first invocation of SIS.
- `TMRSync.sh` — This script is used to synchronize SIS with TMR.

`launch_sis` This script is called by the desktop to launch SIS.

Chapter 5. Tivoli Object Database Backup

Tivoli keeps all its management data and resource definitions in a distributed database. The TMR server's database contains all the objects from the whole TMR. Each managed node keeps a subset related to itself. These databases must be backed up on a regular basis. The backup should be done not just at scheduled regular intervals but before and after a big change, such as a product installation.

You can backup the database for one or several clients and the TMR server or all of the TMR from either the command line or the desktop. Make this backup immediately or schedule a regular backup operation.

There are two methods for restoring a database. The first is the standard method to use if a system is otherwise operational (and the oserv is running). The second is referred to as a rescue operation and is used if the oserv cannot be started.

Note

Tivoli does not store data on a TMA endpoint. Therefore, the backup mechanism discussed in this chapter only applies to managed nodes.

5.1 The Tivoli Backup Process

Tivoli's main management database resides on the TMR server, and there are local databases on each managed node (PC managed nodes do not have databases). If these databases become lost or corrupted, Tivoli will lose track of resources being managed and the applications being used to manage them. Therefore, backups should be performed often while developing the Tivoli environment and regularly once Tivoli is deployed.

Backups are performed on the TMR server and client database files. Because the Tivoli Object Dispatcher (oserv) can be writing to the Tivoli database at any time, the built-in backup mechanism should be used instead of a simple `tar` or `cpio` file. This mechanism uses a snapshot process to capture the data while it is stable to prevent the loss of data.

It is recommended that you shut down the oserv (for example with `odadmin shutdown`) if you need to do a `tar` or `cpio` backup.

The Tivoli backup process provides an easy way to ensure all machines are located and backed up to a single location.

Important

Backups should be performed immediately before and immediately after a product installation or major maintenance procedure, such as the creation of numerous managed nodes. These backups should be kept on a separate tape and in a secure place for the life of the system. The backup could be the *only* way to rescue a management machine in some circumstances.

5.2 The Backup Process

Like the rest of Tivoli, the backup process is an object-oriented process. At managed node install time, a backup object is created in the Tivoli Object Database for each managed node. You can view a list of backup objects using the following command:

```
# wls -o /Library/BackupClient
```

Note that if a backup object was not created for a managed node install, this could indicate that the install did not complete correctly (see Section 5.6.3, “Database Cannot Be Backed Up” on page 125).

5.2.1 Before Starting Tivoli Backups

Besides scheduling, there are two ways to initiate the Tivoli management database backup:

- Tivoli Desktop
- Command line

It is advisable to periodically check the consistency of the database before performing a backup. Tivoli provide the `wchkdb` command for this purpose. The normal invocation would be with the `-u` flag. For more information on `wchkdb` see the on-line manual page or the *Tivoli Framework Reference Manual*.

Note

A wchkdb of a large TMR (say 200 managed nodes) can take as much as 30 hours depending on the complexity of the database. This makes it impractical to run a wchkdb at every backup or impossible if you make a daily backup. An alternative is to use the wchknode command that performs a simple check on a range of object dispatcher numbers you provide. The wchknode command is not as comprehensive as wchkdb. wchkdb, which will verify the integrity of the structure of all the objects in the database. The wchknode command simply checks that all objects respond to an objcall to get their label.

5.2.2 Backup Roles and Access Rights

There are a number of security features used in the backup process. You need to check the following:

- You must have the *backup* or *super* role in the TMR to create a backup and the *restore* or *super* role in the TMR to perform a restore.
- The Administrator also requires a valid user login name and a group name for the machine in which the backup file will be stored.

  To change the user login name and group name, open the Administrators window, right-click the **administrator** icon and select **Edit Properties...**. Be sure you have the desired IDs and then restart the Tivoli Desktop if a change was made. These changes must be done from the Tivoli Desktop because there is no command for this action.
- The ID used will need write permissions for the directory that will contain the backup file.

The following is a list of ways to ensure correct access:

- Create a new backup group of all administrators who will perform backups and assign ownership of the backup directory to that new group.
- Change the permissions on the backup directory to allow anyone to write to the directory. (This is not recommended as a permanent solution).
- Create a task for administrators with the backup role that runs as root and performs the wbkupdb command.
- Schedule the backups as the root administrator and let Tivoli perform the work.
- Change the user login name and group name to a valid one that has enough permissions to make the backup. The steps to change the user

login name and group name of a administrator are explained in “Backup Roles and Access Rights” on page 115.

> **Important**
>
> Be careful when writing backups to an NFS file system. You must consider the implications of using root on NFS file systems.

5.2.3 Running Backup from the Tivoli Desktop

Use the following steps to back up one or more machines in the TMR:

1. Place the TMR in maintenance mode. Note that this completely stops any other systems from communicating to the TMR server’s oserv.

> **Remember**
>
> Under most circumstances, putting the Tivoli oserv in maintenance mode will not affect day-to-day business operations in a live environment unless, for example, the oserv is a central system used for all event reporting. It has no effect on other applications in a system. The stop will simply make Tivoli management of that system wait until the oserv is taken out of maintenance mode. Certain independent processes, such as monitors will still run.

2. Select **Backup...** from the Desktop menu to display the Backup Tivoli Management Region dialog.

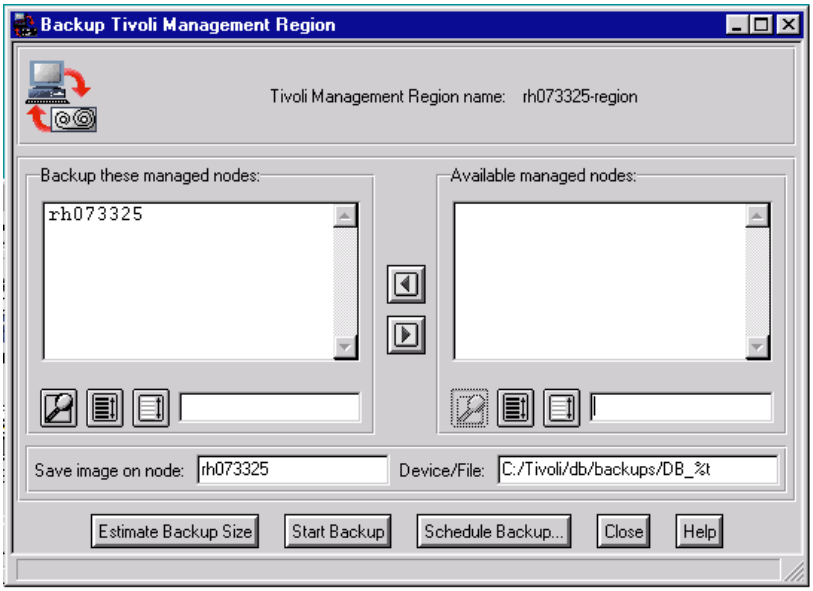


Figure 32. Backup Tivoli Management Region Dialog

3. Select one or more managed nodes from the Available managed nodes scrolling list and press the **left arrow** button. The chosen managed nodes are moved from the Available managed nodes scrolling list to the Backup these managed nodes scrolling list. You can select any combination of clients and the TMR server to be backed up. If you wish to back up the entire TMR, select all managed nodes listed in the Available managed nodes scrolling list. This dialog also allows you to set the destination node of the save image and the device and file name of the backup. The `%t` shown includes the date in the file name.
4. Specify the machine on which the backup image or device is located in the Save image on node field.
5. Specify the device or file name in which the backup is to be saved in the Device/File field.
6. Press the **Estimate Backup Size** button. The Estimate Backup Size dialog is displayed.
7. Assuming the estimated size can be adequately handled, press **Close** to return to the Backup Tivoli Management Region dialog. See Section 5.2.6, “Temporary Backup File Considerations” on page 120 for more information on space requirements during backup.

8. Press **Start Backup** to begin an immediate back up of the selected managed nodes. The Backup Status dialog is displayed, and the backup operation begins. You can also press **Schedule Backup...** to schedule a backup for a later time.
9. Press the **Close** button when the backup operation completes. The Backup Status dialog is closed, and you return to the Backup Tivoli Management Region dialog.
10. Repeat steps 2 through 9 to backup another set of managed nodes in the local TMR or press **Close** to close the Backup Tivoli Management Region dialog and return to the Desktop window.

5.2.4 Running Backup from the Command Line

The `wbkupdb` command backs up and restores Tivoli databases. You can provide a list of managed node names as arguments to the `wbkupdb` command. We give a couple of examples here, but for more information, see the on-line manual page for the `wbkupdb` command or the *Tivoli Framework Reference Manual*.

NT Users Note

The `wbkupdb` command calls a shell script. When the `-d` parameter is used with the `wbkupdb` command, you must specify the path where the file will be saved. In all cases, including the Windows NT DOS command prompt, you must specify the character *forward slash* (`/`) and not the backslash (\). If the backslash (\) is used the script will escape the following character, and the path and file name will not be interpreted correctly. You will see a message, such as `freopen failed with code 13`.

5.2.4.1 Examples of the wbkupdb Command

The following example backs up the TME10 database for all managed nodes in the TMR from which the `wbkupdb` was run. The backups are written to the user-defined file `/usr/backups/TMR1.bk`.

```
% wbkupdb -d /usr/backups/TMR1.bk
```

Important

The backup file will be created in the specified directory in the TMR server unless you specify that the backup will be created in another managed node. Use the `-h` option to specified the managed node in which the backup file will be created.

The second example backs up the database of a single managed node, `rh0255a`. In this example, a destination directory and file name are not specified. The backup is, therefore, written to the directory containing the Tivoli database directory under a subdirectory called backups. The subdirectory is created if it did not exist when the `wbkupdb` command was run:

```
% wbkupdb rh0255a
```

5.2.5 Backup Process Behind the Scenes

The following steps are taken by the backup process during a database backup:

1. The list of clients is generated if it was not passed through the backup dialog or the `wbkupdb` command. This list can be generated outside of the backup process using the command:

   ```
   wls /Library/BackupClient
   ```

2. The list of files needing to be backed up is found for the server and for the clients. These lists can be generated outside the backup process using the following commands:

   ```
   BIO='wlookup -r Classes TMF_BackupImpl'
   idlcall $BIO _get_server_files
   idlcall $BIO _get_client_files
   ```

3. The backup host is contacted, and the backup file is created and opened.
4. A popup dialog stating that the backup process is beginning is displayed on active desktops.
5. A transaction is begun.
6. For each managed node passed in through the dialog, command line, or generated list, the backup process will do the following:
   1. Determine if the managed node is the TMR server or a client.
   2. Contact the managed node and start snapshot method passing in the list of files to be backed up. The method executable is a shell script called $BINDIR/TAS/BACKUP/snapshot.sh for UNIX or %BINDIR%\TAS\BACKUP\snapshot for Windows NT.
   3. The managed node synchronizes its database to write any outstanding transactions.
   4. Database files are tarred and compressed.
   5. The database is synchronized a second time, and the files are tarred and compressed and compared to the first snapshot. If they match, the

backup process continues; otherwise, this step is repeated up to three more times before failing.

7. Data is sent back to the server using an established IOM channel. See Section 7.6.3, “Bulk Data Transfer and Inter-Object Messaging” on page 271 for more information about IOM.
8. The TMR server backs up its own database. By default, backups are put into a directory called backups, one directory up from $DBDIR ($DBDIR/../backups), with a name that includes a time and date stamp.
9. The transaction ends.
10. A notice is logged and sent to a notice group.
11. If the backup process fails, then messages may be written to the backupdb.log file.

5.2.6 Temporary Backup File Considerations

As you can see from the previous list of backup steps, when Tivoli performs a backup, it creates two copies of the backup file to compare. If you backup from the desktop, this compare file is written to the database directory. If you run backup from the command line using the `wbkupdb` command, the compare file is written to the current directory. In both cases, the directory in which the file is written should have as much available disk space as the largest compressed database. If the directory does not have enough space, you can change the directory in which this file is created by setting a Tivoli oserv environment variable. This is the TMPDIR environment variable on UNIX or the TEMP or TMP environment variables on Windows NT Systems.

The following steps detail how to set the TMPDIR environment variable specifically for the oserv. You can also use these steps to set the TEMP and TMP variables by substituting TMPDIR with the appropriate variable name:

1. Retrieve the current environment settings and write them to a file using the following command:

   ```
   odadmin environ get > file_name
   ```

2. Append the new TMPDIR setting to the file you created:

   ```
   echo “TMPDIR=/home/big_dir” >> file_name
   ```

3. Write the environment settings back to Tivoli using the following command:

   ```
   odadmin environ set < file_name
   ```

The temporary backup file is deleted when the backup procedure completes.

5.3 The Restore Process

If a disk drive on a Tivoli client fails, or the file system that stores the management information gets corrupted or is lost, the management data can be recovered by restoring the TMR server and/or Tivoli clients from an earlier backup. Tivoli usually reports irreparable damage to the object database using messages, such as `Persistent storage failure`. When this happens, there is usually little option but to revert to a backup of the database. You can also use this process to restore the database after upgrading a client or the server to a new version of the operating system.

You can restore one client, several clients, the TMR server, or the entire TMR. Since the nature of a restore operation affects the underlying database of a client or the server, you can only restore the data from the command line.

There is a distinction between a standard restore operation and a *rescue* operation. A standard restore can take place when a managed node's object dispatcher is running. If the system is in such a state that even the oserv cannot be started, then there is a rescue procedure to restore the database.

5.3.1 Restore Roles and Access Rights

You must have the *restore* or *super* role in the TMR to perform a restore. If you are performing a rescue operation, you must be *Administrator* (NT) or *root* (UNIX) on the machine where the crashed database is located.

Note

As with the backup, the Administrator also requires a valid user login name and a group name for the machine on which the backup file is stored. They also needs read permissions for the directory that contains the backup file. See also Section 5.2.2, "Backup Roles and Access Rights" on page 115.

5.3.2 Restore Example

The `wbkupdb` command not only backs up but also restores TME10 databases. You can provide a list of managed node names as arguments to the `wbkupdb` command. Here, we present an example of using `wbkupdb` to restore a system. For further information see the on-line manual page for the `wbkupdb` command or the *Tivoli Framework Reference Manual*.

The following command example restores a single managed node, rh0255a. The backup file used to restore the managed node is `/usr/backups/TMR1.bk`.

```
% wbkupdb -r -d /usr/backups/TMR1.bk rh0255a
```

Note

You cannot specify another machine in the `wbkupdb` command as a source for the restore action, so you have to be logged into the machine that has the backup file to make the restore from that file.

5.4 Rescue Operation

If the object dispatcher that is to be restored is not running (and presumably cannot be run because its database is corrupted or missing), you can extract the database manually and put the files in the correct location in the database directory. This process is known as a rescue operation. If you are performing a rescue operation, you must be Administrator (NT) or root (UNIX) on the machine where the crashed database is located. The following is an example using `csh` of a rescue of a TMR server.

Note

For Windows NT, instead of using the `uncompress -c` command shown in this example, use `compress -cd`.

```
# tar xvf /var/spool/Tivoli/backup.db shasta
x shasta, 1027749 bytes, 2008 tape blocks
# uncompress -c <shasta | tar tvf -
rwxr-sr-x 0/0    0   Jun 18 13:53 1994 file_versions.restore/
rwxr-sr-x 0/0    0   Jun 17 12:17 1994
file_versions.restore/RCS/
rwxr-sr-x 0/0    0   Jun 17 12:17 1994
file_versions.restore/.staging/
rwxr-sr-x 0/0    0   Jun 17 12:17 1994
file_versions.restore/.staging/RCS/
rw------- 0/1 786432    Jun 23 17:15 1994 imdb.bdb.restore
rw------- 0/1 16384    Jun 23 17:42 1994 notice.bdb.restore
rw------- 0/0    0   Jun 23 17:42 1994 notice.log.restore
rw------- 0/14194304    Jun 23 17:39 1994 odb.bdb.restore
rw------- 0/0    0   Jun 23 17:41 1994 odb.log.restore
rw------- 0/0  726    Jun 23 16:15 1994 odlist.dat.restore

# set olddir='pwd'

# cd $DBDIR                                  ! %DBDIR% for Windows NT

# uncompress -c < $olddir/shasta | tar xvf -
```

```
# set files="imdb.bdb odb.bdb odb.log odlist.dat"

# foreach fn ( 4s )
? mv $fn.restore $fn
? end
```

If you are rescuing a client rather than the TMR server, the `set files=` line should be:

```
set files="odb.adj odb.bdb odb.log"
```

If you need this process and do not have this great redbook handy, it is documented in the on-line manual page for `wbkupdb`. You should make a point of checking the main page for the release you intend to perform this operation on as file names, or the number of files may change.

5.5 Items Not Restored from a Backup

During a backup, the `wbkupdb` command also saves any old versions of files (file_versions) and the notification database (notice.bdb). Normally, these are not restored since you probably do not want to read old notices that have already been read. If you wish to restore the notices for some reason, such as a problem with the current notices database, you can restore them using the steps given in Figure 33. This example assumes the TMR server is called `odie`. The commands must be run at the TMR server; notices are not held on the managed nodes.

```
# cd $DBDIR/../backups
# ls -l
total 4064
-rw-r--r--   1 root    root   20722064   Nov  26  10:32 DB_Nov26-1032
# mkdir /tmp/work
# cd /tmp/work
# tar -xvf $DBDIR/../backups/DB_Nov26-1032 odie
# mv odie odie.Z
# olddir=‘pwd‘
# cd $DBDIR
# uncompress $olddir/odie
# tar -xvf $olddir/odie notice.bdb.restore
# tar -xvf $olddir/odie notice.log.restore
# odadmin shutdown
# mv notice.bdb.restore notice.bdb
# mv notice.log.restore notice.log
# odadmin start
```

Figure 33. Restoring the Notices Database

Note

In a Windows NT environment, use `compress -d` instead of the `uncompress` command shown.

The file_versions directory is also not restored. If you want to see old revisions of system files, the files can be moved from the file_versions.restore directory as necessary.

5.6 Troubleshooting Backup and Restore Operations

This section describes the pitfalls you are likely to encounter during backup and restore. Be sure to check for the backupdb.log file. If it has a non-zero file size, then look at the file for error messages. Before we describe some additional points, some of those we have already covered in this chapter can be summarized as follows:

Authorization Such as incorrect role or rights to the backup directory. See “Backup Roles and Access Rights” on page 115 and “Restore Roles and Access Rights” on page 121 for more information.

Temporary File Tivoli uses a temporary backup file as a staging site while it copies the databases. For more information, see “Temporary Backup File Considerations” on page 120.

Windows NT Use a forward slash for the `-d` parameter even when using Windows NT. See Section 5.2.4, “Running Backup from the Command Line” on page 118 for details.

Backup Location The backup will always go to the TMR server unless you specify otherwise. See Section 5.2.4.1, “Examples of the wbkupdb Command” on page 118 regarding the `-h` flag.

Notices Notices are not normally restored. See Section 5.5, “Items Not Restored from a Backup” on page 123 for more information on a manual restore process.

Troubleshooting Tip

The snapshot script can be edited with echo statements inserted to help identify where in the process a problem is occurring.

5.6.1 Restore with -r and - r -R Options

Restoring through `wbkupdb` with the `-r` option causes the oserv to be re-executed once the restore is complete. The object dispatcher will then use the changes.

Restoring with `-r -R` copies the backup files (`*.restore`) to the database directory without restarting the oserv. The changes will be picked up at the next oserv start or restart.

5.6.2 Changing the Default Backup Directory

The default backup directory does not get changed if the database directory changes. The backup process does not use the DBDIR environment variable. Use the following commands if you must change the default back up directory or need to view what the system thinks it is:

```
# wlookup TMRBackup                                !find backup class OID
1264987995.1.351#TMF_SysAdmin::InstanceManager#
# idlcall 1264987995.1.351 _get_default_device    !use idlcall to get clean output
"/var/spool/Tivoli/backups/DB_%t"#
# idlcall 1264987995.1.351 _set_default_device '"/usr/local/backup/DB_%t"'
```

Figure 34. Changing the Default Backup Directory

Other interesting methods in the backup class object include _get_default_host, which is the host that will be backed up if none are specified, and _get_server_files and _get_client_files that are the files to be backed up.

5.6.3 Database Cannot Be Backed Up

There may be occasions when Tivoli will refuse to backup the database. This could have been caused by an incomplete installation that was not identified as such. To identify this cause, use the following commands:

```
# wlookup -ar ManagedNode -n rh0255a
```

where `rh0255a` is the name of the machine in question.

A fully installed managed node will look something like this:

```
TaskExecute   1264987995.2.9#TMF_ManagedNodesureTaskExecute#
BackupClient  1264987995.2.13#TMF_Backup::Client#
imp_TMF_UI::DesktopList 1264987995.2.15#TMF_UI::DesktopList#
imp_TMF_UI::Extd_DesktopList    1264987995.2.28#TMF_UI::Extd_DesktopList#
```

If you are missing any of these entries, such as the `BackupClient` object, there is a potential that the Tivoli client was not fully installed and will need to be

removed and reinstalled. Note that since release 3.6.1, the backup process will automatically skip any clients that do not have backup objects.

Important

The above example shows what you would see for TMP 3.0, 3.1.1, and 3.1.2 but not TMP 3.2. You can apply the same method to see if a managed node was not completely installed. Compare the output of the `wlookup` between one node that does backup and another that does not.

5.6.4 Malformed ASCII Exception

Sometimes the database is too busy to be backed up. This is often characterized by the following error:

```
Error - Unknown internal error: shell method wrote malformed ASCII exception
```

This comes from the snapshot program and indicates that the database is being updated too frequently to tar off the database before it changes. An error message stating that the system is too busy also may display.

To resolve this message, you must either put the TMR in maintenance mode to make the backup or try to wait until the TMR server is not so busy.

5.6.5 IOM Route Time-Outs

Clients send their backup files to the server through an Inter-Object Messaging (IOM) channel. The clients must be able to contact the server by the IP address used to install the server. If the server does not get a response from the client, it will generate an IOM time-out error. This is a communication error. One common example is if the IP address of the TMR server has been changed. Be sure that there is communication between server and clients (for example, using `ping` by both name and address) and that you have followed the steps of changing the IP address in the TMR server. These steps are described in the *Tivoli Framework Planning and Installation Guide* in the chapter “Tivoli Maintenance and Troubleshooting.”

5.6.6 Identifying Managed Nodes

The backup file is a tar file of compressed tar files. Sometimes you need to know what machines are in a backup file. To do this, you need to use the `tar` command with the `-t` option as in the following example:

```
# tar -tvf $DBDIR/../backups/DB_Oct20-1609
-rw-------   0 3  2207243 Oct 20 11:09:44 1997 rh0255b.itsc.ibm.com
-rw-------   0 3    15256 Oct 20 11:09:51 1997 rh0255a
```

Figure 35. Using tar to Display the Machines in the Backup

If you want to check that all managed nodes have a backup object and will, therefore, be available for backups, you can use the following command to display a list of backup objects:

```
# wls -o /Library/BackupClient
```

An example of this command is shown in the following figure:

```
# wls -o /Library/BackupClient
1264987995.1.350#TMF_Backup::Client#    stout.tivoli.com
1264987995.2.13#TMF_Backup::Client#     rh0255a
```

Figure 36. Displaying Backup Objects for all the Managed Nodes

Note that the label for the backup object for the TMR server is unlikely to reflect the true name of the server. The name shown will be the name that was in the default database that gets used at install time to build the server database. For Windows NT, for example, the name will be *gator*, and for certain UNIX interpreters the name would be *stout*. Ordinarily, the backup object label is not exported to the user.

5.6.7 Implications of Using an Old Backup

Data would generally be restored from the most recent backup. Reverting to earlier backups could introduce new problems. This is demonstrated in the following example:

Suppose that you have two machines, a TMR server and a managed node, and you have three backups taken for these machines. The first was taken before the managed node install; the second was taken when the managed node was installed; and the third has both machines after an installation of an application (such as Tivoli Distributed Monitoring). The environment is shown in Figure 37 on page 128:

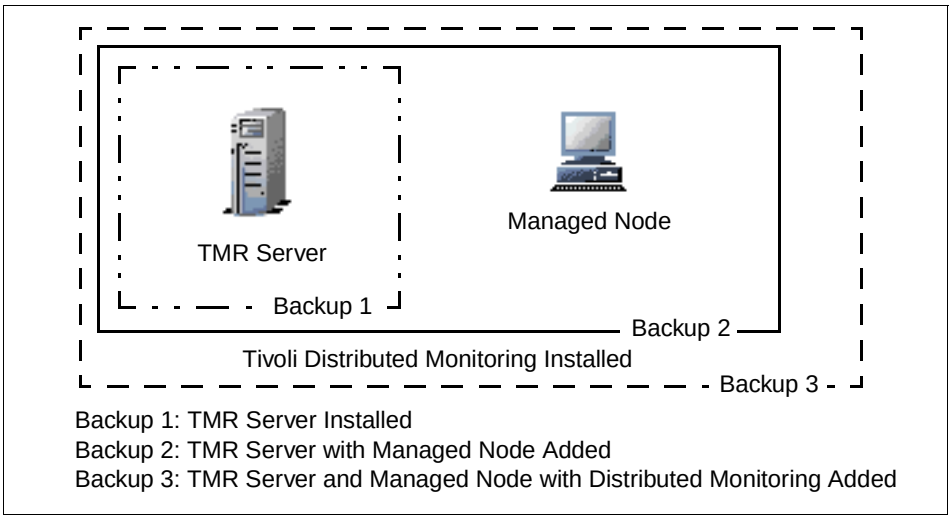


Figure 37. Example of Problem When Using Old Backups

If you restore Backup 1 (which applies to the TMR server only), the managed node does not exist in the backup; so, the TMR server database will no longer contain any record of it even though the managed node has Tivoli installed and running. If, after this, you restore Backup 2, it will only restore the server pieces because the managed node is not an object in the TMR server's database, and the TMR cannot find it even though it exists in the backup. After this, you could experience some strange behavior.

The machines can communicate with each other, but the managed node believes that TME10 Distributed Monitor is installed, but the TMR server does not have it installed after restoring Backup 2. Now, if you try to open the Tivoli Desktop on the managed node, an error occurs because it will try to find TME10 Distributed Monitor objects that are not defined in the TMR server.

To correct this problem, you have to restore Backup 2 a second time. Now, the TMR server has the object (managed node) defined from the previous restore of Backup 2, and it can send the Backup 2 restore to the managed node, and the TME10 Distributed Monitor objects will disappear.

This situation should only arise if there was a problem restoring from your last backup. Otherwise, in this example you would have used Backup 3 from the beginning.

Remember that a restore will restore the entire database and that managed node's memory of objects. Another potential problem would arise if a Tivoli

managed node was reinstalled since the last backup. If it were a complete reinstall, the object dispatcher number (the second part of the OID) would be a new number. To register the reinstalled node with the TMR server, the old version would first have to be removed. If the managed node was now restored from an old backup, the dispatcher number would revert to the old one, and the TMR server would no longer recognize it. The solution to this problem would be to have reliable backups taken following the reinstall of the managed node.

Chapter 6. Commands and Logs for Troubleshooting

When you are troubleshooting problems with Tivoli, there are a number of important commands that will help you. The three you will most commonly use for techniques described in this book are:

`odstat`	Lists currently-running methods and method histories.
`odadmin`	Lists the managed nodes in a TMR and configures different aspects of how the Tivoli object dispatcher (oserv) communicates, such as defining IP addresses and interconnected regions.
`wtrace`	Formats the odtrace.log, which is created when tracing objcalls, services or errors (tracing is invoked with `odadmin trace` options).

Tivoli performs logging by default during installations in the following locations:

- `/tmp` on UNIX
- `%DBDIR%\tmp` on Windows NT or OS/2

Additional logs can be found in the database directory that is locatable through the following variable names:

- `$DBDIR` on UNIX
- `%DBDIR%` on Windows NT or OS/2

The database directory contains other files that can be used as debugging tools:

epmgrlog	The endpoint manager log.
gatelog	The gateway log**.**
odb.log	The Tivoli database transaction log.
notice.log	The Tivoli notice database transaction log.
gwdb.log	The Tivoli gateway database transaction log.
oservlog	The error log of the Object Dispatcher (oserv).
odtrace.log	The file that the `wtrace` command reads and translates.

On a TMA endpoint, there is also a log file in the */lcf/dat/xx* path.

lcfd.log	The endpoint log.

In some cases, you may need to get into the more complex area of direct manipulation of the Tivoli object database. You can hand-run methods, identify method source files, and so on.

Important

Most activities described in this book do not involve changes to the object database. However, direct object invocations, IDL calls, and attribute changes in the Tivoli object database have the potential to cause unpredictable results and possibly the complete failure of your TMR. Recovery, if at all possible, would usually involve at least a restore of the object database. Tivoli support personnel are not able to assist in rectifying such changes. We recommend you backup your object database before performing any direct manipulation, test changes first on an isolated test TMR, and keep a log of every action performed.

Chapter 2, “Tivoli Object Database Architecture” on page 9 describes the object hierarchy in some detail and gives examples of using the following commands that are also briefly described in this chapter in “Other Commands” on page 182:

`objcall`	Performs a Tivoli object call (non-IDL) from the command line.
`idlcall`	Performs a Tivoli Extended IDL call from the command line.
`idlattr`	Gets or sets object implementation attributes.
`irview`	Views all method and attribute names and details for a resource.

We also give an example in this chapter of a command used to review transaction status:

`tmstat`	Displays the status of current transactions and locks.

The full syntax of these commands is given in the *Tivoli Framework Reference Manual*.

Another command sometimes used is `resolve`. This command finds the OID of an object that contains a given method or attribute. We are not covering this command as it is not a fully supported Tivoli command, does not appear in all platforms, and the same function can be achieved with the resolve method - an example of which is shown in Chapter 2, “Tivoli Object Database Architecture” on page 9.

6.1 The odstat Command

This command displays the history of methods executed on the node it is run on. The listing will include the last 200 methods run since the oserv was started including those that are currently running. The output is in the order of currently running first then the history of completed methods starting with the oldest. All of the data contained within an `odstat` output can be retrieved from

the output of `wtrace`. However, using `odstat` is a much friendlier way to view this data. You would use the `wtrace` output once you understood what was going on through `odstat`. A typical output using default parameters looks like Figure 38:

```
_active = 5  n_free = 195
tid type    ptid State  StdO  StdE    Start  Err    Method
 2 O+bhdoq        run     0     0   Wed13:58    1264987995.1.158#TMF_Scheduler::scheduler# start
 29 O+hdoqs       run     0     0   10:25:05    1264987995.1.478#TMF_UI::Extd_Desktop# uiserver
 32 O+hdoq   1-29 run     0     0   10:25:10    1264987995.1.179#TMF_Administrator::Configuration_GUI#
launch
 469 O+           run     0     0   16:31:17    1264987995.1.2 query odstat

---- history ----
  259 O+hdoq  1-29  done    18     0 13:59:34    1264987995.1.488#TMF_PolicyRegion::GUI# launch
  275 O+     1-259  done    55     0 13:59:35    1264987995.1.462#TMF_Query::QueryLibraryPD# _get_pres_object
  276 O+hdq  1-259  done    90     0 13:59:35    1264987995.1.488#TMF_PolicyRegion::GUI# get_policy_default
  277 O+     1-276  done    40     0 13:59:35    1264987995.1.407#TMF_NetWare::PD# _get_label
  278 O+     1-259  done    55     0 13:59:35    1264987995.1.407#TMF_NetWare::PD# _get_pres_object
  279 O+hdq  1-259  done   104     0 13:59:35    1264987995.1.488#TMF_PolicyRegion::GUI# get_policy_default
  280 O+     1-279  done    35     0 13:59:35    264987995.1.334#TMF_PcManagedNode::PcManagedNodePD# _get_label
  281 O+     1-259  done    55     0 13:59:35    1264987995.1.334#TMF_PcManagedNode::PcManagedNodePD#
_get_pres_object
  282 O+hdq  1-259  done    99     0 13:59:35    1264987995.1.488#TMF_PolicyRegion::GUI# get_policy_default
  283 O+     1-282  done    33     0 13:59:35    1264987995.1.316#TMF_ManagedNode::Managed_NodePD#
_get_label
  284 O+     1-259  done    27     0 13:59:35    1264987995.1.316#TMF_ManagedNode::Managed_NodePD#
_get_pres_object
  285 O+hdq  1-259  done   101     0 13:59:35    1264987995.1.488#TMF_PolicyRegion::GUI# get_policy_default
  286 O+     1-285  done    36     0 13:59:35    1264987995.1.271#TMF_CCMS::ProfileManagerPolDef# _get_label
  287 O+     1-259  done    55     0 13:59:35    1264987995.1.271#TMF_CCMS::ProfileManagerPolDef#
_get_pres_object
  288 O+hdq  1-259  done    91     0 13:59:35    1264987995.1.488#TMF_PolicyRegion::GUI# get_policy_default
  289 O+     1-288  done    33     0 13:59:35    1264987995.1.221#TMF_Task::TaskLibraryPD# _get_label
  290 O+     1-259  done    55     0 13:59:35    1264987995.1.221#TMF_Task::TaskLibraryPD# _get_pres_object
  291 O+     1-259  done     6     0 13:59:35    1264987995.1.489#TMF_UI::Presentation# _set_parents
  292 O+hdq  1-259  done   183     0 13:59:35    1264987995.1.14#TMF_SysAdmin::Library#
select_indirectly_managed
  293 O+hdoqs  1-259  done   6     0 13:59:35    1264987995.1.478#TMF_UI::Extd_Desktop# connect
  294 O+ho    1-29  done    18     0 13:59:40    1264987995.1.490#TMF_Task::TaskLibrary# display_view_callback
  295 O+     1-294  done    55     0 13:59:41    1264987995.1.490#TMF_Task::TaskLibrary# _get_pres_object
  296 O+     1-294  done    55     0 13:59:41    1264987995.1.490#TMF_Task::TaskLibrary# _get_pres_object
  297 O+     1-294  done 63754     0 13:59:41    1264987995.1.232#TMF_UI::Presentation# _get_dialogs
  298 O+hdoqs  1-294  done   6     0 13:59:41    1264987995.1.478#TMF_UI::Extd_Desktop# broadcast
  299 O+     1-294  done  7684     0 13:59:42    1264987995.1.232#TMF_UI::Presentation# _get_bitmaps
  300 O+hdq   1-29  done  3463     0 13:59:42    1264987995.1.232#TMF_UI::Presentation# get_icon_info
  301 O+hdq   1-29  done  3752     0 13:59:43    1264987995.1.237#TMF_UI::Presentation# get_icon_info
  302 O+            done    15     0 14:04:15    0.0.0 get_name_registry
  303 O+hdoq        done   109     0 14:04:15    1264987995.1.26 lookup
  304 O+hdq         done   131     0 14:04:15    1264987995.1.14#TMF_SysAdmin::Library# find_members
  305 O+hdq         done   195     0 14:04:15    1264987995.1.227#TMF_SysAdmin::InstanceManager# find_members
  306 O+            done    15     0 14:04:50    0.0.0 get_name_registry
  307 O+hdoq        done   109     0 14:04:50    1264987995.1.26 lookup
```

Figure 38. Typical odstat Output

You can see the two sections: The first containing active methods, and the second with a history of completed methods.

In the example in Figure 38, the first four lines are the method threads that are running. You can see that the threads are:

- Scheduler:

  ```
  2 O+bhdoq  run  0  0  Wed13:58 1264987995.1.158#TMF_Scheduler::scheduler# start
  ```

- Desktop:

  ```
  29 O+hdoqs   run  0  0  10:25:05 1264987995.1.478#TMF_UI::Extd_Desktop# uiserver
  32 O+hdoq 1-29 run  0   0   10:25:10
  ```

```
1264987995.1.179#TMF_Administrator::Configuration_GUI# launch
```

The first desktop thread (tid=29) is started by the `tivoli` command. The second (tid=32) is a child of the first and is the thread of the administrator desktop itself. We can see from this that one administrator currently has a desktop open. We could also look at the administrator object (OID `1264987995.1.179`) to find out who this administrator actually is. We could match the OID with one returned by `wlookup -ar Administrator`, or we look at the object label with something, such as `objcall OID _get_label`.

- `odstat` command:

```
469 O+          run   0   0   16:31:17   1264987995.1.2 query odstat
```

The rest of the lines are the history of completed methods.

6.1.1 Structure of the odstat Output

Refer to the following table during the explanation of the `odstat` output format (note that this table was built from a 2.6 `odstat`, but the principals are the same for all releases):

Table 3. Output from odstat

tid	type	ptid	State	Std O	Std E	Start	Err	Method
9	O+bhdoq		run	0	0	Fri09:35		111111.1.163#TMF_Scheduler::scheduler# start
10	O+bhdoq		run	0	0	Fri09:35		111111.1.546#SentryEngine::engine# run_engine
1056	O+hdoq		run	0	0	Fri11:41		111111.1.371#TMF_UI::Desktop# uiserver
1060	O+hdoq	1-1056	run	0	0	Fri11:41		111111.1.184#TMF_Administrator::Configuration_GUI# launch
1091	O+hdoq	1-1056	run	0	0	Fri11:41		111111.1.630#TMF_PolicyRegion::GUI# launch
1244	O+hdoq	1-1056	run	0	0	Fri11:42		111111.1.656#TMF_CCMS::ProfileManager# launch
1250	O+hdoq	1-1056	run	0	0	Fri11:42		111111.1.632#TMF_CCMS::ProfileManager# launch
1967	O+hdoq		run	0	0	Sat14:14		111111.1.372#TMF_UI::Desktop# uiserver
1974	O+hdoq	1-1967	run	0	0	Sat14:14		111111.1.639#TMF_Administrator::Configuration_GUI# launch
3025	O+hdoq		run	0	0	15:13:18		111111.1.373#TMF_UI::Desktop# uiserver
3029	O+hdoq	1-3025	run	0	0	15:13:21		111111.1.184#TMF_Administrator::Configuration_GUI# launch
3153	O+		run	0	0	15:15:11		111111.1.2 cntl odstat

The column headings are as follows:

1. `tid` - The thread ID. You may sometimes see two threads generated from a single object call. One for the object call itself, and one for the method being invoked.
2. `type` - The thread and method type. The thread type flags are:

`O` Object call thread (attached to an object request), indicating that the method was invoked here but is running elsewhere.

`M` Method thread. The object method was invoked on a different system, but the object is located on this system, and the method is, therefore, running here.

`O+` Object call and method threads are the same, indicating that the caller and method are both situated locally.

The method type flags are:

`a` Asynchronous object call.

`q` Queueing method.

`o` Per-object method.

`b` One-way object call.

`h` Help method.

`d` Daemon (long-running) method. typically handling one request at a time.

So `O+bhdoq` in the first row indicates that the object call and method threads are the same. This is a one-way invocation of a helping, queueing, daemon, per-object method implementation.

3. `ptid` **-** Parent thread ID. This is the thread ID of the object call whose method made the current object call. If this field is blank, the object call is external (such as from the command line). The number before the dash is the dispatcher number where the parent thread resides. This may be another system that you should investigate. The number after the dash is the thread ID in the parent's object dispatcher.

4. `State` - One of the following states for an object call thread:

`init` The thread has been initialized.

`ali` The thread is performing a TME server lookup on the TME server's database. If a thread is in this state for some time, it could be very difficult to get a response from the server. You may have difficulties getting the `odstat` output if this is the case. You should then try the `odstat -k` option. ALI is used to refer to the TMR server and stands for Authorization, Location, and Inheritance, the three major functions of the TMR server.

`mwait` The thread is waiting for the associated method thread to complete.

`rwait` The thread is waiting for the caller to collect the results of an asynchronous (often remote) object call. You will see these, for example, during a distribution.

`done` The object call is complete.

`coord` The method is serving as a transaction coordinator.

`err` An internal error terminated the thread.

The following is a list of states for method threads:

`init` The thread has been initialized.
`gmeth` The thread is obtaining the method code from another dispatcher.
`hdwt` The thread is waiting for the daemon-method process (of a non-queueing daemon) to be ready to accept another request.
`run` The method is running.
`serv` The thread is performing an object services call.
`done` The method is complete.
`twait` The method is waiting on transaction status to commit or abort.

5. `StdO` - Number of bytes written to standard output by the method (hardly used in recent releases).
6. `StdE` - Number of bytes written to standard error by the method. Most threads do not write to standard error.
7. `Start` - When the thread started. This will be a time if it is on the same day. Otherwise, it will be a named day and time. If the thread started more than a week ago, you would not be able to tell it from this output (you would have to refer to the `wtrace` output).
8. `Err` - The thread's error status. If this field is blank, no error occurred. Otherwise, this field is one of the following:

   `e=n` The method returned *n* (decimal) as its exit code. Tivoli avoids codes 0-21 reserving them for system-defined errors. Tivoli application error codes start at 22. You can refer to the o.h file in the Tivoli include directory if Tivoli ADE has been installed for a listing of generic Tivoli errors. See also the paragraph below about exit codes.
   `s=n` The method died due to signal *n*.
   `S=n` The method died due to signal *n* and produced a core file. The core file is unlikely to help anyone unless a debug version of a module was provided by Tivoli development.
   `XXX` An uppercase word indicates an error in the object dispatcher.

   Exit codes (`e=`) can come from the system Tivoli is running on, the Tivoli Framework, or an application with no indication of which it is. It may be necessary to look at different sets of error documentation to find out which is the most likely source. You may be able to use system documentation to obtain help regarding system-produced errors. Most UNIX systems include some form of error.h file that lists the system error codes and often a short description. On OS/2, you can type `HELP number` (such as `HELP 5`). On Windows NT, you can type `net helpmsg number` as follows:

```
net helpmsg 5
Access is denied.
```

```
net helpmsg 1067
The process terminated unexpectedly.
```

Note

System-generated errors can be misleading. There could be many generations involved between the object where the problem really exists and the object where an error was generated. Keep this in mind when looking at system-generated errors.

9. **Method** - Text of the method invocation. The first value is the object ID of the object in whose context the method was invoked. (See Section 2.4.2, “Object IDs” on page 34). Next is a pound sign (#), the CORBA Interface Repository name information, and another pound sign. The next word is the name of the method itself. Any following words are the arguments to the method.

   For example:

   ```
   111111.1.184#TMF_Administrator::Configuration_GUI# launch
   ```

   might be a valid entry. This is an interesting entry because from it we can tell that the administrator with object number 184 has this particular desktop open. (There are three administrator desktops open in the sample listing.) We can see a list of administrators and their object numbers using `wlookup -ar Administrator`. The same applies for the profile managers in the example listing. We could use `wlookup -ar ProfileManager` to compare the object numbers and determine which profile managers were in use.

   The `odstat` command shows the first 80 characters of the method invocation.

Note

For IDL-generated methods, arguments are not visible using `odstat`. See the `wtrace` command for more information.

6.1.2 odstat Options

There are many useful options for `odstat` (see the *Tivoli Framework Reference Manual* for more options and details). Here are a few examples of useful invocations:

`odstat`	List currently-running methods and history of most recent 200.
`odstat -c`	Lists only the currently-running methods.
`odstat -v`	(verbose output) Lists system process IDs and error codes for each method. This can be very useful and is usually a good

set of data to collect for support personnel, but it does make the listing more difficult to read. You may need to experiment to determine whether to use `-v` or not. You might wish to use the process ID to kill a hung process. If this is the case, do not `kill` UNIX Tivoli processes with `-9`; otherwise, the transactions may not close correctly.

`odstat -d` Lists oserv daemon methods that are currently running and belong to Tivoli. Some that you might expect to see here are not displayed in this command (such as the notices daemon).

`odstat -o OID` Runs the command on the specified object dispatcher.

`odstat -k $DBDIR <pid_of_oserv>`
Runs the command on an oserv that is running but not responding. You need root authority to run this command.

6.2 The odadmin Command

The `odadmin` command is the Object Dispatcher (oserv) Administration interface. This command provides configuration information about the server and allows you to change this information. To run this command, you will require a Tivoli administrator with *super* or *senior* role in the TMR.

The `odadmin` command has very good built-in help. Use `odadmin help`, or `odadmin help <option name>` for quick access to help information.

Note

The `odadmin` command requires the oserv daemon to be running to function fully. The oserv should start automatically when the managed node starts up. If you have to manually start the oserv in UNIX, you can run `odadmin start` or `/etc/Tivoli/oserv.rc start`. For Windows NT, use the **Services** icon in the Control panel to start the Tivoli Object Dispatcher service or use NET START OSERV from the command prompt.

6.2.1 Default odadmin Information

You can execute the `odadmin` command alone, and it will output information about the local object dispatcher as shown in Figure 39 on page 139:

```
# odadmin
Region = 1264987995
Dispatcher = 1
Interpreter type = aix4-r1
Database directory = /var/spool/Tivoli/rh0255b.itsc.austin.ibm.com.db
Install directory = /usr/local/Tivoli/bin
Inter-dispatcher encryption level = simple
Kerberos in use = FALSE
Remote client login allowed = TRUE
Install library path =
/usr/local/Tivoli/lib/aix4-r1:/usr/lib:/usr/local/Tivoli/install_dir/iblib/aix4-r1:/
usr/lib:/usr/local/Tivoli/lib/aix4-r1:/usr/lib:/usr/local/Tivoli/lib/aix4-r1:/usr/lib
TME 10 Framework (tmpbuild) #1 Thu Oct 3 08:08:58 CDT 1996
Copyright Tivoli Systems, an IBM Company, 1996.  All Rights Reserved.

Port range = (not restricted)
State flags in use = TRUE
State checking in use = TRUE
State checking every 180 seconds
Dynamic IP addressing allowed = FALSE
```

Figure 39. Default odadmin Information

The information displayed is as follows:

- The Tivoli Management Region number generated when the TMR server was installed.
- The object dispatcher number. Number `1` is the TMR server. Each subsequent client installed gets the next number available.
- The machine interpreter type.
- The path of the database directory.
- The path of the binaries directory.
- Encryption level used within the TMR.
- Whether Kerberos is being used within the TMR.
- Whether Remote desktop connection (using Tivoli Desktop for Windows) is allowed.
- The path of shared libraries (UNIX only).
- Version and copyright information.
- What port range the oserv is restricted to using when requiring access to TCP ports.
- `State flags in use` indicates whether down-stream host information is cached (`TRUE`) or whether the host is contacted each time (`FALSE`). This exists on the TMR server only.
- State checking indicates whether polling is used to keep host state information up to date.
- If state checking is in use, then the polling interval is used.
- Whether DHCP addressing is allowed.

For obtaining information on connected managed nodes, you can use the `odadmin odlist` command. This will show you the status of the managed node, its IP address, the port, and the host name alias. It also includes flags indicating the status of the node's TMR connection as shown in Figure 40:

```
# odadmin odlist
Region      Disp  Flags  Port            IPaddr   Hostname(s)
1264987995     1    ct-    94         9.3.1.234   rh0255b.itsc.austin.ibm.com
               2    ct-    94         9.3.1.233   rh0255a.itsc.austin.ibm.com
1562489759     1    ct-    94         9.3.1.235   rh0255c.itsc.austin.ibm.com
```

Figure 40. Sample odadmin odlist

If the first flag is a question mark (`?`), this indicates that the state of the connection is unknown due to the cache being out of date. If it is a minus sign (`-`), then the remote dispatcher is down. A `c` indicates that the local oserv is connected to the remote oserv.

6.2.2 Configuring the TMR Server

The configuration above can be changed using the different options of the `odadmin` command. Use `odadmin help` and `odadmin help <option name>` or refer to the *Tivoli Framework Reference Manual* for more detail on these options.

The one option we use a number of times in this chapter, and frequently when looking into problems, is the `trace` option. The `trace` option defines how much information the oserv writes to the odtrace.log file. See the next section on `wtrace` for more information about how we use this parameter.

6.3 The wtrace Command

The `wtrace` command reads the file in the database directory, called odtrace.log, that was generated by the oserv. This file is a static 1 MB in size and resides in $DBDIR, (%DBDIR% for NT or OS/2). The trace file size can be changed using the -t switch on the `oserv` command. You will need to manually start the oserv in order to specify this switch.

The rate at which `odadmin` data is written to the file depends on the oserv's trace level. This section gives some directions on how to use `wtrace` to investigate method problems. More detail on the command syntax and output format can be found in the *Tivoli Framework Reference Manual*.

Three types of methods can be tracked by oserv in the trace set with the `odadmin trace` command:

`errors`	Default setting. Record errors in the oservlog file.
`objcalls`	Usual setting to use for problem determination. Trace object method invocation.
`services`	Track Authentication, Location, and Inheritance (ALI) services requested of the TMR server. This will quickly fill a trace file.

Once the data is collected, it is in the file. This is important to remember because, unlike `odstat`, `wtrace`, this can provide debugging information if the oserv is down or has been recycled.

The `wtrace` output contains everything that is also found in an `odstat` output; however, it is quite detailed and can be hard to read. Having a corresponding `odstat` output can make reading `wtrace` much easier. See Section 6.1, “The odstat Command” on page 132 for more information on `odstat`.

6.3.1 Trace Usage Overview

If you are having problems that are not easily resolved by the messages Tivoli provides, the following is a suggested sequence of steps to use tracing to help:

1. Set the tracing option with the `odadmin trace` command. You can use the following options (in order because errors tends to switch the others off):

`odadmin trace errors`	Turns on error tracing (recommended)
`odadmin trace services`	Turns on service tracing (optional)
`odadmin trace objcalls`	Turns on object call tracing (recommended)

2. Execute the command `odstat`. This will help separate the methods that are involved in the problem by inserting two objcalls into the trace. Once you are familiar with reviewing traces, you may wish to skip this step and rely solely on finding process IDs or some other method.
3. Regenerate the problem.
4. Execute another `odstat -v> /<PATH>/odstat_output`.
5. Execute a `wtrace -jk $DBDIR > /<PATH>/wtrace_output` (optionally, use `-jHk` if the trace mentions binary hex data to get the hex data dumped in the output with an attempted ASCII translation).
6. Sometimes you also need the oservlog file.
7. Turn off the trace with `odadmin trace off`.
8. Turn on the default with `odadmin trace errors`.

Note

The tracing introduces some impact on the oserv. Some timing-sensitive problems have been known to go away once tracing is started. After investigating a system, you should always return to tracing errors only. Leaving other tracing enabled will have a negative performance impact.

Figure 41 is an example of the `wtrace` output. For each record, the first line will contain the thread ID, the thread and method flags, the parent thread ID, if one exists, and any error codes. The next few lines are similar to the rest of the data in an `odstat` output. A useful extra in the trace data is the name of the Tivoli principal that effectively called the method and the path of the method executable:

```
  A      B       C       D          E         F
loc-ic   830    M-H    1-828        0        e=12
      Time run:    [Mon 16-Nov 00:35:50]
      Object ID:   1212391543.1.0  <--------- G
      Method:      get_name_registry <---- H
      Method Args: NameRegistry  <----------- I
      Principal:   TIVTOR\Administrator@gblnt.dev.t (0/0)  <--- J
      Path:        getattr <-------------- K
      Trans Id:
             {
               1212391543:1,1212391543:1,39:1166 <---- L
             },
             {
               1212391543:1,1212391543:1,39:1167 <---- M
             },
             {
               1212391543:1,1212391543:1,39:1168 <---- N
             },
```

Figure 41. Typical wtrace Output

A Trace block type code. This example is a local error block.
B Method TID.
C Method type.
D Calling method TID or PTID.
In this sample, TID 828 active on dispatcher 1.
E Size of buffer.
F Error code field.
See `/Tivoli/include/<$INTERP>/tivoli/defines.h` if you have ADE.
G Context object ID.
For `-ic`, `-oc`, `-ec`, and `-eo`, the object that was the target of the object call.
For `-is` or `-os`, the object that is making the service call.

H Method called.
I Arguments to the method.
J Principal (Administrator) that will service object request.
K Full path to program that will service object request.
LMN Active transactions, (See 6.4.1, “Transaction Log Files and tmstat” on page 172).

6.3.2 Using a Trace to Investigate a Method

To find the method, you will use the two output files of the `odstat` and `wtrace` commands:

1. From the `odstat` output, identify the thread ID where the error occurred (column 1 in Table 3 on page 134).
2. Find the same tid in the `wtrace` output.
3. Identify the Method, Input Data, and Results for that method.
4. You can then identify the principal that ran the method, for example:

   ```
   Principal:   root@rh0255b.itsc.austin.ibm.com (-2/-2)
   ```

   The first part is the user that ran the method, and the numbers in the parentheses are the user ID (`-2`) and group ID (`-2`) actually used to run the method.

   Note

   When the numbers in the parentheses are less than `0`, the method is run as *nobody*.

5. If you are experienced and feel confident running this test, you can execute the method manually to reproduce the problem and run tests; in general, this would only be under guidance from an authorized support individual:

   ```
   objcall OID <Method> <Input Data>
   or
   idlcall OID <Method> <Input Data>
   ```

Note

If you are going to invoke any methods by hand, be sure to check any warnings given in this chapter and review Chapter 2, “Tivoli Object Database Architecture” on page 9 and the *Tivoli Framework Reference Manual* for more information. Experimentation should always be limited to a stand-alone test system. Be sure to have excellent backups before trying any of this in a live environment, which you do at your own risk.

Other actions you could take may include looking at the contents of methods or finding the executable for a method. Refer to Section 2.5.1, “Finding the Method Executable” on page 52 for more information about methods and executables.

6.3.3 Troubleshooting a Failure with odstat and wtrace

Figure 42 on page 145 shows an `odstat` taken during a problem using the `wln` command.

```
n_active = 5  n_free = 195
tid type ptid State  StdO  StdE    Start   Err    Method
3 O+b         run     0     0
21:13:42   1367707114.1.327#TMF_ManagedNode::Managed_Node# logfi
5 O+bhdoq     run     0     0 21:13:42   1367707114.1.699#SentryEngine::engine#
run_engine
6 O+bhdoq     run     0     0 21:13:42   1367707114.1.158#TMF_Scheduler::scheduler#
start
296 O+        run     0     0 21:15:41   1367707114.1.2 query odstat
---- history ----
113 O+   1-5  done    15     0 21:13:52   0.0.0 get_name_registry
114 O+hdoq 1-5  done  9954     0 21:13:52   1367707114.1.26 region_get_all
115 O+        done    15     0 21:13:55   0.0.0 get_name_registry
116 O+hdoq    done   109     0 21:13:55   1367707114.1.26 lookup
117 O+hdq     done   132     0 21:13:55   1367707114.1.14#TMF_SysAdmin::Library#
find_members
118 O+hdq     done   127     0 21:13:55
1367707114.1.185#TMF_SysAdmin::InstanceManager# find_members
119 O+hdoq    done    97     0 21:13:55   1367707114.1.26 lookup
120 O+hdq     done    56     0 21:13:55   1367707114.1.4 lookup_id
121 O+hdq     done   168     0 21:13:55   1367707114.1.4##2@TMF_PolicyRegion::GUI
describe
122 O+hdq     done  4614     0 21:13:55   1367707114.1.4##2@TMF_PolicyRegion::GUI
_get_type
124 O+hdoq    done   109     0 21:13:55   1367707114.1.26 lookup
125 O+hdq     done   131     0 21:13:55   1367707114.1.14#TMF_SysAdmin::Library#
find_members
126 O+hdq     done   134     0 21:13:55
1367707114.1.322#TMF_SysAdmin::InstanceManager# find_members
*  127 O+hdq  done    90     0 21:13:55  e=12  1367707114.1.1116#TMF_PolicyRegion::GUI#
add_object
*  128 O 1-127  done     0     0 21:13:55 SCALL
1367707114.34.7#TMF_ManagedNode::Managed_Node# move_to_policy_region
```

Figure 42. odstat - wln

In the above example, we see that a failure occurred in thread ID 128 `move_to_policy_region` that was spawned by tid 127 `add_object`. This failure causes tid 127 to post an `e=12` message, which means the method threw an exception. To find out why, we need to look at a `wtrace`.

The sequence of events was initiated by the command `wln` when an Administrator attempted to link a managed node to a policy region.

The following `wtrace` will show more detail on this failure in tid 128:

```
loc-ic   118  M-hdq   Extern       45
      Time run:    [Mon 18-May 21:13:55]
      Object ID:   1367707114.1.185#TMF_SysAdmin::InstanceManager#
      Method:      find_members
      Principal:   root@hermes.torsm.can.ibm.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/Collection_prog1
      Input Data: (encoded):
             {
               0
             }
              "nt4-region$"  2

loc-oc   118                     127
      Results: (encoded):
             {
               1
               [
                 {
                   "1367707114.1.1116#TMF_PolicyRegion::GUI#" "nt4-region"
                 }
               ]
             }
              "OBJECT_NIL"

loc-ic   126  M-hdq   Extern       46
      Time run:    [Mon 18-May 21:13:55]
      Object ID:   1367707114.1.322#TMF_SysAdmin::InstanceManager#
      Method:      find_members
      Principal:   root@hermes.torsm.can.ibm.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/Collection_prog1
      Input Data: (encoded):
             {
               0
             }
              "^TEST01-MN$"  2

loc-oc   126                     134
      Results: (encoded):
             {
               1
               [
                 {
                   "1367707114.34.7#TMF_ManagedNode::Managed_Node#" "TEST01-MN"
                 }
               ]
             }
              "OBJECT_NIL"
```

Figure 43. Link Failure wtrace Part 1 of 2

```
loc-ic   127  M-hdq   Extern       57
      Time run:    [Mon 18-May 21:13:55]
      Object ID:   1367707114.1.1116#TMF_PolicyRegion::GUI#
      Method:      add_object
      Principal:   root@hermes.torsm.can.ibm.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/Policy_prog1
      Trans Id:
             {
               1367707114:1,1367707114:1,169:3
             },
             {
               1367707114:1,1367707114:1,169:4
             }
             #3
      Input Data: (encoded):
             "1367707114.34.7#TMF_ManagedNode::Managed_Node#"

rem-ic   128    M-H    1-127       51
      Time run:    [Mon 18-May 21:13:55]
      Object ID:   1367707114.34.7#TMF_ManagedNode::Managed_Node#
      Method:      move_to_policy_region
      Principal:   root@hermes.torsm.can.ibm.com (0/0)
      Path:        /w32-ix86/TAS/CCMS/profile_organizer
      Input Data: (encoded):
             "1367707114.1.1116#TMF_PolicyRegion::GUI#"

rem-oc   128 SCALL                  0

loc-oc   127 e=12                  90
      Results: (encoded):
             "Exception:StExcep::SystemException:StExcep::OBJ_ADAPTER"
             {
               38 1
             }
```

Figure 44. Link Failure wtrace Part 2 of 2

In thread 118, we get the OID of the `nt4-region` policy region. This is the policy region to which we will link the Managed Node. Below is the equivalent `idlcall` to find the member from the InstanceManager:

```
idlcall 1367707114.1.185 find_members '{ 0 } "nt4-region" 2'
```

In thread 126, we get the OID of the *TEST01-MN* managed node that we will link to the policy region above. Below is the `idlcall`:

```
idlcall 1377707114.1.322 find_members '{ 0 } "TEST01-MN" 2'
```

Both tid 118 and tid 126 are executed as *nobody* (-2/-2).

In thread 127, we start to add the managed node object to the policy region. Note the OID for the Object ID and in the input data as gathered above. This

spawns thread 128, which will be executed remotely on the Windows NT managed node as seen by the `w32-ix86` path of the executable and the `rem-ic`.

Thread 127 is executed as *nobody* (-2/-2), but tid 128 is executed as *root* (0/0) and fails with SCALL on the remote object dispatcher. Thread 127 now aborts with an OBJ_ADAPTER system exception, error 38.

A quick check on this managed node shows that the oserv service is inactive.

This problem is a very simple example, but the objective is to understand what we see in the `wtrace`.

6.3.4 Another Example of Analyzing wtrace and odstat

Figure 45 shows the output from `odstat` when an un-subscribe of a TMA endpoint from a profile manager fails.

```
* 4791 O+hdoq 1-4614  done   367     0 12:42:20  e=12
1998892590.1.1121#TMF_CCMS::ProfileManager# unsubscribe
  4792 O+    1-4791  done    15     0 12:42:20        0.0.0 get_name_registry
  4793 O+hdoq 1-4791  done   109     0 12:42:20        1998892590.1.26 lookup
  4794 O+hdq 1-4791  done    64     0 12:42:20
1998892590.1.14#TMF_SysAdmin::Library# lookup_object
  4795 O+hdq 1-4791  done     9     0 12:42:20
1998892590.1.958#TMF_PolicyRegion::GUI# is_validation_enabled
  4796 O+hdoq 1-4791  done   111     0 12:42:20        1998892590.1.26 lookup
* 4797 O+hdoqs 1-4791  done   367     0 12:42:20  e=12
1998892590.1.517#TMF_LCF::EpMgr# get_endpoint_key_value
```

Figure 45. Un-Subscribe Endpoint Failure odstat

In this example, we can see from tid 4791 that we have experienced a failure when attempting to unsubscribe a TMA endpoint from a profile manager. Thread 4797, which is spawned by this process, indicates a failure with the endpoint key value in the endpoint manager database (`$DBDIR/epmgr.bdb`)

Let us take a closer look at what steps were performed by reviewing the `wtrace` that follows:

```
loc-ic  4791 M-hdoq   1-4614      131
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.1121#TMF_CCMS::ProfileManager#
      Method:      unsubscribe
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        /aix4-r1/TAS/CCMS/profile_organizer
      Trans Id:
             {
               1998892590:1,1998892590:1,6:27845
             }
             #4
      Input Data: (encoded):
             {
               1
               [
                 {
                   "1998892590.9.508+#TMF_Endpoint::Endpoint#" "pctmp112"
                 }
               ]
             }
              2
             {
               "null" 0 false
             }

loc-is  4791 getattr        0       flags
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.1121#TMF_CCMS::ProfileManager#
      Method:      unsubscribe

loc-is  4791 lock           11
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.1121#TMF_CCMS::ProfileManager#
      Method:      unsubscribe
      Input Data: (ascii):         subscribers

loc-ic  4792    M-H   1-4791        0
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.0
      Method:      get_name_registry
      Method Args: NameRegistry
      Principal:   root@itso2.dev.tivoli.com (0/0)
      Path:        getattr
      Trans Id:
             {
               1998892590:1,1998892590:1,6:27845
             },
             {
               1998892590:1,1998892590:1,6:27846
             }
             #3
loc-oc  4792                       15
      Results: (ascii):         1998892590.1.26
```

Figure 46. Un-Subscribe Failure wtrace - Part 1 of 4

```
loc-ic  4793 M-hdoq   1-4791       42
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.26
      Method:      lookup
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "Library"

loc-oc  4793                      109
      Results: (encoded):
             {
               "1998892590.1.14#TMF_SysAdmin::Library#" "Library"
               {
                 "null" 0 false
               }
             }

loc-ic  4794  M-hdq   1-4791       37
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.14#TMF_SysAdmin::Library#
      Method:      lookup_object
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/Collection_prog1
      Input Data: (encoded):
             "ProfileManager"
             {
               0
             }

loc-is  4794 getattr        0       members
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.14#TMF_SysAdmin::Library#
      Method:      lookup_object

loc-oc  4794                      64
      Results: (encoded):
             "1998892590.1.288#TMF_SysAdmin::InstanceManager#"
loc-ic  4795  M-hdq   1-4791       58
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.958#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/Policy_prog1
      Trans Id:
             {
               1998892590:1,1998892590:1,6:27845
             },
             {
               1998892590:1,1998892590:1,6:27847
             }
             #3
      Input Data: (encoded):
             "1998892590.1.288#TMF_SysAdmin::InstanceManager#"
```

Figure 47. Un-Subscribe Failure wtrace - Part 2 of 4

```
loc-is  4795 getattr        0       classes
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.958#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
loc-os  4795 getattr        1.8K
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.958#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
      Method Args:
                0.1.261#TMF_Query::QueryLibraryPD#d„
      Results: (binary) (dump suppressed)
loc-oc  4795                        9
      Results: (encoded):
             false

loc-is  4791 lock           13
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.1121#TMF_CCMS::ProfileManager#
      Method:      unsubscribe
      Input Data: (ascii):         subscriptions
loc-ic  4796 M-hdoq   1-4791       50
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.26
      Method:      lookup
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        /aix4-r1/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "EndpointManager"
loc-oc  4796                       111
      Results: (encoded):
             {
               "1998892590.1.517#TMF_LCF::EpMgr#" "EndpointManager"
               {
                 "null" 0 false
               }
             }
loc-ic  4797 M-hdoq   1-4791       76
      Time run:    [Fri 20-Nov 12:42:20]
      Object ID:   1998892590.1.517#TMF_LCF::EpMgr#
      Method:      get_endpoint_key_value
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Path:        __epmgr_implid
      Trans Id:
             {
               1998892590:1,1998892590:1,6:27845
             },
             {
               1998892590:1,1998892590:1,6:27848
             }
             #3
      Input Data: (encoded):
             "1998892590.9.508+#TMF_Endpoint::Endpoint#"  "subscriptions"
```

Figure 48. Un-Subscribe Failure wtrace - Part 3 of 4

```
loc-oc  4797 e=12                367
      Results: (encoded):
             "Exception:UserException:SysAdminException::ExException:SysAdminE
             xception::ExInvalid:SysAdminException::ExNotFound"
             {
               "Exception:UserException:SysAdminException::ExException:
               SysAdminException::ExInvalid:SysAdminException::ExNotFound"
               "TasExCat" 4 "%5$t
               {
                 %c
               }
                 (%3$d): resource ‘%7$s’ not found" 911587340
               {
                 0
               }
               ""
             }

loc-oc  4791 e=12                367
      Results: (encoded):
             "Exception:UserException:SysAdminException::ExException:SysAdminE
             xception::ExInvalid:SysAdminException::ExNotFound"
             {
               "Exception:UserException:SysAdminException::ExException:
               SysAdminException::ExInvalid:SysAdminException::ExNotFound"
               "TasExCat" 4 "%5$t
               {
                 %c
               }
                 (%3$d): resource ‘%7$s’ not found" 911587340
               {
                 0
               }
               ""
             }
```

Figure 49. Un-Subscribe Failure wtrace - Part 4 of 4

We begin our operation with the un-subscribe in tid 4791 where we see the OID of the profile manager as well as the TMA endpoint.

Next, the profile manager is locked for the pending change.

In tid 4792, we get the OID of the name registry, and in tid 4793, we perform a lookup of distinguished object Library, and from this, we find the profile manager class object, or instance manager, in tid 4794.

We next check for validation policies on the policy region for this profile manager in tid 4795:

```
idlcall 1998892590.1.958 is_validation_enabled 1998892590.1.288
```

In tid 4796, we perform another lookup to get the OID of the endpoint manager:

```
idlcall 1998892590.1.26 lookup '"distinguished" "EndpointManager"'
```

Thread 4797 attempts to get the endpoint key value for pctmp112 from the endpoint manager, which fails with *resource not found*.

In this example, another Administrator had actually deleted the Endpoint with a `wdelep` command, the subscriber reference, however, still exists in the GUI.

A `wchkdb -u` corrects this and removes the subscriber from the Profile Manager.

6.3.5 Troubleshooting Using Only wtrace

The following `wtrace` is from a subscription failure:

```
loc-ic   826 M-hdoq    1-719      174
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      subscribe
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TAS/CCMS/profile_organizer
      Trans Id:
             {
               1212391543:1,1212391543:1,64:410
             }
             #4
      Input Data: (encoded):
             {
               16777216
               [
                 {
                   {
                     {
                       "1998892590.22.508+#TMF_Endpoint::Endpoint#" "rh2900b-ep"
                     }
                     "OBJECT_NIL" "" ""
                   }
                   0 false
                   {
                     0
                   }
                 }
               ]
             }

loc-is   826 getattr        0       label
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      subscribe
loc-os   826 getattr        17
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      subscribe
      Results: (binary) (dump suppressed)

loc-is   826 getattr        0       pro
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      subscribe
loc-os   826 getattr        50
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      subscribe
      Results: (binary) (dump suppressed)
```

Figure 50. Subscribe Failure wtrace - Part 1 of 6

```
loc-ic   827    M-H    1-826        0
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.0
      Method:      get_name_registry
      Method Args: NameRegistry
      Principal:   TIVTOR\Administrator@dhcp32-186. (0/0)
      Path:        getattr
      Trans Id:
             {
               1212391543:1,1212391543:1,64:410
             }
             {
               1212391543:1,1212391543:1,64:411
             }
             #3
loc-oc   827                        15
      Results: (ascii):         1212391543.1.26

loc-ic   828 M-hdoq    1-826       42
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.26
      Method:      lookup
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "Library"
loc-oc   828                       109
      Results: (encoded):
             {
               "1212391543.1.14#TMF_SysAdmin::Library#" "Library"
               {
                 "null" 0 false
               }
             }

loc-ic   829  M-hdq    1-826       37
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.14#TMF_SysAdmin::Library#
      Method:      lookup_object
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TMF/BASESVCS/Collection_prog1
      Input Data: (encoded):
             "ProfileManager"
             {
               0
             }
loc-is   829 getattr        0       members
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.14#TMF_SysAdmin::Library#
      Method:      lookup_object
```

Figure 51. Subscribe Failure wtrace - Part 2 of 6

```
loc-os   829 getattr        1.8K
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.14#TMF_SysAdmin::Library#
      Method:      lookup_object
      Method Args: venBased„
      Results: (binary) (dump suppressed)
loc-oc   829                      64
      Results: (encoded):
             "1212391543.1.287#TMF_SysAdmin::InstanceManager#"
loc-ic   830  M-hdq    1-826       58
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.195#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TMF/BASESVCS/Policy_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,64:410
             },
             {
               1212391543:1,1212391543:1,64:412
             }
             #3
      Input Data: (encoded):
             "1212391543.1.287#TMF_SysAdmin::InstanceManager#"
loc-is   830 getattr        0       classes
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.195#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
loc-os   830 getattr        1.8K
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.195#TMF_PolicyRegion::GUI#
      Method:      is_validation_enabled
      Method Args:
                543.1.1376#PcRC::RemoteControlPD#d
      Results: (binary) (dump suppressed)
loc-oc   830                       9
      Results: (encoded):
             true
loc-ic   831    M-H    1-826      109
      Time run:    [Sun 22-Nov 12:05:42]
      Object ID:   1212391543.1.195#TMF_PolicyRegion::GUI#
      Method:      pm_val_subscribers
      Method Args: pm_val_subscribers
      Principal:   TIVTOR\Administrator@dhcp32-186. (0/0)
      Path:        getattr
      Trans Id:
             {
               1212391543:1,1212391543:1,64:410
             },
             {
               1212391543:1,1212391543:1,64:413
             }
             #3
```

Figure 52. Subscribe Failure wtrace - Part 3 of 6

```
 Input Data: (encoded):
             {
               50331648
               [
                 "pm_val_subscribers" "Sentry" "rh2900b-ep"
               ]
             }
             {
               0
             }
             {
               0
             }
loc-oc   831                        40
      Results: (encoded):
             {
               67108864 "0x54 0x52 0x55 0x45 "
             }
             {
               0
             }
              0

loc-ic   832    M-H    1-826        0
      Time run:    [Sun 22-Nov 12:05:43]
      Object ID:   1212391543.1.1281#TMF_CCMS::ProfileManager#
      Method:      _get_flags
      Method Args: flags
      Principal:   TIVTOR\Administrator@dhcp32-186. (0/0)
      Path:        i_getattr
loc-oc   832                        12
      Results: (encoded):
             16777216

loc-ic   833 M-hdoq    1-826       50
      Time run:    [Sun 22-Nov 12:05:43]
      Object ID:   1212391543.1.26
      Method:      lookup
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "EndpointManager"
loc-oc   833                       111
      Results: (encoded):
             {
               "1212391543.1.517#TMF_LCF::EpMgr#" "EndpointManager"
               {
                 "null" 0 false
               }
             }
```

Figure 53. Subscribe Failure wtrace - Part 4 of 6

```
loc-ic   834 M-hdoq    1-826       77
      Time run:    [Sun 22-Nov 12:05:43]
      Object ID:   1212391543.1.517#TMF_LCF::EpMgr#
      Method:      get_endpoint_key_value
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        __epmgr_implid
      Trans Id:
             {
               1212391543:1,1212391543:1,64:410
             },
             {
               1212391543:1,1212391543:1,64:414
             }
             #3
      Input Data: (encoded):
             "1998892590.22.508+#TMF_Endpoint::Endpoint#"  "subscriptions"
loc-ic   835 M-hdoq    1-834       46
      Time run:    [Sun 22-Nov 12:05:44]
      Object ID:   1212391543.1.26
      Method:      lookup
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        /w32-ix86/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "InterRegion"
loc-oc   835                      120
      Results: (encoded):
             {
               "1212391543.1.378#TMF_InterRegion::Connection#" "InterRegion"
               {
                 "null" 0 false
               }
             }
loc-ic   836   M-ho    1-834       15
      Time run:    [Sun 22-Nov 12:05:44]
      Object ID:   1212391543.1.378#TMF_InterRegion::Connection#
      Method:      get_connection
      Principal:   TIVTOR\Administrator@dhcp32-186. (10104368/0)
      Path:        g:\Tivoli\bin/w32-ix86/TAS/InterRegion/InterRegion_prog1.exe
      Input Data: (encoded):
             "NULL"  783164535
loc-is   836 getattr        0       TMRs
      Time run:    [Sun 22-Nov 12:05:44]
      Object ID:   1212391543.1.378#TMF_InterRegion::Connection#
      Method:      get_connection
loc-os   836 getattr        1.8K
      Time run:    [Sun 22-Nov 12:05:44]
      Object ID:   1212391543.1.378#TMF_InterRegion::Connection#
      Method:      get_connection
      Method Args: nection#d„
      Results: (binary) (dump suppressed)
```

Figure 54. Subscribe Failure wtrace - Part 5 of 6

```
loc-oc   836                    2.0K
      Results: (binary) (dump suppressed)
rem-ic   837    M-H    1-834       50
      Time run:    [Sun 22-Nov 12:05:44]
      Object ID:   1998892590.1.26
      Method:      lookup
      Principal:   TIVTOR\Administrator@dhcp32-186. (0/0)
      Path:        /aix4-r1/TMF/BASESVCS/TNR_prog1
      Input Data: (encoded):
             "distinguished"  "EndpointManager"
rem-oc   837                      111
      Results: (encoded):
             {
               "1998892590.1.517#TMF_LCF::EpMgr#" "EndpointManager"
               {
                 "null" 0 false
               }
             }
rem-ic   838    M-H    1-834       77
      Time run:    [Sun 22-Nov 12:05:45]
      Object ID:   1998892590.1.517#TMF_LCF::EpMgr#
      Method:      get_endpoint_key_value
      Principal:   TIVTOR\Administrator@dhcp32-186. (0/0)
      Path:        __epmgr_implid
      Input Data: (encoded):
             "1998892590.22.508+#TMF_Endpoint::Endpoint#"  "subscriptions"
rem-oc   838                      577
      Results: (encoded):
                         1998892590.1.1158#TMF_CCMS::ProfileManager# "dataless
                    subsc_all FALSE
                    {
                         "Sentry2.0"
                    {
                         1998892590.1.1163#TMF_CCMS::ProfileManager# "ep-sub"
                    subsc_all FALSE
                   {
                         1998892590.1.1218#TMF_CCMS::ProfileManager# "SWD
                    subsc_all FALSE
                   {
rem-ic   839    M-H    1-826        0
      Time run:    [Sun 22-Nov 12:05:45]
      Object ID:   1998892590.22.508+#TMF_Endpoint::Endpoint#
      Method:      _get_label
      Principal:   TIVTOR\Administrator@dhcp32-186." (0/0)
      Path:
rem-oc   839 DISP_UNAVAIL           0
loc-oc   826 e=12                  91
      Results: (encoded):
             "Exception:StExcep::SystemException:StExcep::COMM_FAILURE"
             {
               570425344 16777216
             }
```

Figure 55. Subscribe Failure wtrace - Part 6 of 6

In the above example, we have a two-way interconnection between dhcp32-186 and itso2. The Administrator on dhcp32-186 has attempted to subscribe a TMA endpoint on itso2 to a local profile manager.

What is not known at the time is that this endpoint is unavailable. It has, in fact, completed a login to a gateway in a 3rd TMR.

We really begin the *subscribe* operation in tid=826 and will be executing methods on `local OID:1212391543.1.1281#TMF_CCMS::ProfileManager#` to subscribe `remote OID:1998892590.22.508+#TMF_Endpoint::Endpoint#`, which is `rh2900b-ep`.

We begin by getting the label attribute for the profile manager OID and the policy region in which it is defined. Next, we get the OID of the name registry and perform further lookups to find out what validation policies are enabled for the policy region, as seen in tid 830.

In tid 832, we validate the endpoint subscriber *rh290b-ep* for the profile manager *Sentry*.

We locate our local endpoint manager in tid 833, and in tid 834, we attempt to get the endpoint key value for *rh2900b-ep* in *region 1998892590*, which is all the subscription information for this endpoint.

In tid 837, we get *OID: 1998892590.1.517#TMF_LCF::EpMgr#* as the remote endpoint manager OID and then send a remote request to this object in tid 838, as shown by the *rem-ic* to get the endpoint key values.

The output of the remote call *rem-oc* lists all the current profile manager subscriptions for this Endpoint. Notice that they are all in remote region *1998892590*.

Finally, in tid 839, we send another remote request to the endpoint to retrieve its *label* attribute value, which fails with rem-oc DISP_UNAVAIL

The initiating thread 826 now throws an exception with a COMM_FAILURE for this endpoint.

Be aware that `wep ls` does not show the connection status, so check on the local TMR in which the endpoint exists with `wep <name> status`.

In this case, we get a response of *unable to determine endpoint status; endpoint may be unreachable*.

6.3.6 HMAC Encrypted Data Error

This `wtrace` looks at an HMAC error following a distribution to an endpoint:

```
loc-ic   815 M-hdoq    1-648      750
      Time run:    [Mon 16-Nov 00:35:42]
      Object ID:   1212391543.1.517#TMF_LCF::EpMgr#
      Method:      push_copy_in
      Principal:   TIVTOR\Administrator@gblnt.dev.t^B (10104288/0)
      Path:        /w32-ix86/TAS/CCMS/profile_manager_GUI
      Input Data: (encoded):
             {
               67108864
               [
                 "push_copy_in" "lcf_endpoint" "1212391543.15.508+#TMF_Endp
                 oint::Endpoint#" "1212391543.1.1601#Sentry::All#"
               ]
             }
             {
               184549376
               [
                 "WD_DIALOG_OWNER=1212391543.15.508+#TMF_Endpoint::Endpoint#" "WD_GADGET_PATH=
                  collectiongroup.subscribers.1212391543x15x508+#TMF_Endpoint::Endpoint#" "
                  WD_SOURCE_PATH=collectiongroup.subscribers.1212391543x15x508+#TMF_Endpoint:
                 :Endpoint#" "WD_DIALOG_NAME=main" "WD_DIALOG_INSTANCE=127: 0" "WD_DESKTOP_OID
                  =1212391543.1.529#TMF_UI::Extd_Desktop#" "WD_DESKTOP_PID=173" "WD_DESKTOP_HOST
                  =gblnt" "WD_DISPLAY=gblnt:0" "WD_OCO_OID=1212391543.1.178#TMF_Administrator:
                 :Configuration_GUI#" "LANG=en"
               ]
             }
loc-ic   854 M-hdoq    1-838      206
      Time run:    [Mon 16-Nov 00:35:55]
      Object ID:   1212391543.1.662
      Method:      rpt
      Principal:   TIVTOR\Administrator@gblnt.dev.t^C (10104288/0)
      Path:        __gateway_internals_implid
      Trans Id:
             {
               1212391543:1,1212391543:1,39:1189
             }
             #4
      Input Data: (encoded):
16777216
             {
               16777216
               [
                 {
                   "1212391543.15.508+#TMF_Endpoint::Endpoint#"
                   {
                     0
                   }
                 }
               ]
             }
"engineUpdate"
             {
               "null" 0 false
             }
              0
             {
               637534208 "0x42 0x63 0x32 0x34 0x35 0x34 0x39 0x38 0x36 0x39 0x34 0x37 0x49
               0x34 0x30 0x36 0x32 0x49 0x31 0x34 0x64 0x38 0x39 0x34 0x38 0x31 0x34 0x64
               0x38 0x39 0x34 0x38 0x20 0x67 0x62 0x6c 0x6e 0x74 "
             }
loc-oc   854                      448
      Results: (encoded):
             {
               16777216
{
                   "1212391543.15.508+#TMF_Endpoint::Endpoint#"
}
```

Figure 56. HMAC Error in wtrace - Part 1 of 2

```
                        12
{
                     "ExErrorMsg" 251658240 false
                    {
                       "Exception:UserException:SysAdminException::ExException:ExErrorMsg
                         " "NULL" 0 "%7$M" -1111929034
                        {
                          16777216
                            {
                              "NULL" "decrypt_data: HMAC does not match encrypted data!" -1
                              {
                                "TMF_Types::_sequence_string_StringList"
                                318767104 false
                                {
loc-oc   815 e=12               1.2K
      Results: (encoded):
             "Exception:UserException:SysAdminException::ExException:SysAdminException::ExFailed"
             {
               "Exception:UserException:SysAdminException::ExException:
               SysAdminException::ExFailed" "CCMS_catalog" -620756992 "Distribute failed for
               some subscribers:" -1011265738
               {
                 33554432
                    {
                      "SysAdminException::ExFailed" 251658240 false
                      {
                        "Exception:UserException:SysAdminException::ExException:SysAdminException
                         ::ExFailed" "CCMS_catalog" 738263040 "Distribute of profile '%7$s' failed
                         for some subscribers " -1011265738
                        {
                        "NT_Disk" }
{
                      "SysAdminException::ExFailed" 251658240 false
                      {
                        "Exception:UserException:SysAdminException::ExException:SysAdminException
                         ::ExFailed" "CCMS_catalog" -637534208 "Distribute failed for subscriber
                         '%7$s': " -1111929034
                        {
                          16777216
                          [
                            {
                              "ExErrorMsg" 251658240 false
                              {
                                "Exception:UserException:SysAdminException::ExException:
                                 ExErrorMsg" "NULL" 0 "%7$M" -1111929034
                                {
                                  0
                                }
{
                                    16777216
                                    [
                                      {
                                        "NULL" "decrypt_data: HMAC does not match encrypted data!" -1
                                        {
                                          "TMF_Types::_sequence_string_StringList" 318767104 false
                                          {
                                            0
                                          }
]
                        }
                        "gblnt-epbad"
                      }
                    }
                 ]
               }
               "NULL"
             }
```

Figure 57. HMAC Error in wtrace - Part 2 of 2

The problem, in this case, was that the endpoint had been reinstalled, but the old subscriber had not been removed from the profile manager, and the old endpoint had not been removed with `wdelep` followed by `wchkdb -u`. In this case, any distribution to the old object will fail with the *HMAC* error. This means that the gateway/repeater information for this endpoint is not correct. Verify all endpoints (they must be on-line) with `wep <EP_name> status` and delete those with `wdelep` that are redundant.

6.3.7 Damaged Database odstat and wtrace Example

> **Note**
>
> There have been numerous notes throughout this redbook telling you not to set attribute values with low-level commands. This is an example of how the database can be damaged and the laborious procedure involved to rectify the situation, if at all possible; in some cases, recovery may not be possible at all. Use `wtrace` and 'w' commands, but low-level commands should only be used under direction from Tivoli support. Always backup your database.

In the remainder of this section, we try to figure out a way of resolving a far more complex problem and one in which the `odstat` and `wtrace` does not provide any errors of the type e=xx, as in the previous examples.

In this sample, an Administrator has attempted to rename a policy region using low-level commands as a test to see if he could do so for all the policy regions through a shell script, that is, changing the *label* of the OID.

The first run of `wchkdb` throws out an exception for every object within the policy region, as shown in Figure 58:

```
wchkdb: Checking object database:
.....................................................................
wchkdb: Processing "dev01_tasks" (1212391543.1.2371#TMF_Task::TaskLibrary#):
The cached policy region name "tivdev01-region" does not match the name of region
1212391543.1.2369#TMF_PolicyRegion::GUI#.
.........................................
wchkdb: Processing "Sentry_dev_NT" (1212391543.1.2372#TMF_CCMS::ProfileManager#):
The cached policy region name "tivdev01-region" does not match the name of region
1212391543.1.2369#TMF_PolicyRegion::GUI#.
.....................................................................
```

Figure 58. Policy Region Label Change wchkdb Errors

The Administrator issued the command `idlattr -ts 1212391543.1.2369 label Object tivprod01-region`, and now the database is in a mess. This OID is that of a policy region, and it was originally labelled `tivdev01-region`.

Our task is to try and fix this database problem.

We begin with tracing the `wchkdb`, but all we can see is that the lookup on the name registry does show the new label/name for the policy region, and that the *check_db* methods for the task library and Sentry policy region return the same error messages seen above during the `wchkdb`. (We will show this shortly). So, next we list the contents of */Library/PolicyRegion* and compare this output with `wlookup -ar PolicyRegion` to see if we can find any obvious errors that we can fix. We do, in fact, find some discrepancies in these object directories, but even after repairing what we see, the `wchkdb` still fails as before.

We will now have to dig a little deeper to see which attributes would have to be updated to correctly reflect the cross-referenced names as expected.

```
loc-ic  7079  M-hdq   1-7011        0
      Time run:    [Thu 26-Nov 18:41:52]
      Object ID:   1212391543.1.2369#TMF_PolicyRegion::GUI#
      Method:      get_type_name
      Principal:   TIVTOR\Administrator@localhost (10104384/0)
      Path:        /w32-ix86/TMF/BASESVCS/Policy_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,82:6024
             },
             {
               1212391543:1,1212391543:1,82:6025
             },
             {
               1212391543:1,1212391543:1,82:6093
             }
             #3
loc-is  7079 getattr        0       class_objid
      Time run:    [Thu 26-Nov 18:41:52]
      Object ID:   1212391543.1.2369#TMF_PolicyRegion::GUI#
Method:      get_type_name
loc-os  7079 getattr        58
      Time run:    [Thu 26-Nov 18:41:52]
      Object ID:   1212391543.1.2369#TMF_PolicyRegion::GUI#
      Method:      get_type_name
      Results: (binary) (dump suppressed)
loc-ic  7080    M-H   1-7079        0
      Time run:    [Thu 26-Nov 18:41:52]
      Object ID:   1212391543.1.184#TMF_SysAdmin::InstanceManager#
      Method:      _get_label
      Method Args: label
      Principal:   TIVTOR\Administrator@localhost (0/0)
      Path:        i_getattr
      Trans Id:
             {
               1212391543:1,1212391543:1,82:6024
             },
             {
1212391543:1,1212391543:1,82:6025
             },
             {
               1212391543:1,1212391543:1,82:6093
             },
             {
               1212391543:1,121239
loc-oc  7080                        29
      Results: (encoded):
             "PolicyRegion"
loc-oc  7079                        29
      Results: (encoded):
             "PolicyRegion"
loc-ic  7081 M-hdoq   1-7011       41
      Time run:    [Thu 26-Nov 18:41:52]
      Object ID:   1212391543.1.26
      Method:      local_lookup
      Principal:   TIVTOR\Administrator@localhost (10104384/0)
      Path:        /w32-ix86/TMF/BASESVCS/TNR_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,82:6024
             },
             {
               1212391543:1,1212391543:1,82:6025
             },
             {
               1212391543:1,1212391543:1,82:6095
             }
             #3
      Input Data: (encoded):
             "PolicyRegion"  "tivprod01-region"
```

Figure 59. Policy Region Label Change wtrace - Part 1 of 2

```
loc-oc  7081                    111
      Results: (encoded):
             {
               "1212391543.1.2369#TMF_PolicyRegion::GUI#" "tivprod01-region"
               {
                 "null" 0 false
               }
             }

loc-ic  7146  M-hdq   Extern      12
      Time run:    [Thu 26-Nov 18:41:53]
      Object ID:   1212391543.1.2371#TMF_Task::TaskLibrary#
      Method:      check_db
      Principal:   TIVTOR\Administrator@localhost (10104384/0)
      Path:        /w32-ix86/TMF/BASESVCS/NameRegBase_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,82:6159
             },
             {
               1212391543:1,1212391543:1,82:6160
             }
             #3
      Input Data: (encoded):
             true  0  true
loc-ic  7147  M-hdq   1-7146      12
      Time run:    [Thu 26-Nov 18:41:53]
      Object ID:   1212391543.1.2371#TMF_Task::TaskLibrary#
      Method:      check_db
      Principal:   TIVTOR\Administrator@localhost (10104384/0)
      Path:        /w32-ix86/TMF/BASESVCS/Collection_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,82:6159
             },
             {
               1212391543:1,1212391543:1,82:6160
             },
             {
               1212391543:1,1212391543:1,82:6161
             }
             #3
      Input Data: (encoded):
             true  0  true

loc-oc  7147                    627
      Results: (encoded):
             {
               16777216
               [
                 {
"Exception:UserException:SysAdminException::ExException:TMF_Application::DBCheckException:TMF_SysAdmin_DBCheck:
:RegionDBCheck::ExBadPRName"
                 {
"TMF_SysAdmin_DBCheck::RegionDBCheck::ExBadPRName"
                  251658240 false
                  {
"Exception:UserException:SysAdminException::ExException:TMF_Application::DBCheckException:TMF_SysAdmin_DBChe
ck::RegionDBCheck::ExBadPRName" "checkdb_errors" 268435456 "The cached policy region name "%8$s" does not match the
name of region %7$s." 1105681718
                   {
                     0
                   }
"1212391543.1.2369#TMF_PolicyRegion::GUI#" "tivdev01-region"
                  }
```

Figure 60. Policy Region Label Change wtrace - Part 2 of 2

As stated before, the `wtrace` does not help us in any way. We will now show the discrepancies in `/Library/PolicyRegion` and `wlookup -ar PolicyRegion` and then fix them.

```
G:\tmp>wlookup -ar PolicyRegion
ACPdefault      1212391543.1.604#TMF_PolicyRegion::GUI#
TEC36Region     1212391543.1.572#TMF_PolicyRegion::GUI#
TME 10 Security 1212391543.1.1253#TMF_PolicyRegion::GUI#
Test1   1212391543.1.1286#TMF_PolicyRegion::GUI#
Test2   1212391543.1.1288#TMF_PolicyRegion::GUI#
Tivoli/Sentry Defaults-gblnt-region     1212391543.1.744#TMF_PolicyRegion::GUI#
TivoliDefaultPhoneRegion        1212391543.1.866#TMF_PolicyRegion::GUI#
TivoliDefaultSecurityPolicyRegion       1212391543.1.1244#TMF_PolicyRegion::GUI#
tivprod01-region 1212391543.1.2369#TMF_PolicyRegion::GUI#
gblnt-region    1212391543.1.195#TMF_PolicyRegion::GUI#
tivdev01-region 1212391543.1.2369#TMF_PolicyRegion::GUI#

G:\tmp>wls -l /Library/PolicyRegion
1212391543.1.195#TMF_PolicyRegion::GUI# gblnt-region
1212391543.1.572#TMF_PolicyRegion::GUI# TEC36Region
1212391543.1.604#TMF_PolicyRegion::GUI# ACPdefault
1212391543.1.744#TMF_PolicyRegion::GUI# Tivoli/Sentry Defaults-gblnt-region
1212391543.1.866#TMF_PolicyRegion::GUI# TivoliDefaultPhoneRegion
1212391543.1.1244#TMF_PolicyRegion::GUI#        TivoliDefaultSecurityPolicyRegion
1212391543.1.1253#TMF_PolicyRegion::GUI#        TME 10 Security
1212391543.1.1286#TMF_PolicyRegion::GUI#        Test1
1212391543.1.1288#TMF_PolicyRegion::GUI#        Test2
1212391543.1.2369#TMF_PolicyRegion::GUI#        tivprod01-region
```

Figure 61. Policy Region Label Change wls and wlookup Output

As seen in Figure 61, the lookup of all policy regions shows two instances of the same OID but with unique labels. The OID is `1212391543.1.2369`.

After unregistering `tivdev01-region` with the name registry, the `wlookup` output is as expected:

```
G:\tmp>wregister -ur PolicyRegion tivdev01-region

G:\tmp>wlookup -ar PolicyRegion
ACPdefault      1212391543.1.604#TMF_PolicyRegion::GUI#
TEC36Region     1212391543.1.572#TMF_PolicyRegion::GUI#
TME 10 Security 1212391543.1.1253#TMF_PolicyRegion::GUI#
Test1   1212391543.1.1286#TMF_PolicyRegion::GUI#
Test2   1212391543.1.1288#TMF_PolicyRegion::GUI#
Tivoli/Sentry Defaults-gblnt-region     1212391543.1.744#TMF_PolicyRegion::GUI#
TivoliDefaultPhoneRegion        1212391543.1.866#TMF_PolicyRegion::GUI#
TivoliDefaultSecurityPolicyRegion       1212391543.1.1244#TMF_PolicyRegion::GUI#
tivprod1-region 1212391543.1.2369#TMF_PolicyRegion::GUI#
gblnt-region    1212391543.1.195#TMF_PolicyRegion::GUI#
```

Figure 62. Policy Region Label Change Name Registry Correction

Unfortunately, `wchkdb` still fails as before.

We now have to think of what attributes for the task library and the Sentry profile manager will reference the policy region. In Figure 63, we use `objcall OID contents` to list the attributes of the task library object:

```
bash$ objcall 1212391543.1.2372 contents
ATTRIBUTE:_BOA_id
ATTRIBUTE:class_objid
ATTRIBUTE:collections
ATTRIBUTE:databases
ATTRIBUTE:flags
ATTRIBUTE:label
ATTRIBUTE:last_failed
ATTRIBUTE:members
ATTRIBUTE:pres_object
ATTRIBUTE:pro
ATTRIBUTE:pro_name
ATTRIBUTE:profile_push_order
ATTRIBUTE:push_trans_commit_behavior
ATTRIBUTE:resource_host
ATTRIBUTE:skeleton
ATTRIBUTE:sort_name
ATTRIBUTE:state
ATTRIBUTE:subscribers
ATTRIBUTE:subscriptions
```

Figure 63. Attributes of Task Library

In Figure 64,we check the pro_name attribute with an `objcall` and correct it using `idlattr`:

```
G:\tmp>objcall 1212391543.1.2372 getattr pro_name
d    tivdev01-region


idlattr -ts 1212391543.1.2372 pro_name Object tivprod01-region
```

Figure 64. Correcting Profile Manager Name for Task Library Object

Updating this attribute value did not resolve the problem either. Let us attempt to return the policy region label to the original from the Tivoli desktop. This too fails due to an object of type Presentation already in existence with the same old label of `tivdev01-region`.

We can see this failure in the following `wtrace` of the operation to change the policy region label from the desktop:

```
loc-is 14178 getattr        0       pres_object
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2369#TMF_PolicyRegion::GUI#
      Method:      _set_label
loc-os 14178 getattr        50
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2369#TMF_PolicyRegion::GUI#
      Method:      _set_label
      Results: (binary) (dump suppressed)
loc-ic 14179  M-hdq  1-14178       26
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2370#TMF_UI::Presentation#
      Method:      _set_label
      Principal:   TIVTOR\Administrator@localhost (10104288/0)
      Path:        /w32-ix86/TMF/BASESVCS/NameRegBase_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,83:12118
},
             {
               1212391543:1,1212391543:1,83:12119
             }
             #1
      Input Data: (encoded):
             "tivdev01-region"
loc-is 14179 getattr        0       label
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2370#TMF_UI::Presentation#
      Method:      _set_label
loc-ic 14180  M-hdq  1-14179       26
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2370#TMF_UI::Presentation#
      Method:      _set_label
      Principal:   TIVTOR\Administrator@localhost (10104288/0)
      Path:        /w32-ix86/TMF/BASESVCS/Policy_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,83:12118
             },
             {
               1212391543:1,1212391543:1,83:12119
             },
             {
               1212391543:1,1212391543:1,83:12120
             }
             #3
      Input Data: (encoded):
             "tivdev01-region"
loc-is 14180 setattr        26      label
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2370#TMF_UI::Presentation#
      Method:      _set_label
      Input Data: (encoded):
             "tivdev01-region"
```

Figure 65. Desktop Policy Region Label Change Failure wtrace - Part 1 of 2

```
loc-os 14180 setattr        0
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.2370#TMF_UI::Presentation#
      Method:      _set_label
loc-ic 14182  M-hdq  1-14180       76
      Time run:    [Thu 26-Nov 20:55:11]
      Object ID:   1212391543.1.132#TMF_TNR::InstanceManager#
      Method:      update_label
      Principal:   TIVTOR\Administrator@localhost (10104288/0)
      Path:        /w32-ix86/TMF/BASESVCS/Collection_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,83:12118
             },
             {
               1212391543:1,121
      Input Data: (encoded):
                 "1212391543.1.2370#TMF_UI::Presentation#"  "tivdev01-region"
loc-ic 14185 M-hdoq  1-14179       75
      Time run:    [Thu 26-Nov 20:55:12]
      Object ID:   1212391543.1.26
      Method:      change_label
      Principal:   TIVTOR\Administrator@localhost (10104288/0)
      Path:        /w32-ix86/TMF/BASESVCS/TNR_prog1
      Trans Id:
             {
               1212391543:1,1212391543:1,83:12118
             },
             #3
      Input Data: (encoded):
             "Presentation"  "tivdev01-region"  "tivdev01-region"
loc-oc 14185 e=12                509
      Results: (encoded):
             "Exception:UserException:SysAdminException::ExException:SysAdminE
             xception::ExInvalid:SysAdminException::ExExists:SysAdminException:
             :ExEntryExists"
             {
               "Exception:UserException:SysAdminException::ExException:
               SysAdminException::ExInvalid:SysAdminException::ExExists:
               SysAdminException::ExEntryExists" "TNR_errors" 100663296
               "A resource instance of type "%8$s" named "%7$s" already exists."
               -2147066314
               {
                 0
               }
               "tivdev01-region#1212391543" "Presentation"
```

Figure 66. Desktop Policy Region Label Change Failure wtrace - Part 2 of 2

What we have discovered is that there is an OID `1212391543.1.2370` that has the policy region label. In Figure 67 on page 171, we will try to set this to `tivprod01-region`.

```
idlattr -tg 1212391543.1.2370 label string
"tivdev01-region"

idlattr -ts 1212391543.1.2370 label Object tivprod01-region

G:\Tivoli\db\gblnt.db>wchkdb -u

wchkdb: Preparing object lists:
wchkdb: Checking object database:
.....................................................................
wchkdb: Processing "tivprod01-region" (1212391543.1.2370#TMF_UI::Presentation#):
    The object named "tivprod01-region" of type "Presentation" does not have an entry in
the name registry when it should.
```

Figure 67. Correcting the Label of the Presentation Object

The final check of the database above ultimately corrects the last problems by updating the name registry entry, and a subsequent `wchkdb` runs clear.

We have, eventually, fully recovered!

6.4 Log Files in the Database Directory

The DBDIR environment variable points to the Tivoli management database directory. This directory contains a number of potentially useful log files listed in Figure 68 (from a UNIX system):

```
-rw-rw-rw-   1 0       0           83077 Nov 16 20:50 epmgrlog
-rw-rw-rw-   1 0       0            6888 Nov 16 20:50 gatelog
-rw-rw-rw-   1 0       0               0 Nov 16 22:14 gwdb.log
-rw-rw-rw-   1 0       0               0 Nov 16 20:46 notice.log
-rw-rw-rw-   1 0       0               0 Nov 16 22:14 odb.log
-rw-rw-rw-   1 0       0         1048576 Nov 16 00:35 odtrace.log
-rw-rw-rw-   1 0       0           33013 Nov 17 13:25 oservlog
```

Figure 68. Log Files in $DBDIR

The `odb.log`, `notice.log`, and `gwdb.log` are transaction files, explained in 6.4.1, “Transaction Log Files and tmstat” on page 172.

The `odtrace.log` can not be read by normal editors, it is parsed by the `wtrace` command (see 6.3, “The wtrace Command” on page 140) and is a log of oserv errors, object method invocations, and ALI services.

The `epmgrlog` and `gatelog` files are new from Version 3.2. They are text files that record status information about the endpoint manager and gateway as well as transactions between the endpoint manager, gateway, and endpoint.

These files are covered in sections 6.4.3, “The epmgrlog File” on page 178 and 6.4.4, “The gatelog File” on page 178.

The oservlog captures all error and information messages from the oserv daemon and is a text file explained in 6.4.2, “The oservlog File” on page 175.

For a TMA endpoint, the /Tivoli/lcf/dat directory will contain a log file called lcdf.log. This is described in 6.5, “Endpoint lcfd.log File” on page 181.

6.4.1 Transaction Log Files and tmstat

The odb.log, notice.log and gwdb.log are used by the oserv, notices, and gateway daemons, respectively. They should never be erased because they could contain transaction data. The odb.log, notice.log, and gwdb.log are used by Tivoli to allow roll-back of database transactions on odb.bdb, notice.bdb, and gwdb.bdb database files in the event of a transaction failure or abort. They may increase in size while transactions are still pending, but once transactions are completed (committed), they should return to zero.

You can use `tmstat` to check for transaction locks. If they do not return to a zero file size, then that could be an indicator of a problem.

An `odadmin db_sync` will flush the object database, which should zero the log file. This synchronization process occurs naturally in the oserv every three minutes. Do NOT delete any of these log files.

The `logls` command can be used to output a readable version of a transaction log file. Working with transactions is not a common problem determination technique; so, the output will not be meaningful to you unless you are already familiar with Tivoli transactions.

Note

Modifying Tivoli transactions should NOT be used as a routine method of problem determination. Altering transaction data can have serious and irreversible affects on the operation of a TMR. We advise you to use this information to find out more about what is going on in the TMR and use other methods to attempt to rectify problems.

6.4.1.1 The tmstat Command

The `tmstat` command provides a way to interact with Tivoli transactions. As already stated, anything that could potentially modify transactions should be used with caution. Unless directed by Tivoli Support, you are only likely to use `tmstat` to look at the current status of transactions.

```
Transactions for 0.0.0

   A          B      C      D        E      F      G      H
 Trid        Type   State  Resources Polling Coord   Parent   MTid
 ------------------------------------------------------------------------
{1212391543:1,1212391543:1,100:310}
             Top-T   running   No      No    running running   642
{1212391543:1,1212391543:1,100:310},{1212391543:1,1212391543:1,100:311}
             Revoke-Trunning   No      No    running running   643
{1212391543:1,1212391543:1,100:310},{1212391543:1,1212391543:1,100:311},
{1212391543:1,1212391543:1,100:325}
             Revoke-Tcommit    Yes     Yes   running running   663
```

Figure 69. Sample of tmstat Output - Part 1 of 2

Figure 69 shows the first part of a `tmstat` output obtained by running `tmstat -av`. The information presented is as follows:

A Transaction ID (contains three parts):
- Home region and dispatcher number
- Originating region and dispatcher number
- Transaction thread ID (two part colon-separated number)

B Type of transaction

C State of transaction

D Some specific Resources are allocated to the transaction - `Yes/No`

E Polling indicates if transaction is waiting or active

F Indicates if Coordinator Manager is active

G Indicates if parent process is active

H Indicates thread assigned to process

```
n_active = 15  n_free = 185
   tid type    ptid State  StdO  StdE    Start   Err    Method
   633 O+hdoq   1-77   run     0     0 11:40:00
     1212391543.1.2307#TMF_PcManagedNode::Pc_Managed_Node# push_copy_in
   642 O+hdoq  1-633   run     0     0 11:40:04
     1212391543.1.1409#TMF_CCMS::ProfileManager# default_push_profiles
   643 O+hdoq  1-642   run     0     0 11:40:05
     1212391543.1.1410#FilePackage::FpoCore# default_push
   673 O+ho   1-643   run     0     0 11:40:11
     1212391543.1.347#TMF_ManagedNode::Managed_Node# fp_dist
   693 O+ahdoq  1-673   run     0     0 11:40:19
     1212391543.1.2307#TMF_PcManagedNode::Pc_Managed_Node# fps_install


               Locks for 0.0.0
Object         Name
-----------------------------------------------------------------
1212391543.1.26 NameRegistry!PcManagedNode  <------------- A
                              ----Locker--------------State----Mode-----
     {1212391543:1,1212391543:1,100:310},{1212391543:1,1212391543:1,100:311}
                                                      held    read
1212391543.1.26 NameRegistry!EndpointManager
                              ----Locker--------------State----Mode-----
     {1212391543:1,1212391543:1,100:310},{1212391543:1,1212391543:1,100:311},
     {1212391543:1,1212391543:1,100:330}
                       ^                              held    read
                       |                               ^       ^
                       |                               |       |
                       B                               C       D
```

Figure 70. Sample of tmstat Output - Part 2 of 2

The information in the second part of the `tmstat` output, shown in Figure 70, appears when locks are present and is as follows:

A Identifies the Object ID and name of a locked resource

B Identifies the Transaction ID of all transactions in the hierarchy that own the lock

C State of transaction - `held` or `released`

D Mode of the lock - `read` or `write`

6.4.1.2 Troubleshooting Process Locks

The best approach is to find the Top level transaction that holds the lock of the resource that is causing the process to lock with `tmstat`, then kill the corresponding process for this thread ID, as found in `odstat`. If it is not possible to kill the process, then you could use `tmcmd` to abort this Top level transaction ID (see below). The abort should cause the locks to be released and, thus, allow the blocked process to run normally. By aborting the Top level transaction, all temporarily held data changes are rolled back. This should result in a consistent state for any of the objects involved in that transaction.

- To prevent false reporting of data and errors, processes that lock should only be terminated from the Top transaction.

  If only the offending Sub transaction is terminated, only portions of the transaction may be rolled back causing potential future problems. Therefore, only kill Top level transactions if possible.

Important

We do not recommend that you manually kill transactions. This example is provided to show you the process if you are directed to do this by Tivoli Support or if you have a test TMR that you do not mind trashing.

In the above examples, if we had a problem, and found we could not kill the processes for thread IDs 633 or 642 (push_copy_in and default_push_profiles), we could abort the transaction with the command:

```
tmcmd abort {1212391543:1,1212391543:1,100:310}
```

To determine if you have a process lock problem, you can monitor the odb.log file (found in $DBDIR for Unix and %DBDIR% for NT). See Chapter 6, “Commands and Logs for Troubleshooting” on page 131 for more information about odb.log. If this file stays at a constant size, or more importantly, consistently grows in size for no obvious reason, then you may have a lock problem.

6.4.2 The oservlog File

This can be a very helpful file, providing information about the oserv daemon, such as when it started, shutdowns, errors, and information messages.

The following is an explanation of some of the more common errors you may see in the oservlog:

6.4.2.1 Hdaemon Exit while in Use

This error message is followed by a hexadecimal exit code that corresponds to an exit code in `odstat`. Usually, this is used when the oserv has crashed or is not running. Sometimes, it can be the scheduler daemon.

The more common error codes are: `a`, `b`, `1`, `5`, `9`. These translate to `s=10`, `s=11`, `e=1`, `e=5`, and `e=9`, respectively.

6.4.2.2 Cannot Map Address

This error message displays when the oserv receives a request from an IP address that is not registered in `odadmin odlist`. It gives the IP address in

hexadecimal. The address can be translated by taking every two digits and translating them from hexadecimal to decimal. This message is common with multi-interface systems.

When you find this error you have to:

1. Convert the address to dotted decimal format.

   Note

   Convert the address two hex digits at a time from **right to left**. This is because, if the address begins with zero, the leading zero will be dropped from the hexadecimal number displayed.

   For example: `x'B35487B'` is `11.53.72.123`

2. Determine the correct dispatcher number for this address.
3. Add the new address in the Tivoli Database using `odadmin odlist add_ip_alias`.

An example of handling this error is as follows:

```
# more oservlog
Oct 27 16:58:01:^/usr/local/Tivoli/bin/aix4-r1/TAS/SCHEDULER/TMF_sched killed by
signal 15
Oct 27 16:59:31: $Exiting gracefully, saving id's...
Oct 27 16:59:31: $reexec: started
Oct 27 16:59:36: TME 10 Framework (tmpbuild) #1 Thu Oct 3 08:08:58 CDT 1996
Copyright Tivoli Systems, an IBM Company, 1996.  All Rights Reserved.
TMR 1264987995. ORB 1. TMR server local:94. Port 94.
Oct 27 14:58:31: @Cannot map address 635a2070 port 94 to an odnum

# bc
ibase=16
70
112
20
32
5A
90
63
99
quit

# odadmin odlist add_ip_alias 2 99.90.32.112
# odadmin odlist
Region      Disp  Flags  Port           IPaddr   Hostname(s)
1264987995     1    ct-    94         9.3.1.234   rh0255b.itsc.austin.ibm.com
             2    ct-    94         9.3.1.233   rh0255a.itsc.austin.ibm.com
                                    99.90.32.112  99.90.32.112
1562489759     1    ct-    94         9.3.1.235   rh0255c.itsc.austin.ibm.com
# odadmin odlist add_hostname_alias 2 99.90.32.112 newalias
#odadmin odlist
Region      Disp  Flags  Port           IPaddr   Hostname(s)
1264987995     1    ct-    94         9.3.1.234   rh0255b.itsc.austin.ibm.com
             2    ct-    94         9.3.1.233   rh0255a.itsc.austin.ibm.com
                                    99.90.32.112  99.90.32.112 newalias
1562489759     1    ct-    94         9.3.1.235   rh0255c.itsc.austin.ibm.com
# odadmin odlist delete_ip_alias 2 99.90.32.112 99.90.32.112
# odadmin odlist
Region      Disp  Flags  Port           IPaddr   Hostname(s)
1264987995     1    ct-    94         9.3.1.234   rh0255b.itsc.austin.ibm.com
             2    ct-    94         9.3.1.233   rh0255a.itsc.austin.ibm.com
                                   99.90.32.112   newalias
1562489759     1    ct-    94         9.3.1.235   rh0255c.itsc.austin.ibm.com
```

Figure 71. Typical oservlog Output

6.4.2.3 ipc_accept Failed: IPC Shutdown

This message generally appears if the oserv is shut down while the desktop is up. Some inter-process communication (IPC) function died. The `wtrace` output will show the time of failure.

This message states that the oserv is unable to start. In this example, `destination dispatcher unavailable` means either the client is not able to

contact the server, or the server does not recognize the client. This problem usually occurs with multiple network interfaces.

6.4.2.4 @od_init: Unable to Establish Connection to ALI

The client cannot contact the TMR server. The client cannot resolve the name it has for the server.

6.4.2.5 !oserv:odlist init Failed. IPC Shutdown

An inter-process communication (IPC) function died unexpectedly.

6.4.2.6 !oserv:odlist init Failed. System Call Failed

Some system function failed. Sometimes, you may be able to find a system-generated message or message number in a log.

6.4.3 The epmgrlog File

This text file will only exist on the TMR server as it is an endpoint manager.

Figure 72 is an example of epmgrlog showing an endpoint login:

```
1998/11/16 22:04:10 +05: * * * booting
1998/11/16 22:04:18 +05: opening 1212391543.1.662
1998/11/16 22:04:18 +05: found 4 endpoints from itso1_gw
1998/11/16 22:04:22 +05: endpoint prototype 508
1998/11/16 22:04:22 +05: * * * boot complete
1998/11/16 22:04:25 +05: iom query 1212391543.1.662
1998/11/16 22:04:32 +05: get_endpoints: Requesting search for 1212391543.18.508+
#TMF_Endpoint::Endpoint#
1998/11/16 22:04:41 +05: get_endpoints: Requesting search for 1212391543.18.508+
#TMF_Endpoint::Endpoint#
1998/11/16 22:09:42 +05: dispatcher 18 logging in with code 2
1998/11/16 22:09:42 +05: 1212391543.18.508+ assigned to 1212391543.1.662#TMF_Gateway::Gateway#
1998/11/16 22:09:43 +05: - itso4-ep 1212391543.18.508+#TMF_Endpoint::Endpoin
t# 1212391543.1.662#TMF_Gateway::Gateway#
1998/11/16 22:09:43 +05: + itso4-ep 1212391543.18.508+#TMF_Endpoint::Endpoin
t# 1212391543.1.662#TMF_Gateway::Gateway#
1998/11/16 22:09:43 +05: writing epmgr.bdb/1212391543.1.662.bdb for 18
1998/11/16 22:09:43 +05: updating ali map
```

Figure 72. Typical epmgrlog Output - Endpoint Login

6.4.4 The gatelog File

This flat text file will exist on any managed node that is defined as a gateway (including the TMR server). The amount of information logged in this file depends on the logging level set.

Some examples of gatelog are shown in the following figures, Figure 73 on page 179 and Figure 74 on page 180:

```
1998/11/06 11:09:45 +06: gateway boot: started booting.
1998/11/06 11:09:45 +06: gateway boot: debug level is 9.
1998/11/06 11:09:46 +06: gateway boot: endpoint manager is 1351550138.1.517#TMF_
LCF::EpMgr#.
1998/11/06 11:09:46 +06: gateway boot: endpoint prototype is 508.
1998/11/06 11:09:46 +06: gateway boot: impl root is c:/Tivoli/bin/lcf_bundle.
1998/11/06 11:09:46 +06: gateway boot: port is 9494.
1998/11/06 11:09:46 +06: found network interface: 146.84.32.173
1998/11/06 11:09:47 +06: gateway boot: received ep cache from epmgr containing 5
 items.
1998/11/06 11:09:47 +06: tcp server: waiting for connection on 0.0.0.0+9494...
1998/11/06 11:09:47 +06: udp server: waiting for connection on 0.0.0.0+9494...
1998/11/06 11:09:47 +06: starting dbcheck
1998/11/06 11:09:47 +06: gateway boot: gateway boot completed successfully.
1998/11/06 11:09:47 +06: dbcheck: cache is clean
1998/11/06 11:09:47 +06: dbcheck: finished
1998/11/06 11:11:23 +06: dgram in: 514 bytes
1998/11/06 11:11:23 +06: udp server: waiting for connection on 0.0.0.0+9494...
1998/11/06 11:11:23 +06: sched: got a job
```

Figure 73. Typical Gatelog with Default Debug Level of 0

```
1998/11/20 11:06:02 +05: tcp server: waiting for connection on 0.0.0.0+9494...
1998/11/20 11:06:02 +05: mdist: Registering Repeater Manager: 1212391543.1.365
1998/11/20 11:06:02 +05: reader_thread: received data: session=2f, type=16, len=218
1998/11/20 11:06:02 +05: new_session: 20f30a1f, connecting to 146.84.32.186+1450...
1998/11/20 11:06:03 +05: upcall start: from=146.84.32.186+1450, class=SentryGateway,
method=ChangeIcon
1998/11/20 11:06:03 +05: mdist: TMF_rptm_mgr::rpt_register called, tuning parms:
1998/11/20 11:06:03 +05: mdist: mem_max = 10000
1998/11/20 11:06:03 +05: mdist: disk_max = 50000
1998/11/20 11:06:03 +05: mdist: disk_hiwat = 50000
1998/11/20 11:06:03 +05: mdist: disk_time = 1
1998/11/20 11:06:03 +05: mdist: disk_dir = g:/Tivoli/db/gblnt.db/tmp/
1998/11/20 11:06:03 +05: mdist: net_load = 500
1998/11/20 11:06:03 +05: mdist: max_conn = 100
1998/11/20 11:06:03 +05: mdist: stat_intv = 180
1998/11/20 11:06:03 +05: mdist: Opening cache file: g:\Tivoli\db\gblnt.db\tmp\pmap2
1998/11/20 11:06:03 +05: mdist: in_spool_thread started: TID = 1386e58
1998/11/20 11:06:04 +05: mdist: out_spool_thread started: tid = 1386ed8 client =
[1212391543.47.508+#TMF_Endpoint::Endpoint#]
1998/11/20 11:06:04 +05: mdist: in_spool_thread finished: TID = 1386e58
1998/11/20 11:06:04 +05: downcall: Method body /bin/w32-ix86/TME/SENTRY/dogEndpoint found.
1998/11/20 11:06:04 +05: downcall: dependency /lib/w32-ix86/libccms_lcf.dll found.
1998/11/20 11:06:04 +05: downcall: dependency /sentry/w32-ix86/wntmon.exe found.
1998/11/20 11:06:04 +05: downcall: dependency /sentry/w32-ix86/wntmon.dll found.
1998/11/20 11:06:04 +05: downcall: dependency /sentry/w32-ix86/wntevlog.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/ntprocinfo.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/ntfsinfo.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/sh.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/bash.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/awk.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/perl.exe found.
1998/11/20 11:06:05 +05: downcall: dependency /bin/w32-ix86/tools/touch.exe found.
.
.
1998/11/20 11:06:05 +05: downcall: dependency /msg_cat/C/SentEng.cat found.
1998/11/20 11:06:05 +05: downcall: dependency /msg_cat/fr_FR/SentEng.cat found.
1998/11/20 11:06:05 +05: downcall: dependency /msg_cat/ja_JP/SentEng.cat found.
1998/11/20 11:06:06 +05: downcall: dependency /msg_cat/pt_BR/SentEng.cat found.
1998/11/20 11:06:06 +05: downcall: dependency /sentry/generic/SNMP/Compaq.OID found.
1998/11/20 11:06:06 +05: downcall: dependency /sentry/generic/SNMP/rfc1213.OID found.
1998/11/20 11:06:06 +05: destroying session 20f30a1f
1998/11/20 11:06:06 +05: idmap: user ($root_user,w32-ix86) -> Administrator
1998/11/20 11:06:06 +05: new_session: 20f30a20, connecting to 146.84.32.186+1322...
1998/11/20 11:06:11 +05: reader_thread: received data: session=20f30a20, type=9, len=52
1998/11/20 11:06:11 +05: reader_thread: received data: session=20f30a20, type=5, len=72
1998/11/20 11:06:11 +05: destroying session 20f30a20
1998/11/20 11:06:11 +05: mdist: Finished out_spool to 1212391543.47.508+#TMF_Endpoint::Endpoint#
1998/11/20 11:06:11 +05: mdist: Result length for 1212391543.47.508+#TMF_Endpoint::Endpoint# = 0
1998/11/20 11:06:11 +05: mdist: out_spool_thread finished: tid = 1386ed8 client =
[1212391543.47.508+#TMF_Endpoint::Endpoint#]
1998/11/20 11:06:11 +05: reconnect_thread: connection from 146.84.32.186+1462
```

Figure 74. Sentry Profile Distribution Gatelog - Debug Level 6

With debug level 6, all endpoint login attempts, as well as upload and download calls to or from the endpoint, will be written to the log. Refer to the `wgateway` command for more information on setting the debug level.

6.5 Endpoint lcfd.log File

The lcfd.log file can be found in the /Tivoli/lcf/dat directory and will contain a log of the endpoint login as well as any upload or download calls to or from the gateway. The amount of data recorded depends on the logging, or debug level. This section shows examples of the log and tells you how to set the level.

Figure 75 shows an example of lcfd.log. This log was produced at debugging level 1 when a Tivoli Inventory profile was distributed:

```
Nov 17 21:08:00 1 lcfd node_login: listener addr '0.0.0.0+2950'
Nov 17 21:08:02 1 lcfd gblnt-eplocal is dispatcher 22 in region 1212391543
Nov 17 21:08:02 1 lcfd write login file 'lcf.dat' complete
Nov 17 21:08:02 1 lcfd Logging into new gateway...
Nov 17 21:08:02 1 lcfd gblnt-eplocal is dispatcher 22 in region 1212391543
Nov 17 21:08:02 1 lcfd write login file 'lcf.dat' complete
Nov 17 21:08:02 1 lcfd final pid: 398
Nov 17 21:08:02 1 lcfd Login to gateway 127.0.0.1+9494 complete.
Nov 17 21:08:02 1 lcfd Ready.  Waiting for requests (0.0.0.0+2950). Timeout 120.

Nov 17 21:37:51 1 lcfd Spawning:
g:\Tivoli\lcf\dat\3\cache\bin\w32-ix86\TME\INVENTORY\INV_ENDPT\inv_endpt_meths.exe, ses: 17bfda6a
Nov 17 21:42:40 1 lcfd Spawning:
g:\Tivoli\lcf\dat\3\cache\bin\w32-ix86\TME\INVENTORY\INV_ENDPT\inv_endpt_meths.exe, ses: 17bfda6b
```

Figure 75. Typical lcfd.log - Inventory Profile Distribution

The log in Figure 76 was also taken using debug level 1 and shows the messages logged when the gateway is not available:

```
Nov 17 13:56:42 1 lcfd Doing initial login broadcast...
Nov 17 13:56:42 1 lcfd No gateway found.
Nov 17 13:56:43 1 lcfd node_login: listener addr '0.0.0.0+2950'
Nov 17 13:56:43 1 lcfd Trying last known gateway ...
Nov 17 13:57:28 1 lcfd gw login failure: i=2147483647 : ../../src/comm/netio.c:2
13 [cti_create_client or cti_timed_create_client] : loc=3, cls=2, dec=999, sys=1
0060, tli=0, evt=0
Nov 17 13:58:13 1 lcfd gw login failure: i=0 : ../../src/comm/netio.c:213 [cti_c
reate_client or cti_timed_create_client] : loc=3, cls=2, dec=999, sys=10060, tli
=0, evt=0
Nov 17 13:58:13 0 lcfd 11/17/98 01:57:44 (4): resource '' not found
```

Figure 76. Typical lcfd.log - Gateway Unavailable

The debug level of `lcfd` can be changed during start-up with the `-d` flag. Refer to the `lcfd` command in the *Tivoli Framework Reference Manual*.

Other important files in the /Tivoli/lcf/dat directory on the endpoint are the last.cfg configuration file and lcf.dat file, which contains the list of available gateways and the endpoint OID.

The cache sub-directory will contain the binaries downloaded from all method calls to the gateway.

6.6 Other Commands

Important

Most activities described in this book do not involve changes to the object database. However, direct object invocations, IDL calls, and attribute changes in the Tivoli object database have the potential to cause unpredictable results and possibly the complete failure of your TMR. Recovery, if at all possible, would usually involve at least a restore of the object database. Tivoli support personnel are unlikely to be able to assist in rectifying such changes. We recommend you back up your object database before performing any direct manipulation. Test changes first on an isolated test TMR and keep a log of every action performed.

This section gives short descriptions of a number of commands you may use in certain problem determination exercises. The full syntax of each command is given in the *Tivoli Framework Reference Manual.* It is strongly recommended that, if you are going to be using any commands that directly manipulate the object database, you become familiar with the Tivoli Advanced Development Environment (ADE) documentation. The ADE manuals go into much more detail about the use and structure of the object database.

6.6.1 The objcall Command

Request the specified object to run the specified method with zero or more arguments. This command exits with the method's exit code and is only used for non-IDL methods.

No Tivoli role is required to run the `objcall` command, but you must have the roles required by the method you want to run:

```
objcall [-a] [-b] [-c  group:role:...] [-e] [-F filedescriptor] [-k
len] [-n] [-p port] [-s] [-T transtype] OID METHOD [args...]
```

For example, we can run an object's *contents* method using the format `objcall OID method` as follows:

```
# objcall 1264987995.1.227 contents
ATTRIBUTE:_BOA_id
ATTRIBUTE:actions
ATTRIBUTE:behavior
ATTRIBUTE:class_objid
ATTRIBUTE:class_type
ATTRIBUTE:collections
```

Various methods are documented throughout the product chapters. Check the index under method for a list of those covered in this publication.

6.6.2 The idlcall Command

Invokes IDL operations from the shell command line. In practice, we will use `idlcall` to provide output in a slightly different (often cleaner) format than `objcall` when exporting and importing data for scripts:

```
idlcall [-T transtype][-v] targetobject operationID [args]
```

The *targetobject* argument specifies the clear text string representation of the target; in Tivoli, we use the OID.

The operationID specifies the operation name, optionally fully qualified as a CORBA repository ID (for example TMF_Task::TaskLibrary::remove_job).

The `args` specifies any input or in/out arguments.

For example:

```
idlcall 1264987995.1.345 TMF_task::TaskLibrary::remove_job job_name
```

6.6.3 The idlattr Command

This command gets or sets object implementation attributes:

```
idlattr -t [-s | -a | -g | -v] targetobject attrname typename [value]
```

The `-t` indicates that the argument list contains the attribute type name in the CORBA repository ID format (this is required).

The `-s` or `-g` indicate a set or get operation.

Note

Note that if neither `-g` or `-s` are specified, then the default is `-s`. You are unlikely to want to do this until you are very familiar with the attribute in question. If you are trying to look at an attribute (get) and omit `-g` and `-s`, `idlattr` will perform a set operation, and because you did not specify any data, `idlattr` will try to obtain the data from stdin.

The `targetobject` will usually be an OID.

The `attrname` specifies the unscoped attributed name, such as *label* or *behavior.*

The `typename` specifies the fully-scoped attribute type.

The `value` argument will be required if you are executing a set operation.

The following example gets the behavior attribute from the OID specified. The OID is of an instance manager (class object). The behavior attribute contains the OID of that class' behavior object. The subsequent `objcall` uses the contents method to list the contents of the behavior object:

```
# idlattr -tg 1264987995.1.322 behavior TMF_SysAdmin::InstanceManager
1264987995.1.324
# objcall 1264987995.1.324 contents
ATTRIBUTE:label
ATTRIBUTE:skeleton
METHOD:_get_label
```

Note that, in most cases, you would use `objcall OID getattr` rather than `idlattr`.

6.6.4 The resolve Command

The `resolve` command may not be implemented on all interpreter types. It is not a standard, documented, or Tivoli-supported command. It does the same job as `objcall OID resolve methodname` (or `attributename`). That is, it determines in which object a given method or attribute resides so long as that method or attribute is inherited by the object whose OID is specified. This command will just save a little typing:

```
resolve 1234567890.1.347 xterm
```

and is equivalent to:

```
objcall 1234567890.1.347 resolve xterm
```

6.6.5 The irview Command

This command is used to find unknown methods and attribute names and arguments to attribute types:

```
irview repository-id [contents | describe | describe_contents |
describe_interface | lookup_name | lookup_id | _get_name| _get_type |
_get_defined_in | _get_mode | check_consistency | get_inherited_interfaces]
```

See 2.5.2, “If the Method is Unknown” on page 53 of Chapter 2 for an example of using `irview`.

6.6.6 The tmstat Command

The transaction mechanism is complex and not open to user manipulation. However, you can use this command if you want to examine the current transactions, locks, and their states. You are unlikely to need this command often, but it does give you a way of seeing what is going on from a transaction perspective.:

```
# tmstat

                        Transactions for 0.0.0

    Trid         Type   State  Resources Polling Coord   Parent   MTid
-------------------------------------------------------------------------

{1264987995:1,1264987995:1,14:10343}
                Top-T   running   No      No    running running 12890
{1264987995:1,1264987995:1,14:10343},{1264987995:1,1264987995:1,14:10344}
                Revoke-Trunning   No      No    running running 12891
{1264987995:1,1264987995:1,14:10343},{1264987995:1,1264987995:1,14:10344},
{1264987995:1,1264987995:1,14:10345}
                Revoke-Tcommit    No      Yes   running running 12892
{1264987995:1,1264987995:1,14:10343},{1264987995:1,1264987995:1,14:10344},
{1264987995:1,1264987995:1,14:10345},{1264987995:1,
1264987995:1,14:10346}
                Revoke-Tcommit    No      Yes   running running 12893
```

The oserv maintains the transaction logs. If there are problems with the logs, then messages will be included in the `tmstat` output.

See 6.4.1, “Transaction Log Files and tmstat” on page 172 and 2.3.4, “Transactions” on page 19 for more information and refer to the `tmcmd` in the *Tivoli Framework Reference Manual* to force a transaction state change.

Chapter 7. Tivoli Framework Core Services

This chapter provides an overview and a look at the internals of the following core Framework services:

- Tivoli Administrators
- Notices and Notice Groups
- Interregion issues
- Tasks Library
- The Scheduler
- Multiplexed Distribution and BDT/IOM
- UserLink and DHCP

7.1 Tivoli Administrators

During this section, the references to the system root account apply to both UNIX root accounts and Windows NT Administrator accounts. Throughout this book, when we refer to administrators, we usually mean Tivoli administrators or the person performing an administration function depending on the context.

Tivoli uses administrators to delegate the use of the system root account without giving those administrators the system password or complete control. There are two ways to facilitate this delegation of system management tasks:

Authorization roles Roles that define the scope of control an administrator has over objects in a TMR.

Policy Regions Resources that can be grouped into specific policy regions, and only the assigned administrators can see and manage the resources.

7.1.1 Authorization Roles

These roles can be set throughout the TMR or on specific policy regions. Setting roles for each policy region gives administrators control over only the resources in that region. For a complete list of roles and the activities they enable administrators to perform, see the *Tivoli Framework Planning and Installation Guide*.

When you generate a task, an administrator with the correct authority can specify that the task will run using the properties of another administrator.

See 7.4.4.2, “Creating a Task” on page 241 for more information about this capability.

7.1.2 Policy Regions

A policy region is a special collection or container in which resources are created and managed. The resources within a policy region can be governed by allowing only certain administrators to have control. We can also make use of resource policies. The two types of resource policies are default and validation:

Default Policy Defines default values for attributes when a new instance of a resource is created.

Validation Policy Defines the valid values for attributes that are checked when a new instance is created or modified. Validation policies run when you set and close the dialog. This will check what values an administrator is attempting to assign against those that they are permitted to assign.

Default and validation policy is described in some detail in the *Tivoli Framework User’s Guide*.

Figure 77 on page 189 shows an administrator’s desktop containing policy regions.

Figure 77. Sample of a Desktop Containing Policy Regions

Each policy region can be assigned Managed Resources. This determines which types of resources can be managed within that policy region. You can change managed resources for a Policy Region by holding the right mouse button on the required policy region and selecting **Managed Resources...**. Note that only the resources listed in Current Resources can be created and managed by this policy region (see Figure 78). Failing to select the correct managed resources for a policy region is a common mistake when creating new regions or adding new Tivoli applications.

Figure 78. Setting Managed Resources for a Policy Region

7.1.3 Creating Administrators

Most system administrators have a Tivoli administrator that maps to their login, but users of multiple systems can use the same Tivoli administrator through the use of the login names feature (see Section 7.1.4, “Using a Single Tivoli Administrator for Multiple Users” on page 195). Depending on your administration policy, you may find it necessary to maintain a Tivoli administrator ID for each system administrator. The following are two examples of when this might be necessary:

- Each administrator needs different authorization roles.
- Actions performed at the target may be logged locally with the user login name or ID-mapped name specified during the creation of a Tivoli administrator. If the Tivoli administrator account was shared, then this would not identify which person performed the action unless something was also generated in a Tivoli notice group.

Each system to be managed by Tivoli may already have user account names defined for administration. In many sites, these administrative accounts may use many different names. In addition, these names may differ from those a Tivoli Administrator will log in with when they first connect to a system. This leads us to three entities defined with a Tivoli administrator:

- The label used to identify the administrator within Tivoli.
- A mechanism for identifying which Tivoli administrator will be used based on the account that the person used when they signed onto the system.

- The ability to perform an operation using another system-specific account name.

The combination of these entities allows someone to log on to a system and perform a Tivoli operation. So, for example, the Tivoli administrator definition determines that the user who logged in as viki should become Tivoli administrator SWDistAdmin, and that when performing an action on a Windows NT endpoint, that action should be performed as the user Administrator.

Therefore, a Tivoli administrator record defines a relationship between the user account name (or names) an administrator logs into the system with to the Tivoli administrator name and to the user names required to perform activities on managed systems. In our example, an administrator may log into UNIX using the name viki. When this user starts the Tivoli desktop, providing the user is authenticated, they will receive a Tivoli administrator desktop that could be of a different name, such as SWDistAdmin. Now, when the SWDistAdmin Tivoli administrator starts a task on a Windows NT managed node, the user that is used for access to the Windows NT system might be another name, such as Administrator. Once defined, viki simply needs to log in to UNIX (or the Desktop for Windows) and start her administrative activities.

Tivoli uses the term Set Login Name (or Current Login Names) to refer to the user name with which an administrator logs into the system. The term Administrator Name (or Administrator Label) refers to the name used to uniquely identify a Tivoli administrator. The terms User Login Name and Group Name refer to the names used to initiate requests on systems of different operating system types. The User Login and Group Names can also be specified by ID maps that can pick out a different user name to use for each platform type to which a management request is going. How these names are implemented is covered in the rest of this section.

The creation of a Tivoli administrator can be done through the desktop or using the command `wcrtadmin`. In the examples presented here, we show the desktop methods. The `wcrtadmin` command is detailed in the *Tivoli Framework Reference Manual*.

Choosing **Create & Close** from the Create Administrator dialog will fail unless you perform the steps to set up an administrator, as follows:

- Enter the administrator name, which also becomes the icon label.
- Enter a User Login Name.
- Enter a Group Name.

- Select **Set TMR Roles...** or **Set Resource Roles...** for this administrator.
- Select **Set Logins...**.

The main fields of this dialog are as follows:

Administrator name The contents of this field will be used to generate an object name and icon label for the administrator.

User login name User name that will be used for Tivoli activities detailed below. This name must be defined as a system user on the TMR server and all targets where this administrator will perform Tivoli activities.

Group name Group name that will be used for certain Tivoli activities detailed below.

The user and group names must exist on all the targets used by the administrator but do not have to match the person's login name. UNIX numeric user and group IDs are not allowed. Since an administrator may have different user names on different systems, it is possible to map the user login name and group name to an alternative name specific to a platform type. See Section 7.1.5, "ID Mapping" on page 199 for more information on this feature. An ID map is specified in these fields by preceding the name with the dollar sign ($).

An administrator uses these names for functions, such as:

- Opening an Xterm for a Managed Node.
- Performing a Tivoli database backup.
- Executing any tasks where the user or group name is specified as asterisk (*).
- Writing to a log file.

7.1.3.1 Create Tivoli Administrator Example

Suppose we want to create a Tivoli administrator for a user called smoore who may only administer from the k124a node. The user name this administrator will need to use for Tivoli activities on the endpoints is ausres48. Depending on the activities performed, this user ID may also need to exist on managed nodes on which processes will be run:

1. Open the Administrators Icon from the main Tivoli desktop.
2. Select **Create > Administrator...**

3. We specified a name/icon label for the administrator (Users_from_k124a), the user account, and group to use for administrative actions (ausres48 and staff).
4. Select **Set TMR Roles...**. This dialog lists current and available roles for the TMR for the chosen administrator. We selected the *user* role for this administrator in the TMR.
5. Selecting **Set Login Names...** brings up the dialog shown in Figure 79. Here we specify the system users that will use this Tivoli administrator ID. For our example, we want smoore to use this Tivoli administrator for Tivoli operations and only when he is logged in at the node `k124a.austin.ibm.com` (`smoore@k124a.austin.ibm.com`). At this point you must press **Enter** to add the Login Name. Only when the newly-entered Login Name appears in the Current Login Names list, can we use **Set & Close**.

Important

Note that login names can not appear in the Current Login Names list for more than one Tivoli administrator. Framework prevents you from adding a login name that already exists in the current login names list of another Tivoli administrator in the same TMR, but it does not prevent you from specifying an unqualified name in one record and a qualified name in another. Prior to Framework 3.2, this would result in the first match succeeding. From 3.2 onward, Framework first checks all entries for an exact match. For example, if you had damian as a current login name for one administrator and `damian@basingstokeMN` in another administrator's list, prior to 3.2, a user logging in as damian from basingstokeMN could function as the administrator with just damian in the current login name list if that one matched first.

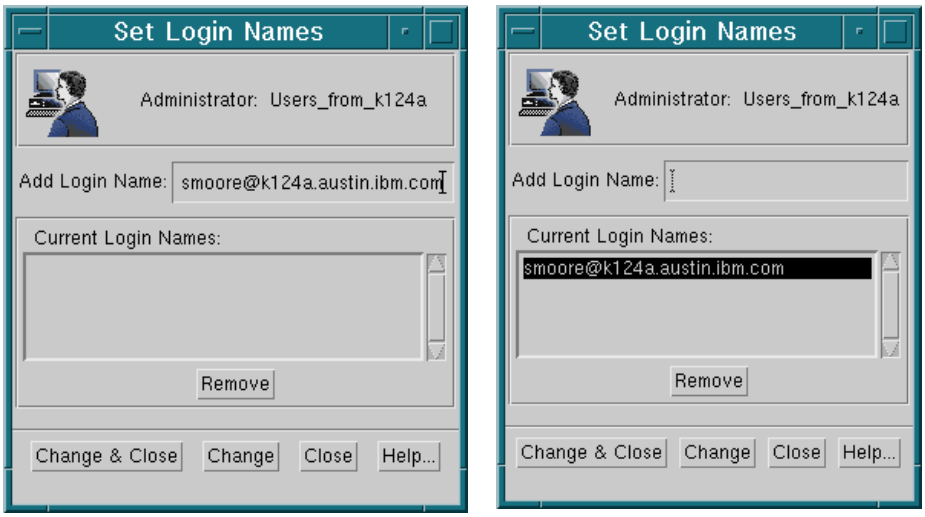


Figure 79. Set Login Names Dialog - before and after Pressing Enter

At this point, we can select **Create & Close** from the Create Administrator dialog. However, there are two further properties that you can set:

- Resource Roles - Rather than specify a role that would apply to every resource in the TMR, it is preferable to restrict the definition of roles to particular resources. Typically, this would be used to enable a chosen role in selected policy regions.
- Notice Groups - You need to choose which notice groups to which this administrator will subscribe. These should be limited to just those relevant to the activities that administrator will perform. See also “Notices” on page 209.

7.1.3.2 Example of User Login Name and Current Login Name

Refer to Figure 80 on page 195 for the screens for this scenario:

- User `smoore@k124a.austin.ibm.com` starts the desktop on the k124a managed node from a dtterm session.
- Then, they open the `rh0255c.itsc.austin.ibm.com-region` Policy Region.
- Then, they click with the right mouse button on the k124a icon and select **xterm**.

  Even though smoore started the desktop on the k124a managed node, the xterm still runs with user ID ausres48, the User Login Name entered in the Create Administrator dialog.

The xterm method has been designed to use the User Login Name. The group that is selected is *nobody.* This behavior is up to each method to determine.

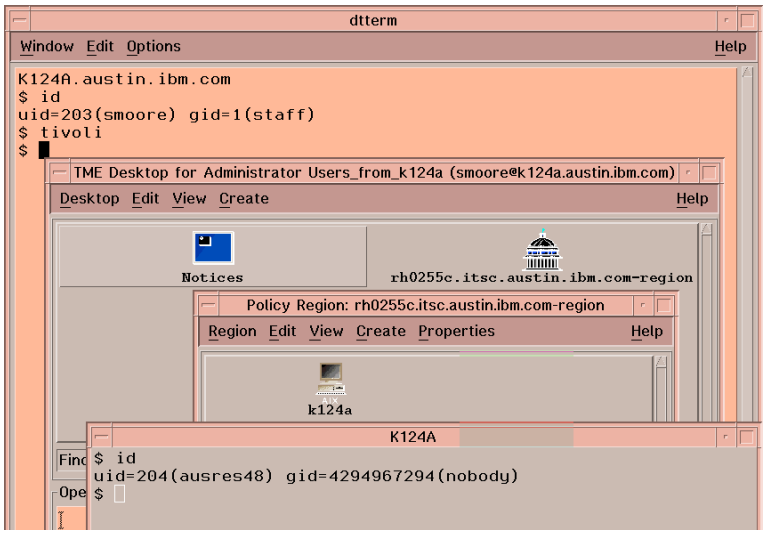


Figure 80. Administrator Login Name versus Current Login Name

NOTE

The ausres48 user ID is defined on k124a as a UNIX user ID. It must also be defined on all other managed nodes where smoore will be expected to run Tivoli-related processes.

7.1.4 Using a Single Tivoli Administrator for Multiple Users

Often, at large sites, there are groups of people who perform similar jobs staffing that job on a seven days a week, 24 hours a day basis. One example of these kinds of individuals would be an operations group whose job it may be to watch for any alerts generated by Distributed Monitoring. Another group may be a call center group, whose job can entail pushing out software distribution packages. In any case, these groups of individuals may all have the same icons, notices, and other items on their desktop. As these groups grow, and people change roles within an organization, the administration of these groups of users can become a burden.

Using the set login names function on a desktop, you can define a single Tivoli administrator, but associate multiple login names with that one

administrator. All actions that occur in Tivoli will be logged to a notice group with the login name not the Tivoli administrator name. With this, you can maintain a single desktop but still enforce accountability for actions within Tivoli.

To demonstrate an example of this, we created a Tivoli administrator, unix_oper. As shown in Figure 81, the user login name that will be used to execute actions on the endpoints was tivop, and the group was staff.

Figure 81. Multiple Use Tivoli Administrator - Administrator Properties

We also set the login names to be childres and rhonda, two UNIX operations personnel, shown in Figure 82.

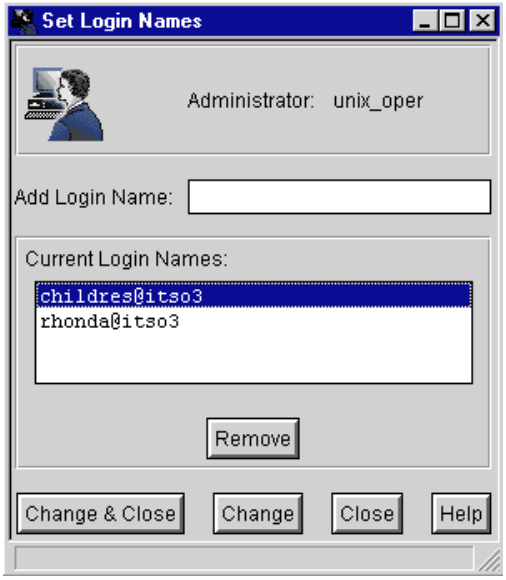


Figure 82. Multiple Use Tivoli Administrator - Set Login Names

The UNIX operator rhonda started a Tivoli desktop and distributed software. The software distribution notice group had the entry shown in Figure 83 on page 198. Note that, although the Tivoli administrator used for the operation was unix_oper and the user account used to execute the operation was tivop, the notice is logged under the name the operator logged into the system with, rhonda.

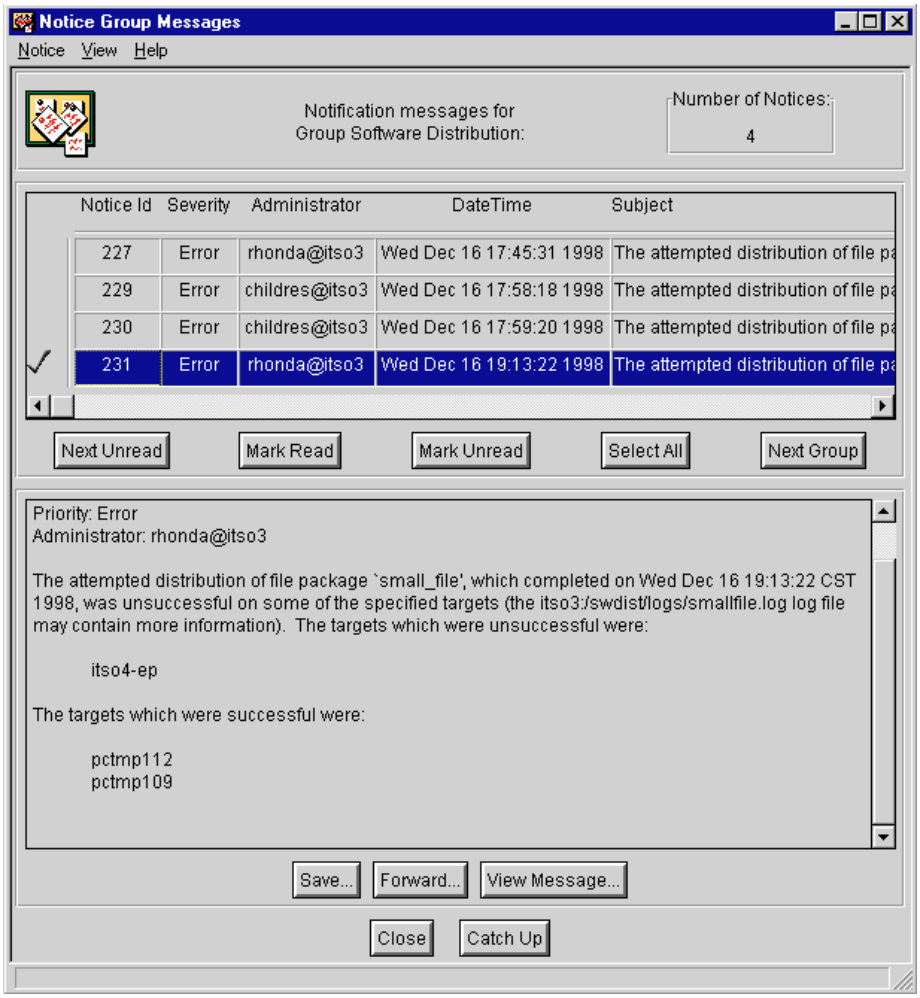


Figure 83. Multiple Use Tivoli Administrator - Notice Group Messages

Next, the UNIX operator childres started a Tivoli desktop and did a software distribution. We can see in Figure 84 on page 199 that the entry in the software notice group is logged under the childres user.

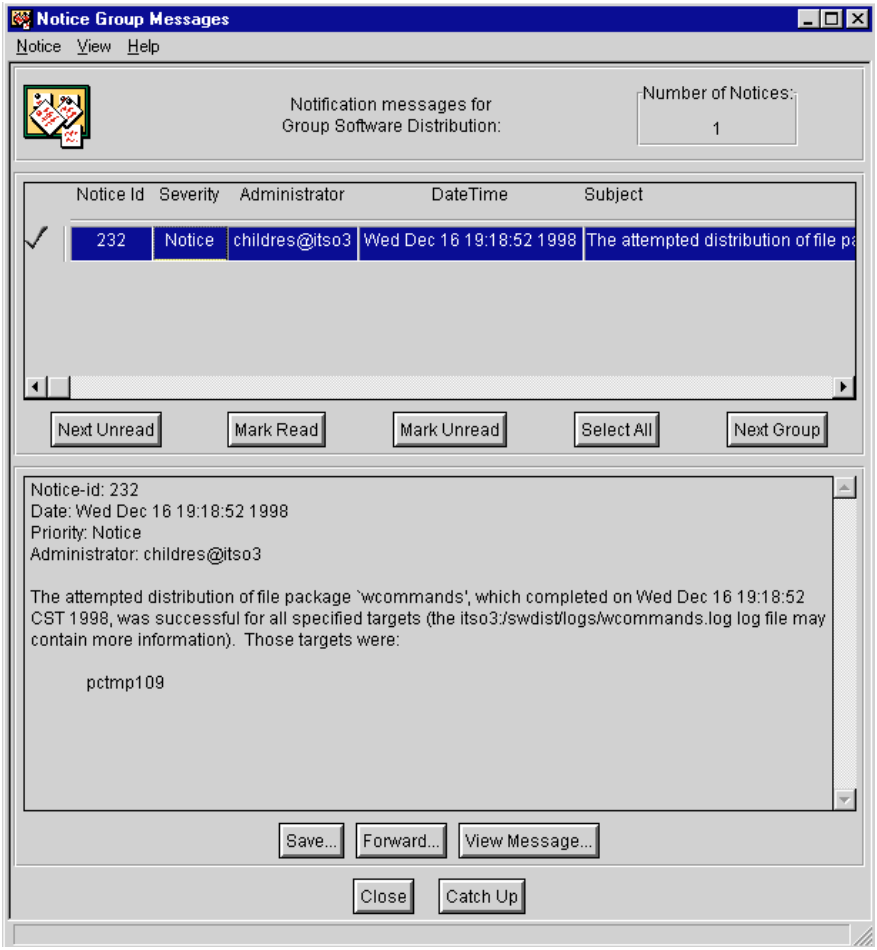


Figure 84. Multiple Use Tivoli Administrator - Notice Group Messages 2

So, for each of the notices, you can see the operator who performed the action is logged, and so an audit trail is provided.

7.1.5 ID Mapping

Tivoli Administrator *set (or current) login names* determine which Tivoli administrator attributes and desktop will be used based on which ID someone used to login to the operating system. The *user login name* or properties determine which ID will be used to run tasks and other activities that the administrator initiates. But what happens if the ID for running tasks needs to be different on different platform types? For example, the root user of the Tivoli Framework is *root* on UNIX systems and *Administrator* on Windows NT.

Tivoli allows these types of differences through the use of ID mappings that can be managed with the `widmap` command. Two default ID maps, `root_user` and `root_group`, are set when the Framework is installed, as detailed below:

```
#widmap
usage:
    widmap list_maps
    widmap add_map|rm_map|list_entries <mapname>
    widmap add_entry <mapname> <interp> <idname>
    widmap rm_entry|resolve_entry <mapname> <interp>

#widmap list_maps
root_group root_user#

#widmap list_entries root_group
default root
sunos4 wheel
aix3-r2 system
aix4-r1 system

#widmap list_entries root_user
default root
w32-ix86 Administrator
nw3 supervisor
nw4 Admin
os400-v3r2 QTIVROOT
os400-v3r7 QTIVROOT
```

Use of an ID map is specified when the administrator is created using the Create Administrator dialog. Instead of entering a user name or group name, an ID map name can be specified by preceding it with a dollar sign ($), as in `$root_user`.

When an ID map is used, Tivoli will check the interpreter type on which the task is to be run and then look up the name to use as specified in the ID map.

7.1.5.1 ID Mapping Example

In “Creating Administrators” on page 190, we created a Tivoli administrator with the label Users_from_k124a. We originally had the login name for this administrator as ausres48 and the group as staff. We also added a Set Login Name for smoore. The result was that when smoore logged in, he became the Tivoli administrator Users_from_k124a, and when he performed certain operations, they were executed as ausres48 on the targets.

To demonstrate ID mapping, we are going to modify the way the Tivoli administrator Users_from_k124a is represented on different platforms by using an ID map. The map name that we will use is ausres48_user. With this

ID map, we wish to continue to use ausres48 to execute operations on all platforms except AIX. On AIX, we now wish to use the name stanley.

Implementation Tip

There is no parameter that defines whether a map is for a group or a user; so, it is a good idea to name the map with an extension of either *user* or *group*. It will be a lot easier to find them using `widmap list_maps`.

The following `widmap` commands add the new ausres48_user map, add an entry to that map to specify stanley as the user to use on AIX, and set the default to ausres48 that will be used for all other platforms. Finally, we check that the entries were created successfully:

```
#widmap add_map ausres48_user
#
#widmap add_entry ausres48_user aix4-r1 stanley
#
#widmap add_entry ausres48_user default ausres48
#
#widmap list_entries ausres48_user
aix4-r1 stanley
default ausres48
#
```

If the system is recognized as the `aix4-r1` interpreter type, then Tivoli will use stanley. For any other system (Windows NT, Sunsoft Solaris and so on), Tivoli will use `ausres48` as the user ID.

We must now modify the properties of the Tivoli administrator, Users_from_k124a, by changing the user login name from ausres48 to the ID map, $ausres48_user, as shown in Figure 85 on page 202.

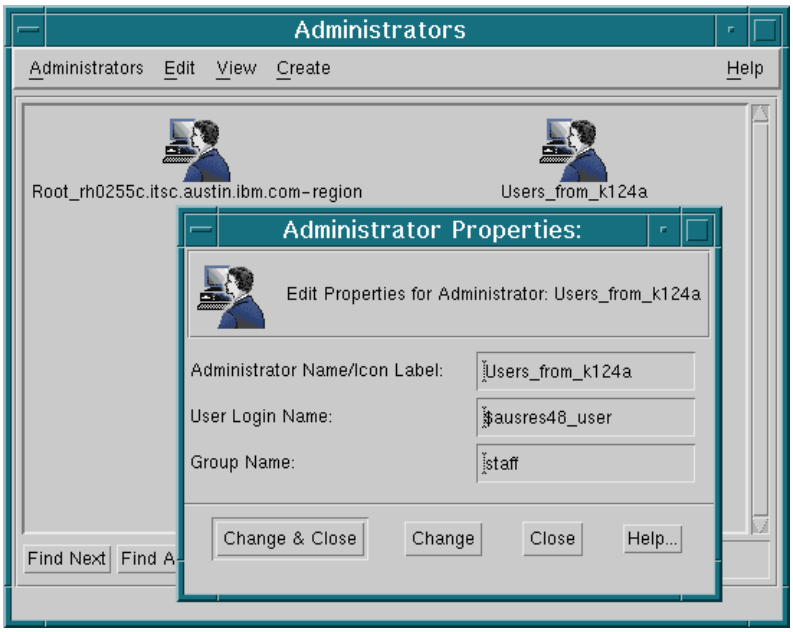


Figure 85. Entering an ID Map for a Tivoli Administrator User Login Name

We can run the same test for the User Login Name that we used in “Example of User Login Name and Current Login Name” on page 194. The user smoore will start a desktop on the k124a managed node and select an `xterm` from the policy region, then goes back to the same machine, k124a.

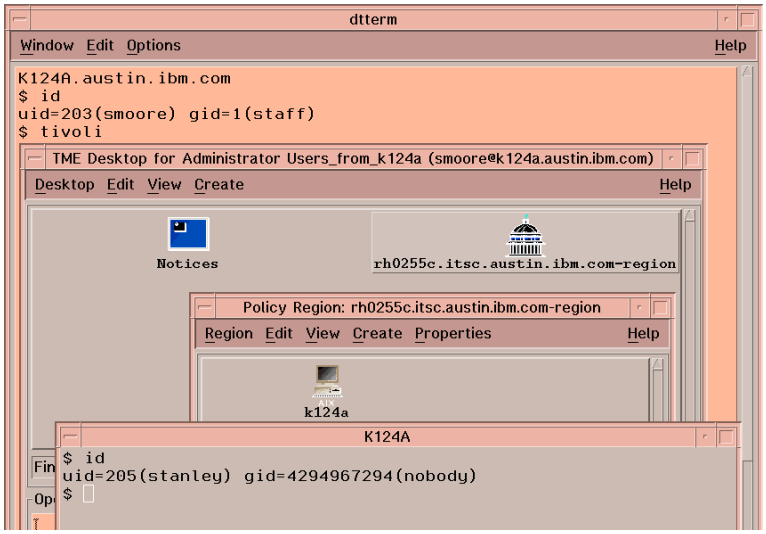


Figure 86. Administrator Using an ID Map

As we can see in Figure 86, the user ID selected by Tivoli is stanley because the k124a managed node has the aix4-r1 interpreter type.

Note

You cannot have more than one user for an interpreter type in a single map. It is not possible, for example, to specify the use of one user in one Windows NT domain and a second user in another domain.

There can also be a problem when mixing National Language Support (NLS) versions of operating systems, such as Windows NT. The Administrator ID must be the same spelling on all systems. Windows NT, by default, will use language-specific names for Administrator in each NLS version.

7.1.6 Removing and Deleting Administrators

A common problem with deleting administrators from the Administrator dialog is that users choose **Remove** instead of **Delete**. Removing administrators from the Administrator dialog does not remove them from the database, but deleting them does. The Remove option exists because some users wanted to create administrators and then remove them from the GUI where less-experienced administrators might accidentally delete them. Most often, the less-experienced administrator accidentally removes an administrator that they wanted to delete.

To solve this problem, use `wlookup -ar Administrator` to list administrators defined to Tivoli and then use `wln` to link the required administrator object back to a the Administrators collection, as in the following example:

```
# wlookup -ar Administrator
Root_outback-region     1999200098.1.192#TMF_Administrator::Configuration_GUI#
Viki    1999200098.1.566#TMF_Administrator::Configuration_GUI#
root@outback    1999200098.1.192#TMF_Administrator::Configuration_GUI#
viki    1999200098.1.566#TMF_Administrator::Configuration_GUI#
# wls /Administrators
Root_outback-region                  ! Viki is defined but not in the collection
# wln @Administrator:Viki /Administrators
```

The `wrm` command can be used to remove a resource from an administrator's desktop without deleting it. Note that `wln` and `wrm` can be used to add and remove other types of resource and not just administrators, as detailed below:

`wln <resource> <administrator>`

where `<resource>` is `@Resource:name` or `/Library/Resource/name` and `<administrator>` is `@Administrator:adminname` or `/Administrators/adminname`.

For `wrm`:

`wrm adm_resource`

where `<adm_resource>` is `/Administrators/adminname`.

7.1.7 Administrator Commands

There are several commands that can be used when troubleshooting administrator problems. Two of these commands are listed below:

Table 4. Administrator Commands for Troubleshooting

Activity	Context	Required Roles
Get Administrator Information `wgetadmin`	Administrators	admin (to get information about other administrators)
Set Administrator Information `wsetadmin`	Administrators	super or senior

Note

From 3.6, there is a new command related to managing administrators called `wauthadmin`. This command sets and revokes Tivoli root authority in the TMR. The first installed administrator in a TMR begins as a root administrator, and this command allows others to be added. A Tivoli root administrator has root authority on UNIX and Administrator authority on Windows NT. Use `wauthadmin -l` or `-lv` to show a list of root administrators.

Figure 87 on page 205 details the output of the `wgetadmin` command.

```
# wls /Administrators
Root_rh0255c.itsc.austin.ibm.com-region
Users_from_k124a
Root_tivdev02-region
Root_rh0255b.itsc.austin.ibm.com-region
#
# wgetadmin Root_rh0255c.itsc.austin.ibm.com-region
Administrator:  Root_rh0255c.itsc.austin.ibm.com-region
logins: root@rh0255c.itsc.austin.ibm.com
roles:  global  super, senior, admin, user, install_client, install_product
        security_group_any_admin        user
        Root_rh0255c.itsc.austin.ibm.com-region admin, user, rconnect
notice groups:  TME Administration, TME Authorization, TME Diagnostics, TME Scheduler
#
# wgetadmin Users_from_k124a
Administrator:  Users_from_k124a
logins: smoore@k124a.austin.ibm.com
roles:  Users_from_k124a        admin, user, rconnect
        security_group_any_admin        user
        global  user
        Profiles-rh0255c        Inventory_view, Inventory_scan, Inventory_edit,
Inventory Query
notice groups:
#
```

Figure 87. Output from the wgetadmin Command

The information that can be listed is:

- Administrator label
- Administrator logins
- Global roles: TMR resource roles
- Policy region roles: roles specific to a policy region. The policy region listed for Users_from_k124a is:

  `Profiles-rh0255c`

If a Policy Region is in a disconnected TMR, you will see only the Object ID instead of the name, but the roles will still display.

- Security groups: non-policy region groups that can have specific roles:
  - Scheduler (not listed)
  - Administrator (not listed)

    Note that the following two should **not** be modified from the default.

  - `security_group_any_admin`: allows methods with the role of any to be used.
  - Specific roles for an administrator. For example, with `Root_rh0255c.itsc.austin.ibm.com-region` the roles are:

    `admin`, `user`, `rconnect`.

Note

The `wgetadmin` and `wsetadmin` commands do not enable viewing or changing an administrator's user name or group name.

7.1.8 Administrator Roles

Administrators performing specific functions (those not administering everything), should have minimal roles in the TMR and the appropriate roles in the policy regions that they administer. This enables the creation of new resources without making them automatically accessible.

There are two disadvantages to this policy:

- Appropriate roles must be added to each administrator managing the resource.
- Adding new administrators takes longer because you have to select each resource and assign roles individually.

Tivoli users are not always aware of where roles are required. Appropriate roles are often given in a subset of the locations where those roles are needed, but less obvious locations are often missed.

Take, for example, an administrator who wants to distribute a file package. The file package lives in a profile manager that lives in a policy region. But, in order for the administrator to successfully distribute the file package, the administrator must have the appropriate role in the policy region where the file package resides and in the policy region (or regions) where the endpoints exist.

Note

Roles are not hierarchical. A senior role does not imply administrator or user. However, some applications and methods do implement their own form of hierarchy. For example, a user with super role could execute a user role method for example. To be sure, always assign the correct roles as listed by the product documentation.

7.1.9 Interregion Administration

Given the correct interregion connections, a Tivoli administrator created in TMR A can perform actions on TMR B but cannot run commands directly from TMR B. The oserv authenticates a user as an administrator in the local TMR, and then any actions performed by the administrator are authorized before being executed. An oserv will not authenticate an administrator from a remote TMR.

Also, you cannot have more than one administrator with the same login name. It is recommended that administrators perform work in their own TMR.

There are several layers of checking to determine whether an administrator has permission to perform an activity:

- Administrator is authenticated as a Tivoli administrator. This only happens within the local TMR. Each administrator who will be performing tasks must be defined in the TMR where they perform the task.
- Administrator is authorized to perform the activity based on the authorization roles.
- Activity is performed as the user and role specified in the method. This requires the correct user, group, and roles to be defined.

7.1.10 Summary of Hints for Defining Administrators

This list summarizes hints from the previous sections:

- You should avoid giving remote root access. Always specify a host name if you want to allow a root user to use a Tivoli administrator name (use something like `root@hostx.domain`).
- If you set a Login Name, no checking for the validity of the user name is done by Tivoli when the administrator is created or modified. There is no check that the user actually exists until a method needing the user name is executed.

- Using the asterisk (*) in the user ID and group fields when defining a task will map them to the user login name and group name in the Create Administrator dialog. ID maps can also be used in the task fields when defining tasks. See Section 7.4.4.2, “Creating a Task” on page 241 for more information.
- Create Administrator dialog - The minimum setup is to set the fields in the Create Administrator dialog plus setting the logins and roles.
- You will have to subscribe the relevant users to notice groups as new applications are installed.
- You cannot grant roles that you do not have unless you are the root administrator on the TMR where you are granting the roles.
- Roles are not hierarchical. They must be selected and used individually.
- User and Group ID maps can be used when the same administrator has different user names on different platforms.
- You should specify minimal roles in the TMR and specific roles in the appropriate policy region.
- Roles do map directly across connected TMRs, except super for the TMR, which maps to user for the remote TMR.
- You should not try and set the same current login name (through the Set Login Name dialog) for more than one administrator.
- Tivoli, at Versions prior to 3.1.2, would not prevent you from deleting the last Tivoli administrator with the `wdel` command.
- For an NT administrator user ID, you should also enter the NT domain when setting a Login Name. For example:

  ```
  domain/user@managed_node
  ```

7.1.11 Hints for Troubleshooting Administrators

- Does the administrator have the proper role to perform the activity?
- If trying to perform something out of Tivoli, such as write a file, does the Tivoli administrator have the appropriate platform (for example, UNIX or Windows NT) permissions?
- BAD_ID in the error means the account name is not valid on the target on which the method is being executed.
- UNAUTHORIZED could be the logon is not valid, or the authorization role is not high enough to perform the task.

7.2 Notices

Almost every systems management action in Tivoli will result in a notice being posted in a notice group. This is particularly useful for an administrator overseeing the work of other administrators.

7.2.1 Subscribing Tivoli Administrators to Notice Groups

Each time a new product is installed, the administrators who are administering that application must be subscribed to any new notice groups if they want to see the notices. Unfortunately, they must be assigned to each individual administrator either through the desktop or the command line.

7.2.2 Notice Commands

The following commands are used to debug and view notices. The syntax for these commands is in the *Tivoli Framework Reference Manual*:

`wlsnotif` Lists all current notices for an administrator. This command can be used to list the subjects or full text of notices.

`wtailnotif` Lists new notices as they become available. This command can be very useful when re-creating problems. This will connect to the notification server and display new notices as they are posted. When you are re-creating a problem, you can have a window open with `wtailnotif` running for notices posted by specific administrators or watching for notices to selected notice groups or select notices to be displayed based on their priority. The output will include the date and time of the notice, the group it was posted to, its priority, the administrator, and the activity that took place (notice text). This saves having to keep going back to the notice board and scanning through for unread notices. Note that using `wtailnotif` does not mark the notices as read; the notice board activity is unaffected.

`wsndnotif` Sends a notice to a specified notice group. It is useful in debugging TEC Notification Adapter problems and can be used in custom scripts. See also Section 7.2.5, “wsndnotif - Adding a Notice from the Command Line” on page 212.

`wexpnotif` Expires all or some notices of a specified notice group. This command is helpful if there was a run-away application that posted too many notices and filled up disk space. Notices expire automatically after 186 hours (one week).

The database containing notices is automatically extended as new ones are added and will automatically reduce itself as they expire.

Table 5. Notice Group Commands

Activity	Context	Required Role
`wlsnotif`	TMR	user, admin, senior, super
`wtailnotif`	TMR	senior, super
`wsndnotif`	TMR	user, admin, senior, super
`wexpnotif`	TMR	senior, super

7.2.3 Restoring the Notices Database

Notices are not restored during a normal restore. This is because the backup could be several weeks/months old, and the old notices may or may not be relevant. Prior to restoring a previous Notices database, especially if you had an unplanned restore of the TMR, you will want to explore the information contained in the notice group for any information that may lead to cause of any problems. The old notices and the notice database can be restored with the `wbkupdb` command. See Section 5.5, “Items Not Restored from a Backup” on page 123 for more details.

Notices previously marked as read can only be re-read from the desktop if there is at least one notice in the notice group. Old notices can be re-read from the command line using the `wlsnotif` command if you have the number of the notice. See Section 7.2.4, “Re-Reading Old Notices” on page 210 for details on how to read notices again in a group where all notices are marked as read.

Notices work like Usenet News in that there is one central notice database called notice.bdb where all notices are posted, and each administrator object is responsible for keeping track of which notices have been read.

7.2.4 Re-Reading Old Notices

Suppose you want to look at the old notices in the Authorization group for the root administrator. If you left at least one notice marked as unread, then this would work, but the last time you went into the group, you selected all of them as read. Now, when you try to select the TME Authorization Notices, you get an error:

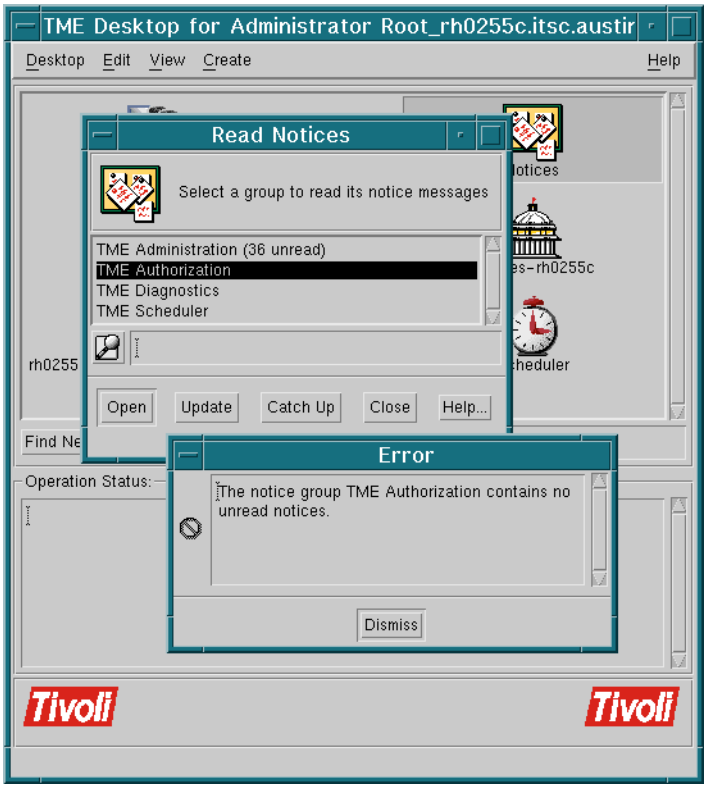


Figure 88. Error Opening a Notice Group - Contains No Unread Notices

There are several ways to get around this problem:

1. Make sure you never close a Notice Group unless you have at least one notice marked as unread.
2. Look at the list of notices from the command line. Use the `wlsnotif`, but you must know the notice number if you want to view a notice marked as read. The `wlsnotif` command lists only unread notices by default.
3. Manually add a new notice to the Notice Group you want to open using the `wsndnotif` command. Now you can go back to the GUI and open the Notice Group. See Section 7.2.5, “wsndnotif - Adding a Notice from the Command Line” on page 212.

The following example shows a subscription to four notice groups using the command `wlsnotif -g`. We can read a notice from the TME Authorization Notice Group as long as we know the number. We can list only the header by using the `wlsnotif -l` option.

```
#wlsnotif -g
TME Administration
TME Authorization
TME Diagnostics
TME Scheduler
#
#wlsnotif -n 'TME Authorization' 23 | more
Notice-id: 23
Date: Mon Nov 17 20:46:45 CST 1997
Notice-Group-Name: TME Authorization
Priority: Notice
Sent-By-Administrator: root@rh0255c.itsc.austin.ibm.com

Roles Changed for Administrator Users_from_k124a
The resource roles for the administrator named Users_from_k124a were changed. Us
ers_from_k124a now has the following resource roles:

        Tivoli/Sentry Defaults-tivdev02-region


#
#wlsnotif -l -n 'TME Authorization' 23
#23       Mon Nov 17 20:46:45 Roles Changed for Administrator Users_fr
#
```

Figure 89. Looking at Notices from the Command Line

Note that we had to put the notice group name inside quotation marks because of the space in the label name: `TME Authorization`.

7.2.5 wsndnotif - Adding a Notice from the Command Line

If you need to add a new notice so that you can re-open the notice group from the GUI, use the `wsndnotif` command, as shown in Figure 90:

```
# wsndnotif "TME Authorization" Notice
This notice is to allow me to re-open the Notice Group from the GUI
Simon
#D
#wlsnotif -n "TME Authorization"
Notice-id: 25
Date: Fri Nov 21 12:10:17 CST 1997
Notice-Group-Name: TME Authorization
Priority: Notice
Sent-By-Administrator: root@rh0255c.itsc.austin.ibm.com

This notice is to allow me to re-open the Notice Group from the GUI
Simon


#
```

Figure 90. Using wsndnotif to Create a Notice

The `wsndnotif` command will allow input on new lines when you press **Enter**. To finish input and send the notice, you must press **Ctrl-D**.

The `wlsnotif` command will now show that there is an unread notice in the TME Authorization group without supplying a notice number.

If you want to view the Notices from the GUI again:

1. You have to re-open the **Notices** object from the main desktop. You can see that the TME Authorization notice group has one unread notice:

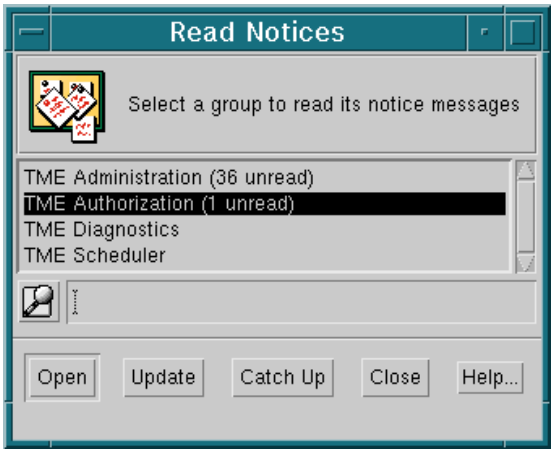


Figure 91. The New Notice Shown in the Read Notices Dialog

2. Select the group and **Open**. You can see the new notice listed in the panel.

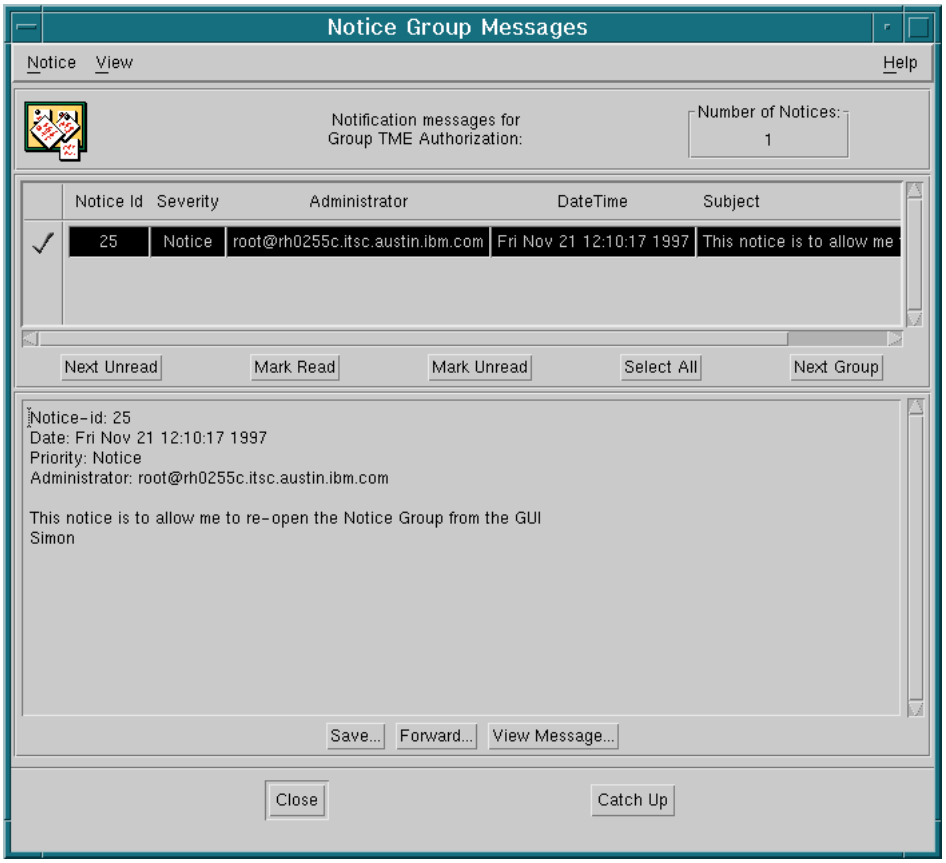


Figure 92. Looking at the New Notice in Notice Group Messages

3. To see the old notices, you must select **View>Display Old Messages**.

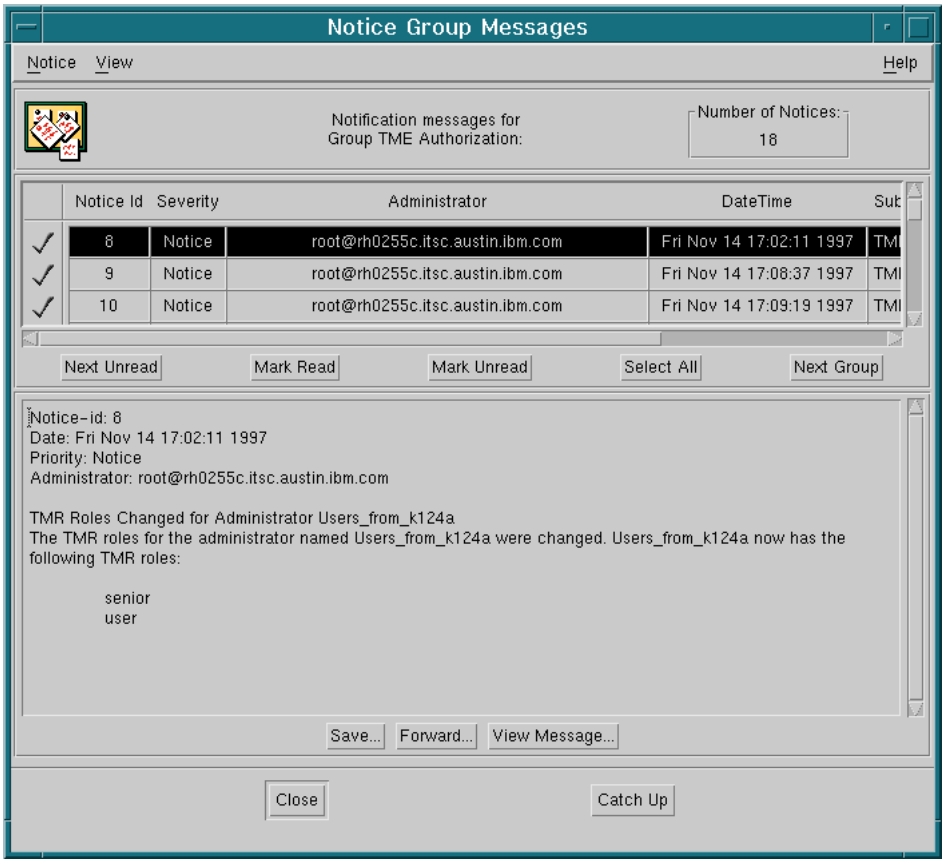


Figure 93. Looking at Previously-Read Notices in Notice Group Messages

You can see from this figure that the Notice numbers now start at 8 again, and they have a tick mark to signify that they have been read.

7.2.6 Troubleshooting Notice Groups

In the case where notices are not being posted to a notice group, check the following:

- Is the appropriate notice group defined to the Distributed Monitoring monitor, Software Distribution file package, and so on?
- Through the command line, can you post to the notice group?
- If multiple people are using a Tivoli Administrator desktop, check the previous notices. Someone else may have read the notice and marked it as read.

- Is the administrator you are using subscribed to the notice group?

7.3 Interconnected TMRs

Tivoli Framework allows TMRs to be connected. This gives greater flexibility in the following areas:

- Performance - The network and memory demands may become too large for one server. See the release notes for products you are installing to find the performance considerations.
- System administration - Multiple TMRs enable administration at a local level, either to meet organizational or geographical needs. An additional benefit of local administration is fault resiliency due to that fact that a failure will only affect resource management for the one TMR and not all resources in the enterprise.
- Security - An additional TMR can be used to restrict local administrators' access to certain machines within the enterprise. An additional TMR enables differing encryption levels within a Tivoli environment.

With the super authorization role and the TMR password, a user can connect or disconnect two or more TMRs.

7.3.1 The Tivoli Name Registry

The Tivoli Name Registry (TNR) provides a fast, space-saving way to access objects in:

- Large server object databases
- Multiple connected TMRs with differing speed connections

Each TMR contains a name registry object. The name registry lists names of both local TMR and interregion objects. It resembles a table of contents for the Tivoli object database and all remote databases and contains only references to objects in those databases.

All objects in the TMR that need to be referenced should be registered in the name registry when created, unregistered when deleted, and updated when their label is changed. Information from remote TMRs must be updated regularly to maintain accurate data.

You can use the `wlookup` command to view the objects from remote TMRs.

For a detailed description of the Tivoli Object Hierarchy and the name registry, see Chapter 2, “Tivoli Object Database Architecture” on page 9.

7.3.1.1 Interconnected TMR Name Registry Usage

A Tivoli application can look at the local name registry to find the references to the remote resources. This removes the need for an expensive process to scan for a resource in each of the connected TMRs. Even though a remote TMR may not be available, a complete list of managed nodes can be retrieved locally.

The following screen shows output from several commands displaying the connections and the local and remote resources in the Name Registry:

```
# wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255b.itsc.austin.ibm.com-region rh0255b.itsc.austin.ibm.com 1515280903
  <----> tivdev02-region  tivdev02   1482082604

# odadmin region
Region  TME Srvr     ipaddr     port
1360991896      rh0255c.itsc.austin.ibm.com     9.3.1.235  94  1360991896.1.0
1482082604      tivdev02.itsc.austin.ibm.com    9.3.1.134  94  1482082604.1.0
1515280903      rh0255b.itsc.austin.ibm.com     9.3.1.234  94  1515280903.1.0

# wls /Library/PolicyRegion
rh0255c.itsc.austin.ibm.com-region
Profiles-rh0255c
Queries-rh0255c

# wlookup -ar PolicyRegion
Profiles-rh0255c        1360991896.1.552#TMF_PolicyRegion::GUI#
Queries-rh0255c 1360991896.1.562#TMF_PolicyRegion::GUI#
TEC31Region     1515280903.1.673#TMF_PolicyRegion::GUI#
TEST-tivdev     1482082604.1.641#TMF_PolicyRegion::GUI#
Tivoli/Sentry Defaults-tivdev02-region  1482082604.1.596#TMF_PolicyRegion::GUI#
rh0255b.itsc.austin.ibm.com-region      1515280903.1.196#TMF_PolicyRegion::GUI#
rh0255c.itsc.austin.ibm.com-region      1360991896.1.196#TMF_PolicyRegion::GUI#
test-rh0255b    1515280903.1.536#TMF_PolicyRegion::GUI#
tivdev02-region 1482082604.1.195#TMF_PolicyRegion::GUI#

# wlookup -ar ManagedNode
k124a   1360991896.2.7#TMF_ManagedNode::Managed_Node#
rh0255a 1515280903.2.7#TMF_ManagedNode::Managed_Node#
rh0255b.itsc.austin.ibm.com     1515280903.1.327#TMF_ManagedNode::Managed_Node#
rh0255c.itsc.austin.ibm.com     1360991896.1.327#TMF_ManagedNode::Managed_Node#
rh0255e 1360991896.3.7#TMF_ManagedNode::Managed_Node#
rh0255f 1515280903.3.7#TMF_ManagedNode::Managed_Node#
tivdev02        1482082604.1.326#TMF_ManagedNode::Managed_Node#

# wlookup -ar TaskLibrary
T/EC Tasks      1515280903.1.675#TMF_Task::TaskLibrary#
tl-tivdev02     1482082604.1.633#TMF_Task::TaskLibrary#
```

Note the relationship between the region numbers in the result of the `wlsconn` command and the output from the `wls` and `wlookup` commands. Here is a summary of interregion-related commands:

`wlsconn`	Displays a list of connected TMRs listed in the database.
`odadmin region`	Displays local and remote TMRs registered by the oserv.
`wls /Library/PolicyRegion`	Lists only local policy regions.
`wlookup -ar PolicyRegion`	Displays the local and remote policy regions.
`wlookup -ar ManagedNode`	Displays local and remote managed nodes.
`wlookup -ar TaskLibrary`	Displays local and remote task libraries.

Note that `wls` does not use the name registry and, therefore, only displays locally-defined objects. In contrast, `wlookup` reads the name registry and will return information about resources in any connected TMR as long as those resources have been updated (see also Section 7.3.3, “Updating Resources” on page 221).

It is important to try and give a good label name to resources that will be shared across TMRs. When you look at the Top Level policy regions (Desktop> TMR Connections> Top Level Policy Regions...), only the label names appear under each icon. This could become very confusing if label names are the same for policy regions and other resources in different TMRs.

This can be equally frustrating when looking at items in a selection list, for example, in the Set Resource Roles dialog for an administrator.

7.3.2 Connecting TMRs

A secure or remote TMR connection can be made from the desktop or with the `wconnect` command. The required role to connect TMRs is *super*.

A user must determine the following before making a TMR connection:

- The region name, region number, root or Administrator password, and interregion password for the remote TMR.
- Whether to use a one-way or two-way connection.
- Whether to use a secure or remote connection type.

Remote Connect means that you can initiate the entire connection from one machine. In this case, you must enter the password of the root administrator for the remote TMR server.

Figure 94 on page 219 details the Interregion Remote Connect dialog:

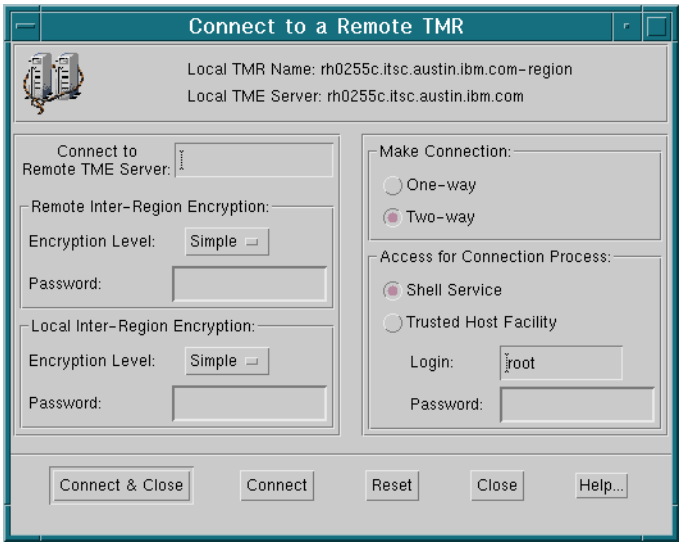


Figure 94. Interregion Remote Connect Dialog

Secure connect means that you must initiate the connection from both TMRs. It is secure because you do not have to enter the password of the root administrator in the remote TMR, and that root password is, therefore, not sent over the wire.

Figure 95 details the Interregion Secure Connect Dialog. Note that no trusted host login information is required in the dialog:

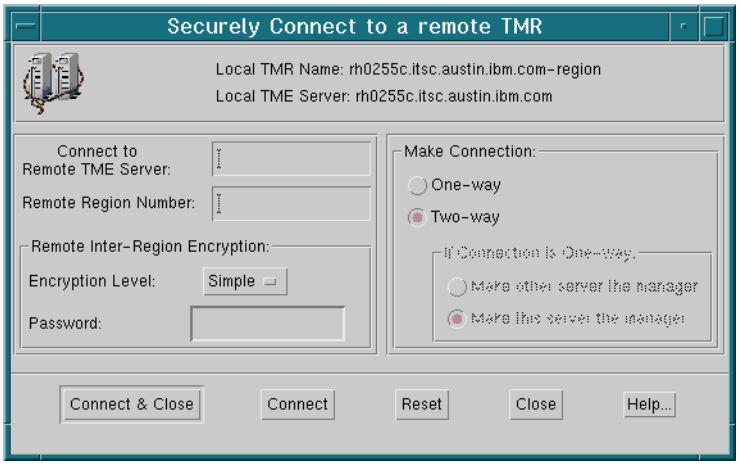


Figure 95. Interregion Secure Connect Dialog

7.3.2.1 Secure and Remote Connections

A secure connection does not require a remote login to connected TMRs. Connection requests are required by each communicating TMR. Each side will do the following:

- Add the remote TMR's host name, region number, and encryption level to the Interregion object (`wlookup InterRegion`). The interregion object keeps this data in an attribute called TMRs.
- Try to communicate with the remote TMR.
- If communication is successful, add TMR name to interregion object and exchange resources.

A remote connection allows one TMR to provide all of the connection request information and make the connection. Remote connections perform the same steps as a secure connection, but the `rexec` or `rcmd` is used to communicate to and start the connection in the remote TMR. To perform remote connections, the TMR making the connection must have the remote TMR server's root password or be a trusted host. Note, that for Windows NT, we cannot use trusted host, we must use TRIP with the shell service option. As it is possible to disable TRIP after the initial install, you may need to check that TRIP is installed and running to perform the TMR connection.

Note

Secure and remote are only methods of making a connection not different types of connection. The only different types are one-way and two-way.

7.3.2.2 One-Way and Two-Way Connections

One and two-way connections determine the visibility of interconnected TMRs resources.

A one-way connection requires one of the TMRs to act as the manager of the other TMRs resources. The managed TMR will not be able to view the managing TMRs resources.

Important

This type of connection can cause applications to fail. For example, Tivoli Distributed Monitoring can be set up so that monitors will change indicator collections on the other side of a connection. In a one-way connection scenario, the remote *managed* TMR where the monitor is running cannot see the local *managing* TMR's Indicator Collection resources.

A two-way connection allows each interconnected TMR to view all of the other's resources as long as they are updated regularly.

7.3.3 Updating Resources

Updating resources, sometimes referred to as a resource exchange, is performed manually from the Update Resources dialog (TMR Connections->Update Resources...) or with the `wupdate` command.

Updating resources is a *pull* operation. A TMR can request an update of its name registry, but it cannot *push* its current name registry resource to a remote TMR. There is one exception to this. When TMRs are first connected, the administrator is asked if resources should be updated immediately upon connection. The request is made on each TMR before the interconnect is completed.

Figure 96 details the Updated Resources from Multiple TMRs dialog.

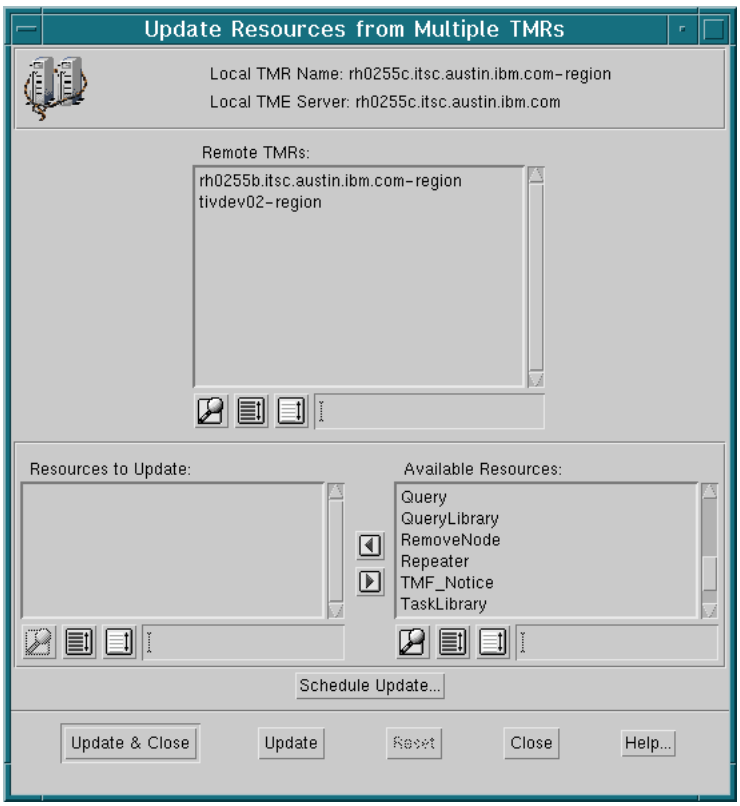


Figure 96. Update Resources from Multiple TMRs

Because resource information is updated between name registries at administrator-defined intervals, there are caching issues with viewing remote resources in the local name registry. New resources created in the remote TMR are not available in the name registry of a connected TMR until the next update is performed.

Important

Plan resource updates carefully. Not all updates in a TMR will affect the exchanged resources. Tivoli product manuals generally list which resources can and should be exchanged. See also Section 7.3.7, “Scheduling Updates” on page 230.

- If you create a new managed resource (something that requires the Create option on the menu bar of the desktop) then you will need to perform an update of resources between TMRs. To save time, you can select a single resource type to be updated using the TMR Connections > Update Resources... option, or the `wupdate` command.
- If you remove a resource, such as a file package or a monitor, then an update of resources would also be required.
- If you modify an existing resource, for example, update a task or change the attributes of an administrator, you do not require an update of resources. The name registry on connected TMRs only contains a reference to instance labels and their object IDs not the contents of the objects.

7.3.4 Resource Visibility

Visibility describes available resources in the name registry or in collections. A collection is an object (an administrator’s desktop, a policy region, and so on) that can hold a list of references to other objects in the system. When an administrator opens a collection object, the member objects are displayed. Once TMRs have been connected, member objects could be located in a remote TMR.

For example, a newly-created managed node object in TMR A can be visible in TMR B as soon as it is created if it resides within a collection path that has a point visible in TMR B. This is true even though the new managed node will not be visible in the managed node resource type in the name registry in TMR B until the next update of resources from TMR A.

Name Registry visibility applies to:

- Available lists, as in Available Subscribers in a Profile Manager window.

- Resources listed with the `wlookup` command.
- Resources referenced from the command line with `@Resource:instance`.

Collection Path visibility applies to:

- Resources viewed by opening collections on the desktop. Collections here refer to any kind of a container object that is visible from the GUI.
- Resources managed through the file system-type commands (`wls`, `wmv`, and so on).
- Resources referenced from the command line with `/xxx/yyy`.

The following example shows screens from TMRs rh0255b and rh0255c. A query has been created in TMR rh0255c. The name of the query is `FIND_NT`.

In TMR rh0255c, the query is in the local database confirmed with `wls` and is also in the name registry, as shown with `wlookup`. In this case, only the queries in the local database are in the name registry. Another way to see this with `wlookup` is that all the object IDs start with the TMR number of rh0255c:

```
#wls -l /Library/Query
1360991896.1.565#TMF_Query::Query#      Find-AIX
1360991896.1.566#TMF_Query::Query#      Q-AIX-Cmdline
1360991896.1.567#TMF_Query::Query#      Q-AIX-Cmd
1360991896.1.576#TMF_Query::Query#      FIND_NT
#wlookup -ar Query
FIND_NT 1360991896.1.576#TMF_Query::Query#
Find-AIX        1360991896.1.565#TMF_Query::Query#
Q-AIX-Cmd       1360991896.1.567#TMF_Query::Query#
Q-AIX-Cmdline   1360991896.1.566#TMF_Query::Query#
#wruninvquery -l FIND_NT
rh0255e
#
```

In TMR rh0255b, `wls` shows that the only query local to this TMR is `Get-AIX`. Using `wlookup`, the `FIND_NT` query has not been added to the local name registry because the resource type `Query` has not been exchanged since `FIND_NT` was created. Three other queries were previously exchanged with TMR rh0255c: `Find-AIX`, `Q-AIX-Cmd`, and `Q-AIX-Cmdline`. Therefore, the lookup for the `FIND_NT` query fails when you use `wruninvquery` from the command line:

```
# wls -l /Library/Query
1515280903.1.773#TMF_Query::Query#      Get-AIX
# wlookup -ar Query
Find-AIX        1360991896.1.565#TMF_Query::Query#
Get-AIX 1515280903.1.773#TMF_Query::Query#
Q-AIX-Cmd       1360991896.1.567#TMF_Query::Query#
Q-AIX-Cmdline   1360991896.1.566#TMF_Query::Query#
# wgetquery -f FIND_NT
An instance named "FIND_NT" of resource "Query" was not found.
# wruninvquery -l FIND_NT
An instance named "FIND_NT" of resource "Query" was not found.
```

However, in TMR rh0255b, the query can be seen through the GUI by selecting **Desktop> TMR Connections> Top Level Policy Regions...**, opening the rh0255c-Queries PolicyRegion and then Q-test. The policy region containing the query library, Q-test, has been exchanged before, so the collection path is available to TMR rh0255b. If you run a distribution from the GUI and want to select from your subscribers using this query, it will run correctly. Figure 97 details a GUI query:

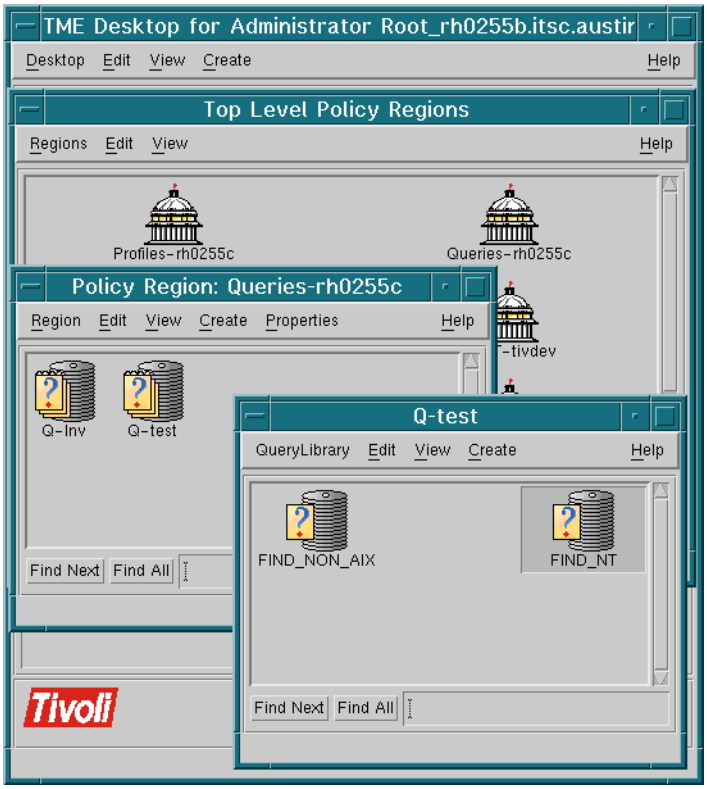


Figure 97. Remote TMR Can See Query in the GUI

In TMR rh0255b, you can select to update the resources from rh0255c. This can be achieved using the GUI **Desktop> TMR Connections> Update Resources...**, or by using the `wupdate` command. Now you can see the FIND_NT query listed in the output from `wlookup`, as detailed below. Using `wruninvquery` will also work.

```
# wupdate -r Query rh0255c.itsc.austin.ibm.com-region
#
# wlookup -ar Query
FIND_NT 1360991896.1.580#TMF_Query::Query#
Find-AIX        1360991896.1.565#TMF_Query::Query#
Get-AIX 1515280903.1.773#TMF_Query::Query#
Q-AIX-Cmd       1360991896.1.567#TMF_Query::Query#
Q-AIX-Cmdline   1360991896.1.566#TMF_Query::Query#
#
# wgetquery -f FIND_NT
Name:           FIND_NT
Description:    Find machines with Windows_NT
RDBMS User:     inventory
View:           INVENTORYDATA
Fields:
        TME_OBJECT_ID
        TME_OBJECT_LABEL

Where Clause:
--------------------
   (BOOTED_OS_NAME = 'Windows NT')#
# wruninvquery -l FIND_NT
rh0255e
#
```

7.3.5 Interregion Updates and Object Time Stamps

Each resource type in the name registry carries a time stamp. This time stamp is updated when:

- A new resource instance is added.
- An existing resource instance's information changes.
- A resource instance is removed.

Each interregion object has a per-TMR/per-resource time stamp for the last time it received an update from a resource in a connected TMR. This time stamp is used to determine whether or not an update of a remote resource type is necessary.

For example, in TMR B, the resource type ManagedNode has a time stamp. In connected TMR A, an interregion object keeps a time stamp that records the last time the resource type ManagedNode was updated in TMR B. If an administrator in TMR A requests an update of the ManagedNode resource type from TMR B, the interregion object in TMR A first checks the local time stamp against the time stamp in the name registry of TMR B. If the local time stamp is not older than the remote one, the resource is assumed to have not changed, and no exchange of data takes place.

If the interregion time stamp in TMR A is older than the one in the name registry in TMR B, then the entire resource is updated. This means that all of the instances of the resource ManagedNode from TMR B are brought into TMR A and merged into the ManagedNode resource in the name registry in TMR A.

A user can force an update of resources, regardless of the time stamp, with the `wupdate -f` option.

In 3.6, you can now see the date and time stamp when a resource was last exchanged with another TMR. The command to use is `wlsconn remote-regionname`.

In Figure 98, we look at a partial listing of the TMR itso3 resources exchanged with TMR `rh2900a`. On `rh2900a`, we looked at the resources prior to issuing the `wupdate` command. Notice the date and time stamp on the Administrator resource of `11/19/98 07:11:55`:

```
rh2900a: wlsconn Guyincharge

Name: Guyincharge
Server:     itso3
Region:     1295714281
  Mode:     two_way
  Port:     94

Resource Name           Last Exchange
-------------           -------------
TMF_Notice              11/09/97 03:22:34
Administrator           11/19/98 07:11:55
PolicyRegion            11/16/98 04:24:17
TaskLibrary             11/16/98 04:46:08
```

Figure 98. Partial Listing of wlsconn

We then issued the `wupdate -r Administrator Guyincharge` command to update the Administrator resource. We had previously added two Tivoli Administrators in the `Guyincharge` region on itso3. Reissuing the `wlsconn Guyincharge` command gives the result in Figure 99 on page 228.

```
rh2900a: wlsconn Guyincharge

Name:   Guyincharge
Server:     itso3
Region:     1295714281
  Mode:     two_way
  Port:     94

Resource Name           Last Exchange
-------------           -------------
TMF_Notice              11/09/97 03:22:34
Administrator           12/16/98 07:08:44
PolicyRegion            11/16/98 04:24:17
TaskLibrary             11/16/98 04:46:08
```

Figure 99. wlsconn After Updating the Administrator Resource

Note the time and date stamp on the Administrators resource has changed to `12/16/98 07:08:44`, the date and time the `wupdate` command completed.

7.3.6 Resource Flags

Each resource in the name registry carries a set of flags that affect how it is exchanged. A resource can be *exchangeable*, *non-exchangeable*, or *custom*:

- **exchangeable**

  An exchangeable resource can be updated between TMRs. When an instance of an exchangeable resource is created in TMR A, and the resource type is updated in connected TMR B, the newly-created resource from TMR A becomes visible in the name registry of TMR B. exchangeable is the default for all resource types.

- **non-exchangeable**

  A non-exchangeable resource cannot be updated between TMRs. This is for resources that need only be visible within a single TMR. Examples of non-exchangeable resources in the Tivoli Framework are:

  - Distinguished - use `wlookup -a` to display distinguished resources.
  - Classes.
  - Presentation objects.
  - ActiveDesktopList objects.

- **custom**

  A custom resource defines its own methods when exchanged between TMRs. If a resource's custom flag is set, all instances of the resource receive a callback when updated.

  The resource that is implemented needs to support the InterRegionUpdate interface defined in the TMF_Application.idl file by providing an implementation for the update resource operation. This is documented in the *Tivoli Advanced Development Environment* manuals.

  There are three custom resource types shipped with the Tivoli Framework:

  - TaskRepository
  - AdministratorCollection
  - TopLevelPolicyRegion

Figure 100 is a script that shows whether resources are exchangeable, non-exchangeable, or custom in your environment:

```
#!/bin/sh -e

TNR=‘wlookup NameRegistry‘

for resource in ‘wlookup -R‘
do
    exchangable=‘idlcall $TNR TMF_TNR::Resource::get ’"’$resource’"’ |
awk
’{print $4}’‘
    if [ $exchangable = 0 ]
    then
         echo "$resource\t\tnon-exchangable"
    elif [ $exchangable = 1 ]
    then
         echo "$resource\t\texchangable"
    elif [ $exchangable = 2 ]
    then
        echo "$resource\t\tcustom"
    elif [ $exchangable = 3 ]
    then
        echo "$resource\t\texchangable,custom"
    fi
done
exit 0
```

Figure 100. Resource Exchangeable Status Script

7.3.7 Scheduling Updates

Updating resources can be a fairly expensive process. The update runs as a single transaction. The update method obtains a write lock in the local name registry for any resource types that are updated and a read lock for resources in remote TMR(s). If the update method runs for any significant amount of time, other methods are locked out of portions of the name registry until it completes. In remote TMRs, where only read locks are held, this may not be a problem. However, in the local TMR, where write locks are held, even lookups are blocked. This can cause severe performance problems.

7.3.7.1 Rules for Updating Resources

- Do not schedule updates too often. Once a TMR deployment reaches a steady state, there is no reason to update resources more than twice a day. For example, once a day, during non-peak hours is preferable. An immediate update of a single resource type can be forced with the `wupdate` command.
- Do not schedule updates in interconnected TMRs at the same time. Stagger scheduled updates at least 30 minutes apart.

7.3.8 Disconnecting TMRs

TMRs can be disconnected from the desktop or with the `wdisconn` command.

When TMRs are disconnected, all resources in the name registry from the remote TMR are automatically removed. However, be aware that collection/collection-member inter-object references and subscriber/subscribee relationships that cross the region boundaries are not cleaned up. Managed nodes subscribed to profile managers in remote TMRs is one example. These references must be cleaned up with the `wchkdb -ux` command.

The disconnect is secure if one of the TMRs is not available. If one of the TMRs is not available, you should use `wdisconn -s`.

7.3.9 Troubleshooting TMR Connections

Some useful commands used for Troubleshooting connection problems are:

`wdisconn <region-name>`

Disconnect a TMR. You can specify a region name, or use the `-r` option to specify the region number. Using the desktop TMR Connections> Disconnect... you can only select a TMR by name.

`wdisconn -s <region-name>`
: Disconnects only one side of the TMR connection. The `-r` option can be used for the region number if required.

`wlsconn`
: Lists the current connections. The desktop option is TMR Connections> List Connections....

`wupdate`
: Exchanges resources between TMRs. Select **TMR Connections> Update Resources...** from the desktop.

`odadmin region`
: List all the regions recognized as currently available by the oserv. This includes the local region and is only available from the command line. Useful options are:

  `add_region`
  : Allows the addition of a region - useful when connections have only been partially made and a failure occurred.

  `delete_region`
  : Another useful option to delete a region to clean up partially-completed or failed connections.

A TMRs name used for a connection is the same as the default policy region created during the installation of the TMR sever.

After disconnecting TMRs, you should always run a `wchkdb -ux` to clean up any invalid references.

7.3.10 Troubleshooting Interconnected TMRs

This section contains advice on problems you may encounter when managing interconnected TMRs. Many of these problems are easier to explain with reference to a diagram; so, the following problems will refer to Figure 101 on page 232.

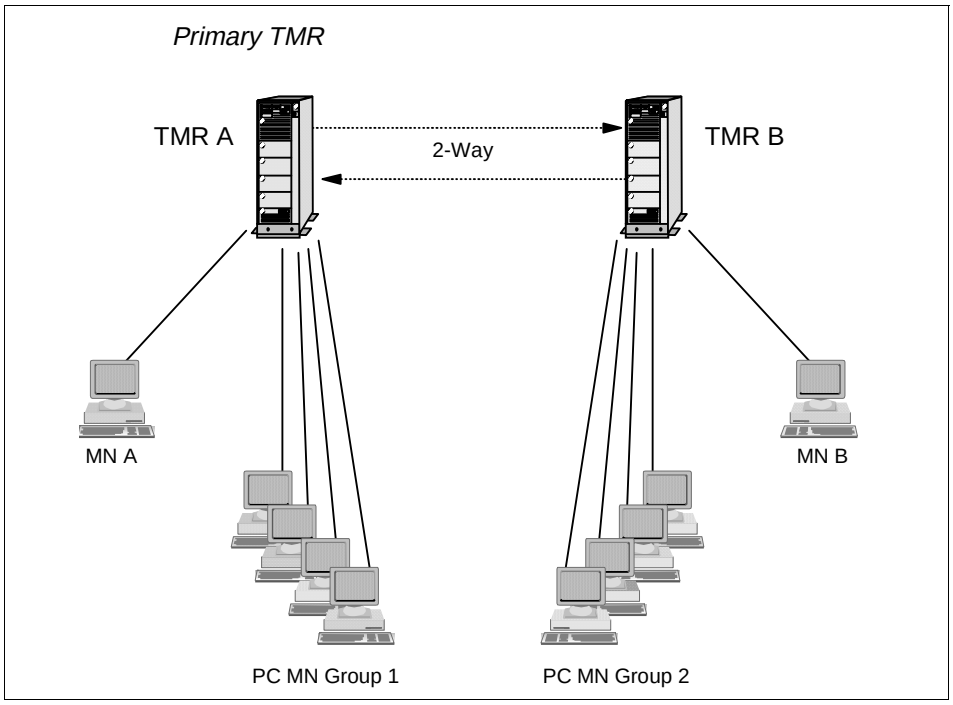


Figure 101. Two-Way Connected TMRs

7.3.10.1 Unable to Connect Previously-Connected TMRs

Issue the `wlsconn` command on TMR A to see if there is still a connection to TMR B. Login to TMR B's server and do the same. If only TMR B shows that the connection is active, issue the `wdisconn` command with the `-s` flag from TMR B.

```
# wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255c.itsc.austin.ibm.com-region rh0255c.itsc.austin.ibm.com 1360991896
  <----> tivdev02-region  tivdev02.itsc.austin.ibm.com 1482082604
# wdisconn -s tivdev02-region
# wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255c.itsc.austin.ibm.com-region rh0255c.itsc.austin.ibm.com 1360991896
# wchkdb -ux
```

Figure 102. Using the wdisconn Command to Disconnect TMRs

It is advisable to run a `wchkdb -ux -o <filename>` after the disconnect has completed. The `-o` stores references to problem objects in a file. Subsequent

checks can then just use that file as input rather than rechecking every object.

If errors occurred during the check, you can try taking some corrective actions before running `wchkdb` again. This time, however, replace `-o` with `-f`.This speeds up the check by just looking at the objects listed in filename. The next `wchkdb` will show if the errors persist. There may be occasions where simply running the check more than once is enough to resolve the errors. This is the format of `wchkdb` to use when specifying an input filename:

```
wchkdb -ux -f <filename>
```

7.3.10.2 Unable to Disconnect a TMR

If you are unable to disconnect a TMR, either both or just one TMR will show the connection.

Both TMRs show a connection:

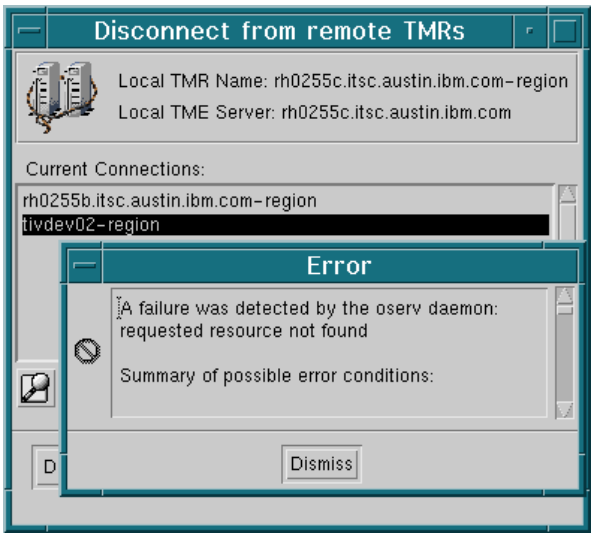


Figure 103. TMR Disconnect Failed

If disconnecting TMRs fails, check that both TMRs are showing a connection by using either the desktop TMR Connections> List Connections... or the `wlsconn` command.

If TMR B does not have the TMR A region listed in the output from the `odadmin region` command, use the following command on TMR B:

```
odadmin region add_region <region_#> <regionname> <oserv_port>
<encryption_level>
```

where:

`region_#`	The region number of TMR A.
`regionname`	The name of the TMR A region.
`oserv_port`	The port number that oserv is running on in TMR A (default is 94).
`encryption_level`	The level of encryption used for the TMR Connection. The default is simple.

You will also be prompted for a password, which should be the same encryption password that you used for the initial connection.

Once you have added the region manually, you can reissue the `wdisconn` command from TMR B.

```
# wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255b.itsc.austin.ibm.com-region rh0255b.itsc.austin.ibm.com 1515280903
  <----> tivdev02-region  tivdev02   1482082604

# odadmin region
Region  TME Srvr     ipaddr     port
1360991896      rh0255c.itsc.austin.ibm.com     9.3.1.235  94  1360991896.1.0
1515280903      rh0255b.itsc.austin.ibm.com     9.3.1.234  94  1515280903.1.0

# odadmin region add_region 1482082604 tivdev02.itsc.austin.ibm.com 94 simple
Remote region key:

# odadmin region
Region  TME Srvr     ipaddr     port
1360991896      rh0255c.itsc.austin.ibm.com     9.3.1.235  94  1360991896.1.0
1482082604      tivdev02.itsc.austin.ibm.com    9.3.1.134  94  1482082604.1.0
1515280903      rh0255b.itsc.austin.ibm.com     9.3.1.234  94  1515280903.1.0

#wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255b.itsc.austin.ibm.com-region rh0255b.itsc.austin.ibm.com 1515280903
  <----> tivdev02-region  tivdev02   1482082604

# wdisconn tivdev02-region

# wlsconn
     MODE NAME             SERVER         REGION
  <----> rh0255b.itsc.austin.ibm.com-region rh0255b.itsc.austin.ibm.com 1515280903

# odadmin region
Region  TME Srvr     ipaddr     port
1360991896      rh0255c.itsc.austin.ibm.com     9.3.1.235  94  1360991896.1.0
1515280903      rh0255b.itsc.austin.ibm.com     9.3.1.234  94  1515280903.1.0
#
```

The sample screen shows that:

1. `wlsconn`, the `tivdev02-region` is connected.
2. The oserv does not know about the resource when `odadmin region` is entered.
3. The `tivdev02-region` is added manually using `odadmin region add_region`.
4. `odadmin region` now agrees with the `wlsconn` command, `tivdev02-region` is connected, and oserv knows it.
5. `wdisconn tivdev02-region` command is issued to disconnect the TMRs.
6. `wlsconn` and `odadmin region` now agree that `tivdev02-region` has been disconnected.

Only one TMR shows a connection:
If only one side shows a connection, then you can try a one-sided disconnect from the TMR where the connection appears active, as shown below:

```
wdisconn -s region-name
```

7.3.10.3 Unable to See the Remote Resources from Your TMR

To ensure that both TMR servers reflect that there is a connection use `wlsconn` or (Desktop> TMR Connections> List Connections...).

Have the relevant resources been updated across the TMR connection since they were added/deleted in their *home* TMR?

Updates can be scheduled or done manually using the GUI (Desktop > TMR Connections > Update Resources...) or through the command line with `wupdate` (or `wupdate -f` to only update changed resources based on time stamps. See the *Tivoli Framework User's Guide* section on resource updates for details).

See also Section 7.3.3, “Updating Resources” on page 221.

7.3.10.4 Unable to Perform Actions on Remote Objects

Does your administrator user ID have the authority to operate on objects in policy regions in the remote TMR?

This can be achieved two ways:

1. Give the administrator the correct level of access across the policy regions in the remote TMR. Update the TMR Roles option when updating or creating an Administrator.
2. Select specific roles for selected resources using the Resource Roles option when creating or updating an administrator.

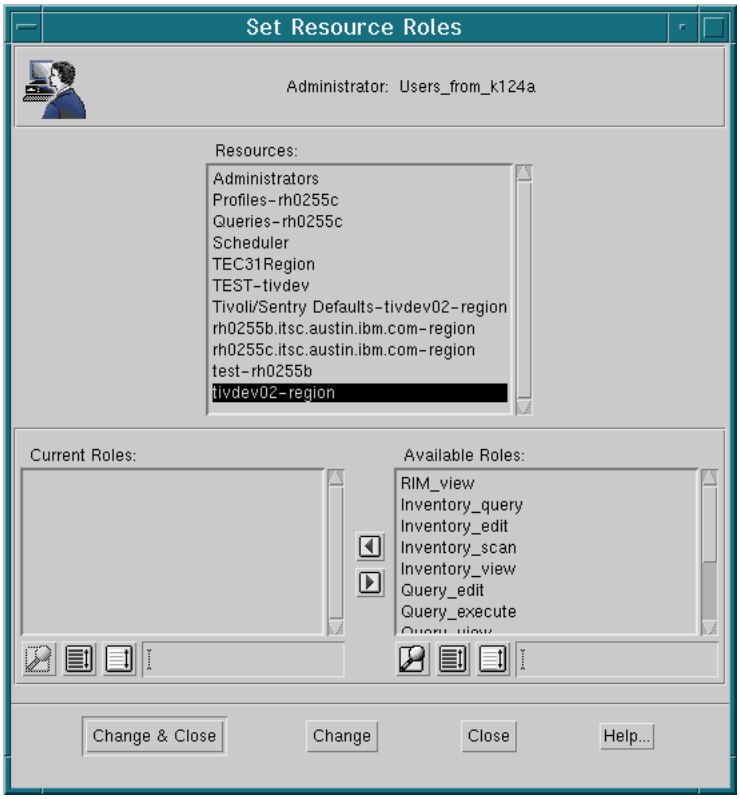


Figure 104. Update Resource Roles for an Administrator

You cannot update the roles for resources that you cannot manage yourself or update the resource roles of an administrator that has higher privileges.

7.3.10.5 Only One TMR Is Seeing the Shared Resources

- Remember that the exchange of resources is a *pull* operation. The TMR not seeing any remote resources has probably not issued a `wupdate` to the other TMR.
- If you have a one-way connection, only the *managing* TMR will see the remote resources. The **manager** option can be selected during a secure connection or the managing TMR will be the one from which the remote connect was started.

See also Section 7.3.3, “Updating Resources” on page 221.

7.3.10.6 An Application Fails across TMR Boundaries

- Check that you have a two-way connection. Many applications that run across a connection need to return information to the TMR from which they were started. If the process and information can only flow one way, the application may not function correctly.
- Check that the application is installed on both TMR A and TMR B plus all the managed and PC managed nodes. For instance, running an Inventory scan on TMR A for resources in TMR B requires that the product be installed on both TMR servers and all of the machines you wish to scan in both TMRs.

7.3.10.7 Installation Fails across TMR Connections

The installation of applications is not supported over TMR connections. Unfortunately, the standard installation process does not check this, and it will allow the installation to start.

Important

Do NOT attempt installing across TMR Connections; Tivoli may overwrite and make updates in either TMR. It could mean a complete recovery of both TMRs from backup is necessary, or if you do not have a complete backup of your machine and databases, a complete reinstall!

The potential for this situation is known by Tivoli, and they are working on a resolution.

7.3.10.8 Both TMRs in a One-Way Connection Are Set to Manager

Go to the TMR that should be managed and do a one-sided disconnect:

```
wdisconn -s <region-name>
```

Once the disconnect is complete, you can reissue the connection request with the correct settings.

7.4 Task Library

Today, it is a common requirement that you must execute an operation across the network. Examples include executing a binary file, making a backup, shutting down a process, and many others.

The Tivoli Framework provides the mechanism for the execution of processes through tasks and jobs.

Like all the resources in Tivoli, the tasks and jobs are grouped in another resource called the *task library*. All these resources are available once you have installed the Framework.

7.4.1 Tivoli Tasks

A task in Tivoli:

- Is the definition of the network operation that has to be executed.
- Can be executed several times.
- Is stored in a task library.

The definition of the task includes:

- Name of the task (Label).
- Platform in which the task will be executed.
- Tivoli role(s) required to execute the task.
- User and group under which the task will be executed.

Note

The required role defined in the task is not the role the administrator has in the policy region where the task library is but the one that you have assigned in the policy region in which the task endpoint resides.

For example, the administrator test has *admin* as the role in the policy region where the task library resides, but in the policy region in which the task will be executed, the administrator has *user*. Assuming this administrator has no TMR roles, the role required to execute the task must be user in order for this administrator to be able to execute the task in that policy region. If the administrator had a TMR role specified, then that role could be used as the required role for the task.

7.4.2 Tivoli Jobs

A job is a task, but it already has a number of run-time parameters set including:

- The list of targets in which the task will be executed.
- The execution mode - Serial, one target at a time or parallel, all subscribers simultaneously.
- The output format - Desktop or saved to a file.

The principal difference between a task and a job is that the job has sufficient data associated with it to be scheduled, and the task does not.

Therefore, if you want to execute the operation several times at predetermined intervals:

1. Define the task.
2. Define the job.
3. Schedule the execution of the job.

7.4.3 Task Library Features

Tasks libraries store binary files or scripts that we generally refer to as *executables*. When you create a task, an image of the executable that you have specified to run will be stored in the TMR server's binary tree.

> **Note**
>
> If you modify the executable used in a task, you will have to re-specify the executable in the task definition so that it can be refreshed in the TMR server binary tree.

You can pass arguments to a task, but by default, you can only do this executing the task from the command line. If you want to pass any argument using the GUI environment, you will need to use the Task Library Language (TLL). You will need to use `wtll` to export task library definitions into a flat text format. You can then modify the TLL and use `wtll` to import the definition again.

The task library has default and validation policies:

- Default task library policies set the available options of endpoints and profile managers to run the task or job.
- Validation task library policies validate the creation and execution of the task or job in a task library

  All these policies can be customized using a shell script to set or validate data. This customization will enable you to set options to determine which users you can run as well as validating users.

7.4.4 Task Library Survival Guide

All of the following actions can be executed by the desktop or by command line.

7.4.4.1 Creating a Task Library

When creating a task library, you will need to follow these steps:

1. You will need the *senior* role to create a task library.
2. Select the policy region where the task library will reside.
3. From the menu, select the option of **Create** and then **Task Library**. If the resource does not appear, you will have to assign this resource to the policy region (select the **policy region**, press the right button of the mouse, and then select **Managed Resources**, select the **Task Library** resource and move it to the current resources list).
4. Set the label of the task library.
5. Press **Create and Close**.
6. Now, you can create any task and job in the task library.
7. Or, at the command line, type `wcrttlib library_name pr_name`.

7.4.4.2 Creating a Task

Refer to the following steps when creating a task:

1. You will need *admin* role to create a task.
2. Select the task library where the task will reside and double click this icon.
3. From the menu, select **Create** and select the **Task** option.
4. Fill in the name of the task, the platform, the role or roles required, and the user and group to run the task.

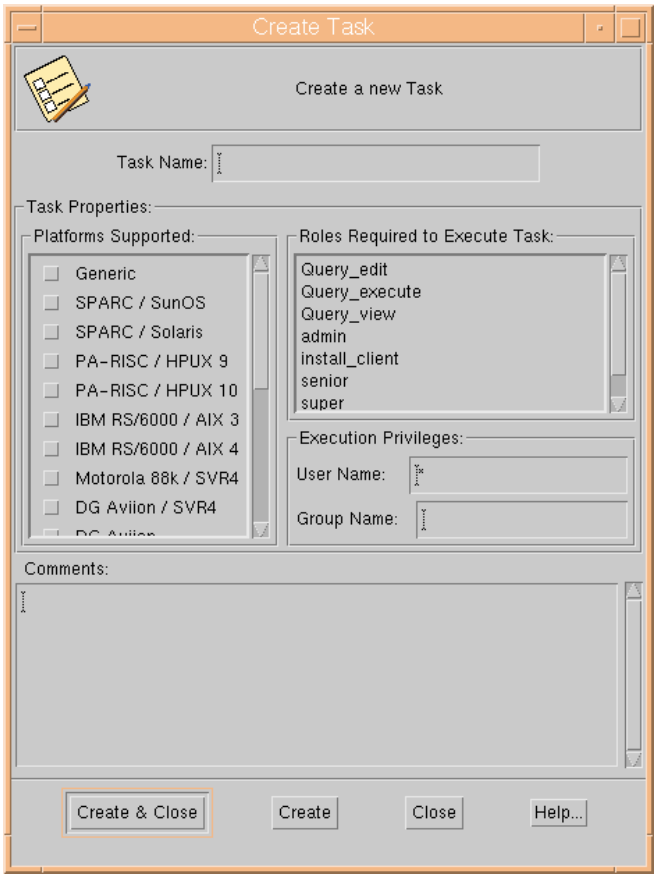


Figure 105. Creating a Task

5. Optionally, at the command line type: `wcrttask -t task_name -l lib_name [-g group_name] [-u user_name] -r role [-c comments] {-i interp_type mannode_name filename}...`

Note

If you specify asterisk (*) in the Execution Privileges, then the task will run using the user ID (or mapped ID) that the person who executes the task logged in with. This ID must exist in the endpoint where the task is to run.

7.4.4.3 Creating a Job

Follow these steps when creating a job:

1. You will need *admin* role to create a job.

2. Select the task library where the job will reside and double-click this icon.
3. From the menu select **Create** and select the **Job** option.
4. Fill in the name of the job.
5. Select the **Task Name**.
6. The execution can be:

   Parallel Will execute the job for all the endpoints at the same time.

   Serial Will execute one at a time.

   Staged Will execute in blocks of n endpoints. For example, in blocks of five endpoints with an interval of ten seconds.
7. Select the parameters of time-out (in seconds), and if you selected the staged option, you will have to fill in the staging count and the interval of time between them.
8. Select the **task endpoints** or **profile managers**.
9. Select **Create and Close**.

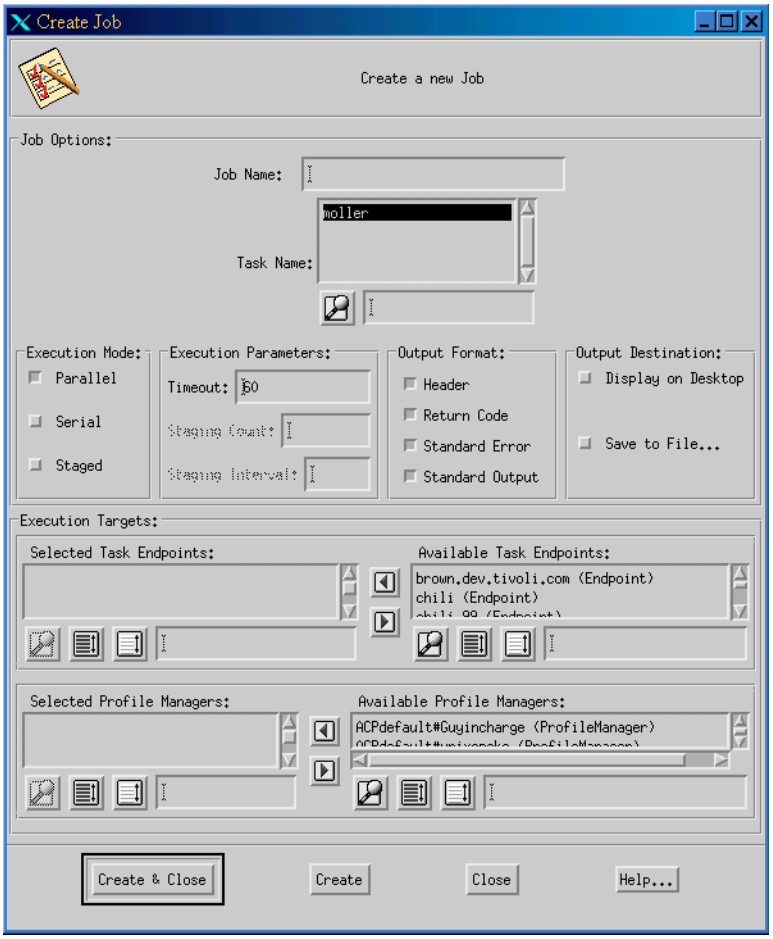


Figure 106. Creating a Job

Or at the command line type: `wcrtjob -j job_name -l library_name -t task_name -M mode [-s interval -n number] -m time-out -o output_format [-D | -d mannode_name -f file_name][-h mannode_name][-p prof_manager_name]`

Note

The only difference between executing a task and creating a job is that the job will save all the information so that it can be used easily by double-clicking on the icon. A job can also be scheduled as it has all the information included that is needed for it to run.

7.4.4.4 Executing Tasks and Jobs

1. Select the task or job you want to execute. Double-click the icon of the **task** or **job**.
2. If you select a task, you will have to fill in the execution mode, the platform, the role or roles, and the user and group to run the task. Then select the option of **Execute & Dismiss**.
3. If you double-click a job, it will be executed.
4. You can the see if the execution of the task or job was successful in the desktop or in the log file you specify.

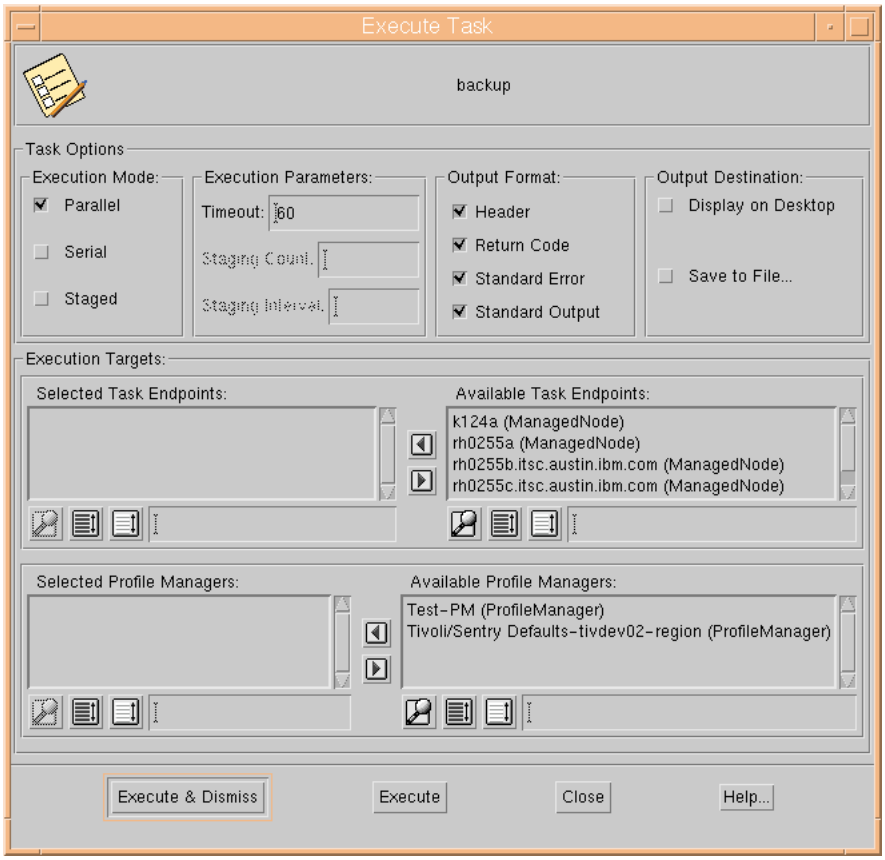


Figure 107. Executing a Task

Or at the command line: `wruntask -t task_name -l library_name {-h node...| -p profile_mgr...} [-a argument] [-e name=value] [iE] [-T transtype] [-M mode [-s interval -n number]][-m time-out][-o output_format]`

7.4.5 Task and Job Internals

This section gives a little more detail about the task creation, distribution process, and the use of default and validation policies.

7.4.5.1 Creating a task

When you create a task, the following occurs behind the scenes:

- The host that was specified in the Executable for Task window is contacted, and the task is copied to the TMR server. The directory that is used is

  $BINDIR/../interp/TAS/TASK_LIBRARY/bin/regionnumber/

  The name of the task is:

  tasklibraryname_machinegenerated

> **Note**
>
> The name of the task is now stored differently than pre 3.6. Previously, every time a task was edited and modified, the previous copies were kept in the directory. The same task was appended with a version number starting at 0 for the original task. Framework Version 3.6 now only keeps a single copy of the task.

7.4.5.2 Distribution of the Task Executables

Every time you execute a task, the server distributes to the targets:

- Executable files to run
- Access control list for the task
- Arguments for the task
- Environmental variables
- User and group name required
- Time-out value

The Tivoli method that is invoked is the *run_task* method. At the target, this method decrypts the task information, gets the correct executable for that interpreter type, and forks/execs the executable to perform the task. When the task is done executing, the distributed task is removed, and the output is collected and returned to the caller.

Framework 3.6 now allows tasks to be run on PC managed node or TMA endpoints. In the case of a PC managed node, the TMR distributes the task to the temporary directory, and the run_task method on the TMR will contact the

home host of the target PC managed node and will spawn the task through the PC Agent.

For TMA endpoints, the task is sent to the TMA endpoint. The TMR method run_task then spawns the run_task method on the gateway. The gateway will then cause a downcall of task_endpoint to the TMA endpoints, and the task is run returning any status to the TMR server.

Note

There is no extension associated with the task; therefore, a default extension is associated with the task on the endpoint when it is distributed. This extension depends upon the interp type being used. For generic and any of the UNIX interp types, the extension is sh. For Windows 95 and NT, the extension is cmd.

By default, the distribution of the binaries is only in the TMR server (ALI), but when you have Tivoli applications, like Distributed Monitoring, that use executables stored in the task library, it will be useful to distribute the binaries to the file servers either in the local TMR (LOCAL) or interconnected TMRs (GLOBAL).

Distributing the executable to all the file servers in a TMR gives the application faster access to the executable, and it will be more flexible and extensible.

Note

The distribution of tasks does not use MDist. The TMR Server will still use an IOM channel if data >16K, but the server makes a direct contact with each target. There is no use of mem_max or other tuning parameters.

You can distribute the task binaries with the command line:

```
wdisttask -q library_name
wdisttask -s library_name mode
wdisttask -d library_name task_name
```

Note

You cannot use the `wdisttask` to pre-stage tasks on target systems. These are useful only if an application, such as Distributed Monitoring, is running tasks. Running tasks or jobs from the TMR will still cause the tasks to be distributed to the target systems.

7.4.5.3 Task Library Policies

As we have said before, the default and validation policies are used in the task library.

Figure 6 provides a list of the different policies in a task library. First, the default policies:

Table 6. Default Policies in a Task Library

Default Policies	Description
tl_def_dist_mode	Default mode for distributing task binaries throughout a TMR. The default is **ALI**.
tl_def_man_nodes	Default list of managed nodes, as displayed in the Execute Task and Create Job dialogs.
tl_def_prof_mgrs	Default list of profile managers, as displayed in the Execute Task and Create Job dialogs.
tl_def_set_gid	Default group ID - This is an actual ID not GID
tl_def_set_uid	Default user ID - This is an actual ID not UID.

Figure 7 provides the validation policies:

Table 7. Validation Policies in a Task Library

Validation Policies	Description
tl_val_dist_mode	Validates the endpoints on which a task or job will run.
tl_val_prof_mgrs	Validates the profile managers on which a task or job will run.
tl_val_set_gid	Validates the assigned group name of a task or job (Uses the actual name, not GID)
tl_val_set_uid	Validates the assigned user name of a task or job (Uses the actual name, not UID).

You can look at the policies with:

```
wlspol [ -d | -v ] TaskLibrary
```

Use `wlspol` to list the names of the policy default objects, such as `BasicTaskLibrary`.

```
wlspolm [ -d | -v ] TaskLibrary
```

Use `wlspolm` to list policy methods assigned to the `TaskLibrary` resource. It will display a list, such as the ones shown in the tables above.

```
wgetpolm [ -d | -v ] TaskLibrary BasicTaskLibrary {policy}
```

Use `wgetpolm` to list the body or constant value of a default or validation policy method.

```
wputpolm [ -d | -v ] TaskLibrary BasicTaskLibrary {policy}
< binary of the new policy>
```

Use `wputpolm` to replace a policy method's body.

For more details about these commands, consult the *Tivoli Framework Reference Manual*.

7.4.6 Task Library Commands

The following table summarizes the task library commands:

Table 8. Task Library Commands

Command	Purpose	Role
`wcrtjob`	Creates a new job in a task library	Admin, Senior, Super
`wcrttlib`	Creates a new task library	Admin, Senior, Super
`wcrttask`	Creates a new task in a task library	Admin, Senior, Super
`wdeljob`	Deletes a job from a task library	Admin, Senior, Super
`wdeltask`	Deletes a task from a task library	Admin, Senior, Super
`wdisttask`	Controls the distribution of task binaries fro a task library	
`wgetjob`	Lists information about a job	User, Admin, Senior, Super
`wgettask`	Lists information about a task	User, Senior, Super
`wlstlib`	Lists information about a task library	User, Senior, Super
`wrunjob`	Executes a job	The role specified in the job definition

Command	Purpose	Role
wruntask	Executes a task	The role specified in the task definition
wsetjob	Sets the properties of a task	Admin, Senior, Super
wsettask	Sets the properties of a task	Admin, Senior, Super
wtaskabort	Aborts a task transaction and rolls back any uncommitted changes	Can only be used in a script and does not work by command line
wtll	Imports and exports task library definitions	User, Admin, Senior, Super

7.4.7 Troubleshooting Tasks and Jobs

This is a collection of tips that may be useful in investigating problems with tasks:

- Be sure that your script or binary file is working as it should be, independent of the task/job process.
- You can use Tivoli to open an `xterm` on the machine where the problem task is running. You can use:

  ```
  wxterm -h ManagedNode -display mydesktop:0
  ```

  Not only does this give you the `xterm`, but it also confirms that remote initiation of programs is possible. The `xterm` will run with the same environment as the task.
- Remember, that normally the first line of a task script must be `#!/bin/sh`. However, if the task is to execute in a PC managed node or TMA endpoint, then this line must be omitted.
- If the task is created on a UNIX TMR, and the target systems are Windows 95 or NT, the lines of the task must end in `^M` (**Ctrl-M**).
- The binary directory of the tasks must be writable for creating new tasks. The path of this directory is:
  `/<install-dir>/<interpreter-type>/TAS/TASK_LIBRARY/regionnumber/bin`
- To change the policy region validation to allow *root* to run tasks, perform the following steps:

  1. `wcrtpol -v TaskLibrary new_name_for_library_copy`
  2. `wgetpolm -v TaskLibrary new_name_for_library_copy tl_val_set_uid >file_name`

3. Edit the `file_name` and remove the checks for root. Keep the part that says: `echo TRUE and exit 0`.
4. `wputpolm -v TaskLibrary new_name_for_library_copy tl_val_set_uid <filename`
5. If the desktop is running, exit the desktop and restart it.
6. Open the **policy region** window.
7. Under Properties/Managed Resource Policies, open the button for **Validation Policy** and when you see this new policy, select it.

- If you are using commands from the Framework, you MUST set up the environment variables in the script, as follows:

  UNIX: `/etc/Tivoli/setup_env.sh`

  NT: `c:/winnt/system32/drivers/etc/Tivoli/setup_env.cmd`

- List the task and jobs within a library. Use the command:

  `wlstlib library_name`

  Does the task associated with the job still exist?

- List the properties of a task:

  `wgettask [ -F  file_name ] task_name library_name`

  For example:

```
[root@itso3]/> wgettask endpoint TaskLibrary
Task Name               endpoint
User Name               *
Group Name
Task ACL                senior:super:user
Supported Platforms
        w32-ix86        <install-dir>/w32-ix86/TAS/TASK_LIBRARY/bin/1295714281/T
askLibrary__vhyadwba
Task Comments
        Task Name            : TaskLibrary/endpoint
        Task Created         : Thu Dec 17 18:05:13 1998
        Task Created By      : root@itso3
        Task Files
            w32-ix86         itso3           /tmp/task.txt
        Distribution Mode    : ALI
        Task Comments        :
------------------------------------------------------------
```

- List the properties of a job:

  `wgetjob job_name library_name`

  For example:

```
[root@itso3]/swdist/logs/task> wgetjob datejob TaskLibrary
Job Name             :  datejob
Task Name            :  leedate
Execution Mode       :  parallel
Timeout              :  60
Output Format        :  task header
                        return code
                        standard output
                        standard error output
                        save output to file
                                itso3
                                /swdist/logs/datejob.log

Managed Nodes        :
Profile Managers     :  UNIXPM (ProfileManager)
```

Note

If the target of a job is a profile manager, the subscribers of the profile manager are resolved when the task is run. So, if you originally create a job that has the profile manager UNIXPM as a target, the subscribers to the UNIXPM profile manager are retrieved EACH TIME the job is run.

- Exporting and Importing task library definitions:

  `wtll [ -F ] export_file -l library_name`

  `wtll [ -i ] [ -r ] -p policy_region [-P Preprocessor] import_file`

The following screen on page 251 shows a `wtll` export file:

```
TaskLibrary “task-rh0255b” {
        Context = (“_!_”,”*”,1);
        Distribute = (“_!_”,”ALI”,1);
        HelpMessage = (“_!_”,”Conventional Task Library”,1);
        Requires = (“_!_”,”>2.5”,1);
        Version = (“_!_”,”1.0”,1);

        ArgLayout Filename{
                TextChoice FileBrowser;
                ButtonLabel = (testmsg_BrowserButton);
        };

        Task backup {
                Description = (“_!_”,”Upgraded Task”,1);
                HelpMessage = (“_!_”,”No Help Available”,1);
                Uid = (“_!_”,”root”,1);
                Gid = (“_!_”,”bin”,1);
                Comments = (“_!_”,”Task Name            : task-rh0255b/backup
Task Created         : Thu Oct 16 14:49:31 1997
Task Created By      : root@rh0255b.itsc.austin.ibm.com
Task Files
    default          rh0255a          /usr/local/bin/backuptmr
Distribution Mode    : ALI
Task Comments        :

------------------------------------------------------------
“,1);
                Roles = (“_!_”,”user”,1);
                Argument (testmsg_ArgDirname){
                        Layout = “Filename”;
                        MustMatch = “^/.*”;
                };
                Implementation (“default”) Binary “0.default”;
        };

}
```

7.4.8 Task Library Common Errors

- `time out`

  The task exceeded the amount of time allowed in the time-out setting.

- `getpwname failed with code ##`

  You are trying to execute the task with a user that does not have an account on the destination managed node where it is trying to run the task.

  The user *MUST* exist before the task can be run.

- Getting `method fork failed` errors.

  This can be an OS resource problem, for example, swap space, lack of threads, and so on.

- `command exited with signal 5, core=false`

  This error can occur if you change any default or validation policy, and the script has an error.

  For example:

  You might want to change the `tl_def_dist_mode` to `LOCAL`. If you remake the script with the `echo` command, you will have this error because the `echo` adds a new line character, so you will have to use `printf` like this:

  ```
  #!/bin/sh
  printf LOCAL
  exit 0
  ```

- `'open' failed with code '13': 'Permission denied`

  This error occurs in tasks when the user ID running the task does not have permission to write to the log file. If you are receiving this on tasks, also have the output come to the desktop. If the desktop output is correct, but you are receiving the code 13, then check permission. On jobs, this error was seen under two circumstances. You did not have permission to write to the log file or the task specified in the job does not have a valid userid on the target system.

- `(14): no permission for 'TaskLibrary/rhondatask' for operation 'run_task'`

  The person running the job or task does not have the authority to run the task. Check the Task ACL with the `wgettask` command. Check the TMR roles for the administrator. Remember: Roles are not hierarchical. Add an appropriate role to either the administrator or to the task.

- `'COMMconnect_host' failed with code '79':'rh2900c'` or `pctmp109 (Endpoint): ipc_create_remote failed: unable to connect to 146.84.32.208+9494: (67) IPC shutdown`

  The target endpoint agent (TMA or PC) was not running on the endpoint.

- `(4): resource 'leedate' not found`

  The task `leedate` was deleted, but the job still references it. This error was received when running the job referencing the `leedate` task.

Note

If you are still having problems, you can do the following to gather more information about the errors:

1. Enable the `wtrace` of errors and objcalls.
2. Execute an `odstat`.
3. Regenerate the problem.
4. Execute an `odstat` and keep it in a file.
5. Execute `wtrace -jk $DBDIR` and keep it in another file.

See Chapter 6, “Commands and Logs for Troubleshooting” on page 131.

7.5 Scheduler

As its name suggests, the scheduler can be used to schedule jobs, backups and profile distributions. To begin to schedule a job, you simply drag and drop the icon of the job onto the icon of the scheduler.

7.5.1 Scheduler Commands

Table 9 is a quick summary of the scheduler commands:

Table 9. Scheduler Commands

Command	Purpose	Role
`wdelsched`	Removes jobs from the scheduler	Super, Senior, Admin
`wedsched`	Edits a job that currently exists in the scheduler	Super, Senior, Admin
`wenblsched`	Disable or enables scheduled jobs	Super, Senior, Admin
`wgetsched`	Retrieves information on jobs currently scheduled	Super, Senior, Admin, User
`wschedjob`	Schedules a job that exists in the task library	Super, Senior, Admin
`wstartsched`	Starts the TME10 scheduler	Super, Senior

Refer to the *Tivoli Framework Reference Manual* for more details.

7.5.2 Tips for Working with the Scheduler

Use the `wgetsched` command to make sure the job you are scheduling is really in the scheduler. Output from the command is shown in Figure 108:

```
[root@itso3]/swdist/logs/task> wgetsched
 Job ID    Job Label         Admin            Date & Time        Enbld Repeat Re
try Cancel
 ------ --------------- --------------- ----------------------- ----- ------ --
--- ------
 000001 update X-TMR re root@itso3      Thu Dec 17 20:30:00 1998  YES    YES
NO    NO
 000004 filepackage     root@itso3      Fri Dec 18 03:20:00 1998  YES    NO
NO    NO
 000008 going to be del Lee@itso3       Fri Dec 18 16:45:00 1998  YES    YES
NO    NO
```

Figure 108. Output of the wgetsched Command

Check for the name of the job, the date and time scheduled, and the Administrator.

Check the scheduler for entries that repeat. Do not delete a Tivoli Administrator that has jobs in the scheduler. Doing this will cause the job not to run. You need to use the `wgetsched` command and find all jobs that the administrator is running and get the scheduler ID number. Then issue the `wgetsched` command in verbose mode and get all information relative to the job so you can recreate the job once the administrator is deleted. Finally, you need to delete the job from the scheduler with the `wdelsched` command and re-add it using a different administrator. If you have already deleted the administrator, then every time the scheduled job is run, it will fail, and a notice will be logged.

```
[root@itso3]/swdist/logs/task> wgetsched
 Job ID    Job Label         Admin            Date & Time        Enbld Repeat Re
try Cancel
 ------ --------------- --------------- ----------------------- ----- ------ --
--- ------
 000001 update X-TMR re root@itso3      Thu Dec 17 21:00:00 1998  YES    YES
NO    NO
 000004 filepackage     root@itso3      Fri Dec 18 03:20:00 1998  YES    NO
NO    NO
 000008 going to be del Lee@itso3       Fri Dec 18 16:45:00 1998  YES    YES
NO    NO

[root@itso3]/swdist/logs/task> wgetsched -v -s 8
ID               : 8
Name             : rhondajob
Label            : going to be deleted
Description      : This is a job from the Task Library.
Administrator    : Lee@itso3
Original Time    : Fri Dec 18 16:45:00 1998
Next Time        : Fri Dec 18 16:45:00 1998
Enabled          : Yes
Repeat Type      : Finite
Repeat Increment : 5
Repeat Unit      : Minute
Repeat Times     : 5
Retry Type       : None
Retry Increment  : 0
Retry Unit       : Minute
Retry Times      : 0
Cancel Job       : No
Cancel Increment : 0
Cancel Unit      : Minute
Email            :
```

And so on

Figure 109. Example of a wgetsched Command with Verbose Output

For jobs that are regularly scheduled or repeat jobs, at least once a week run the `wgetsched` command in verbose mode and save their definitions. If the scheduler must be cleaned, you can use these definitions to recreate the scheduler jobs.

7.5.3 Troubleshooting Common Scheduler Errors

In some cases, when you try to obtain the list of jobs scheduled, you may obtain the error shown in Figure 110 on page 258. If so, you will need to start the scheduler again by typing the following command at the command line:

`wstartsched`

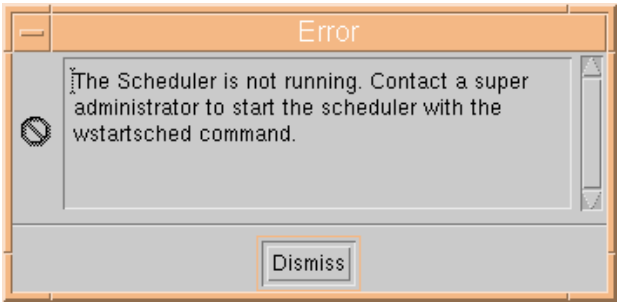


Figure 110. Scheduler Not Running Message

When you execute the command `wstartsched`, it may seem to start the scheduler, but when you try to retrieve the list of jobs, you still get the scheduler not running error. If this happens, the scheduler must be cleaned. Another indicator of this problem is the following in the oservlog:

```
Oct 22 17:01:34: ^hdaemon exit while in use: (0xa)
Oct 22 17:03:08: ^hdaemon exit while in use: (0x6)
```

7.5.3.1 Cleaning the Scheduler

You can clean up the scheduler by executing the following script in the TMR server:

```
#!/bin/sh
. /etc/Tivoli/setup_env.sh
index=0
SCHED=‘wlookup Scheduler‘
objcall $SCHED stop
set -e
NUM_CRED=‘objcall $SCHED contents| grep CredDatabase| wc -l‘
while $index -lt $NUM_CRED
do
        objcall $SCHED rmattr BDBPG:CredDatabase:$index
        index=‘expr $index + 1‘
        echo $index
done
index=0
NUM_SCHED=‘objcall $SCHED contents| grep SchedulerDatabase| wc -l‘
while $index -lt $NUM_SCHED
do
        objcall $SCHED rmattr BDBPG:SchedulerDatabase:$index
        index=‘expr $index + 1‘
done
wstartsched
```

7.6 Multiplexed Distribution and Bulk Data Transfer

Tivoli depends on the distribution of data to collections of managed nodes in order to implement the management by subscription model. In addition, there are many applications and services that utilize the transfer of data from one managed node to another or to many others. The following is a list of the main concepts we will describe here:

Multiplexed Distribution — This Framework service, usually known as Mdist, handles the simultaneous distribution of data from one managed node to one or many others through the use of repeaters.

Repeaters — Repeaters are responsible for forwarding data to a collection of one or more clients nominated as targets served by that repeater.

Bulk Data Transfer — Usually abbreviated to BDT, data transfers over 16 K in size (alterable from 3.2 onward) do not rely on the usual oserv to oserv communication over port 94. Instead, they use BDT, which sets up an Inter-Object Messaging channel.

Inter-Object Messaging — Known as IOM, this mechanism allows an objects to open up a direct connection to another object on a remote node for data transfer.

These are services that are provided by the Tivoli Framework.

7.6.1 Mdist

This function can be used whenever a managed node needs to transfer information to more than one endpoint. It is used for the distribution of many kinds of information, not only by products, such as Tivoli Software Distribution.

Mdist is designed to maximize data throughput by:

- Distributing in parallel to multiple systems.
- Sending a single copy of data to one or more designated hosts called repeaters, which redistribute it in parallel to other hosts.
- Spreading the distribution load across hosts.

Its usage is highly-dependent on how much data is to be distributed, how often it is to be distributed, and in what time frame. If a node invokes Mdist,

the data to be distributed will always be sent through a repeater. The basic steps for starting this process are:

- The managed node determines Mdist will be used because the distribution is to more than one node.
- Mdist contacts the TMR to request a distribution. A method called `obj_route` is called. This sends a list of target nodes to the repeater manager in the TMR server. The TMR server will return information to the Mdist node listing which (if any) repeater should be used to achieve the distribution.
- If the data is greater than 16 KB, Mdist passes a dkey (see “Bulk Data Transfer and Inter-Object Messaging” on page 271) to the TMR server for a node to come back to the Mdist system to read the data. The TME server contacts the Mdist node’s repeater and passes on the dkey.
- If the data is less than 16 KB, then the TMR server returns the repeater location to the Mdist node, which will then contact the repeater directly.
- If greater than 16 KB, the repeater contacts the Mdist node with the dkey and establishes an IOM channel to receive the data.

Note

UNIX - There are no explicit limitations to the number of active distributions able to be processed through a TCP/IP and Mdist repeater. 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, local disk buffering) and the surrounding network connections.

NT - Inbound maximum connections for an NT server is 10. Outbound is subject to the same conditions as UNIX.

7.6.2 Repeaters

Tivoli defines a repeater as a Tivoli managed node that can distribute data in parallel to one or more clients. A repeater can receive and distribute data or be the source and, therefore, only distribute it.

When a TMR server is created, it becomes:

- The *default repeater*. The TMR server will distribute to any client that is not in the range of another repeater where the range is a configurable listing of target managed nodes
- The *repeater manager* keeps the configuration of all the repeaters.

The number of repeaters needed in your enterprise will depend on a number of factors:

- Number of unique data packages.
- Complexity of each package.
- Frequency of distributions.
- Availability of targets.
- Network bandwidth.

There are some important considerations about what type of machine to designate as a repeater:

- Free space for the Mdist temporary files.
- Virtual memory. Each target may require up to 1.5 MB when you count memory required for the spawning of new processes, and so on.
- Enough TCP connections.

Each repeater has the following features, all of which can be tuned with the `wrpt` command using the parameter name given in parenthesis (see also “The wrpt Command” on page 262 and Section 10.4, “Repeaters and Networks” on page 344):

- **Maximum Memory (mem_max)**

  The maximum amount of memory the distribution consumes on this repeater before paging to disk. The default is 10,000 KB.

- **Maximum Disk (disk_max)**

  The maximum amount of disk space the distribution consumes on this repeater before halting incoming data. The default is 50,000 KB.

- **Working Directory (disk_dir)**

  The directory used for disk paging. This directory is also used for swap files on most operating systems. Multiple distributions performed simultaneously tend to fill this space causing the distributions to fail. If this is a heavily-used repeater for large file distributions, the working directory should be moved. The default is `/tmp`.

- **Maximum Simultaneous Connections (max_conn)**

  The maximum number of simultaneous parallel connections. If the number of targets is greater than the number of maximum simultaneous connections, and the size of the data distribution exceeds mem_max and disk_max combined, only the first max_conn targets will receive the distribution, and the rest will fail. The default is 100.

- **Operation Time-out (stat_intv)**

  The maximum amount of time to wait (in seconds) before aborting a nonresponsive network connection to a target. If a connection has been made, and no data is pulled during the time specified by the stat_intv variable, an `Operation Timeout Exceeded` message is displayed. The default is 180 seconds.

  Two similar situations are based on the operating system of the source and not based on Tivoli MDist configuration parameters:

  - If the source times out contacting the target when establishing contact, you get a `Dispatcher Unavailable` message.
  - If the target times out in trying to contact the source when establishing contact, you get a `High level TCP timeout` message. This `timeout` time is dependent on the operating system of the source machine and is not under the control of the Tivoli software.

- **Network Load (net_load)**

  The maximum amount of data (in KB/sec) that the repeater will put on the network for each distribution. The default is 500 KB/sec.

- **Network Spacing (net_spacing)**

  The amount of time (in milliseconds) to wait between each 16 KB write to the network. This variable is not displayed by `wrpt` query unless it has a non-zero value. The default is `0 ms`.

- **Disk Threshold (disk_hiwat)**

  The amount of disk usage (in KB) after which a delay occurs between new disk block allocations. The default is 50,000 KB. In general, this should match disk_max.

- **Disk Usage Rate (disk_time)**

  The delay (in seconds) to wait between disk block allocations after the disk threshold has been reached. The default is `1`.

> **Note**
>
> For the best performance the two most important tuning parameters in a repeater are net_load and max_conn.

7.6.2.1 The wrpt Command

The `wrpt` command is used to create, tune, and delete repeaters. See also Section 10.4, “Repeaters and Networks” on page 344 for more details on repeaters and repeater configuration.

Note

To execute the `wrpt` command you need *senior* role.

- `wrpt` - Lists all the repeaters.

In the example below, the TMR server is connected with another two TMR servers; so, each server is a default repeater.

The first output column contains the host names followed by the host number in square brackets `[]`. The flags in the second column can be:

w Indicates that the entry is a WAN entry site.

d Indicates the entry is the default repeater site.

The third column contains the range of hosts served by the repeater.

```
# wrpt
rh0255b.itsc.austin.ibm.com [1]  wd-  [default]
tivdev02 [1]          wd-  [default]
rh0255c.itsc.austin.ibm.com [1]  wd-  [default]
```

- `wrpt -t <repeater_name>` - Lists the current configuration parameters for a repeater.

In the example below, the first is an NT server, and the second is a UNIX server, all the features are the same, the only difference is the `disk_dir` configuration.

```
# wrpt -t tivdev02
mem_max      = 10000
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "C:/Tivoli/db/tivdev02.db/tmp/"
net_load     = 500
max_conn     = 100
stat_intv    = 180
# wrpt -t rh0255b.itsc.austin.ibm.com
mem_max      = 10000
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "/tmp"
net_load     = 500
max_conn     = 100
stat_intv    = 180
```

- `wrpt -n <repeater_name> <range=#,#,#>` - Creates a new repeater.

  The repeater name must be the same as the one used when you created the managed node. If you have interconnected TMRs, you can only create a repeater in the local TMR.

```
# odadmin odlist
Region      Disp  Flags  Port            IPaddr   Hostname(s)
1562489759     1    ct-    94         9.3.1.235   rh0255c.itsc.austin.ibm.com
               2    ct-    94       9.53.65.156   k124a.austin.ibm.com
               3    ct-    94         9.3.1.173   rh0255e.itsc.austin.ibm.com
1482082604     1    ct-    94         9.3.1.134   tivdev02.itsc.austin.ibm.com
1264987995     1    ct-    94         9.3.1.234   rh0255b.itsc.austin.ibm.com
               2    ct-    94         9.3.1.233   rh0255a.itsc.austin.ibm.com
# wrpt -n k124a range=3
# wrpt
rh0255b.itsc.austin.ibm.com [1]  wd-  [default]
tivdev02 [1]          wd-  [default]
rh0255c.itsc.austin.ibm.com [1]  wd-  [default]
k124a [2]             ---  [3]
```

- `wrpt -t <repeater_name> parameter <new_value>` - Tunes a repeater.

  You can tune all the repeaters including the ones that are in TMRs that are interconnected.

```
# wrpt -t tivdev02
mem_max      = 10000
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "C:/Tivoli/db/tivdev02.db/tmp/"
net_load     = 500
max_conn     = 100
stat_intv    = 180
# wrpt -t tivdev02 max_conn=20
# wrpt -t tivdev02
mem_max      = 10000
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "C:/Tivoli/db/tivdev02.db/tmp/"
net_load     = 500
max_conn     = 20
stat_intv    = 180
```

- `wrpt -L` - Lists the active distributions.
- `wrpt -k <id> -t <repeater_name> parameter <new_value>` - Tunes a repeater temporarily but only for the distribution already in progress.

```
# wrpt -L
    8 fp_distribute Oct 31 10:50:11       464/0 [80-464]
# wrpt -k 8 -t rh0255b.itsc.austin.ibm.com net_load=200
```

`wrpt -q <source> <target> [...<target>]` - Runs the tst_route method (similar to obj_route) and returns the distribution route to the specified targets.

```
# wrpt -q k2 aspen vail hood cook orac
--[RPT:k2 [2]]
   |--[RPT:orac [103]]
   |   |--aspen [46]
   |   |--vail [73]
   |
   |--hood [19]
   |--cook [41]
```

Here, k2 is a repeater to hood and cook and identifies orac as the repeater serving aspen and vail.

7.6.2.2 Repeater Examples

In the following examples, we use the simple environment shown in Figure 111 on page 266. We will show here the different possibilities involved in repeater configurations.

Note

Throughout these examples, you should note that any managed node that is not a repeater can only distribute data to one other managed node at a time. Multiple distributions must always go through a repeater.

TMA Endpoints do not act as repeaters or distribute data to other nodes.

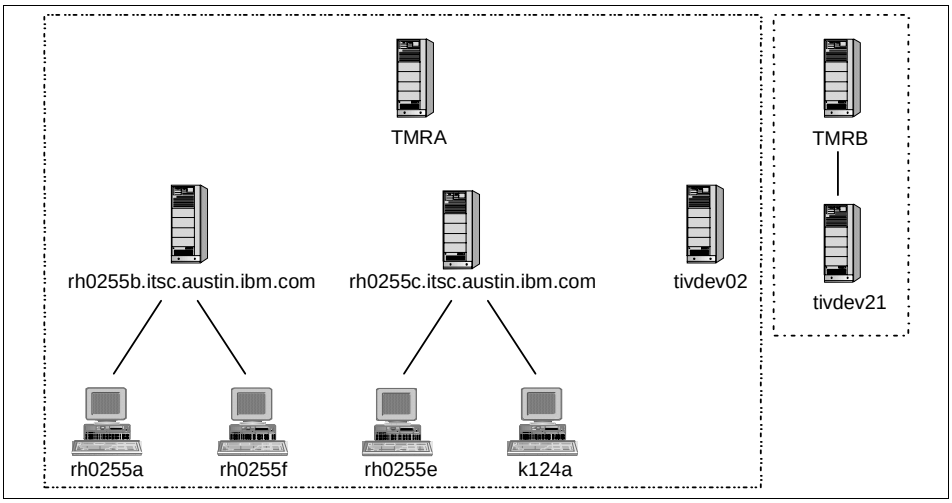


Figure 111. Repeater Example Environment

The systems rh0255b, rh0255c, and tivdev02 are all managed nodes and repeaters within TMR A (TMRA is the TMR server). The lines indicate the repeater relationship. For example, rh0255b is a repeater for rh0255a and rh0255f.

Point-to-Point

In point-to-point, the source node is not necessarily a repeater itself, and it is distributing to a single target within its own repeater range (see Figure 112). A query of the route would look like this:

```
# wrpt -q rh0255e k124a
--[RPT:rh0255e [3]]
   |--k124a [2]
```

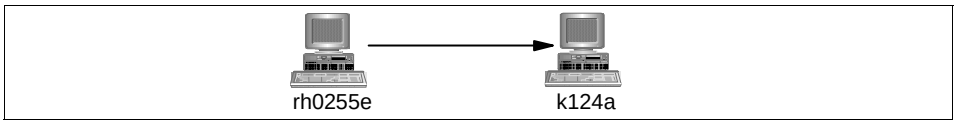


Figure 112. Point-to-Point Distribution

Here, rh0255c is a repeater that is distributing to one node, k124a.

Source repeater to many nodes

Here, the source machine is again a repeater, but this time it is distributing to many nodes (Figure 113 on page 267). The repeater rh0255b wants to distribute to rh0255a and rh0255c. We can see from the following routing

information that rh0255a and rh0255c are both targets for rh0255b, and rh0255b will distribute direct to them simultaneously:

```
# wrpt -q rh0255b rh0255a rh0255c
--[RPT:rh0255b.itsc.austin.ibm.com [6]]
   |--rh0255a [2]
   |--[RPT:rh0255c.itsc.austin.ibm.com [4]]
```

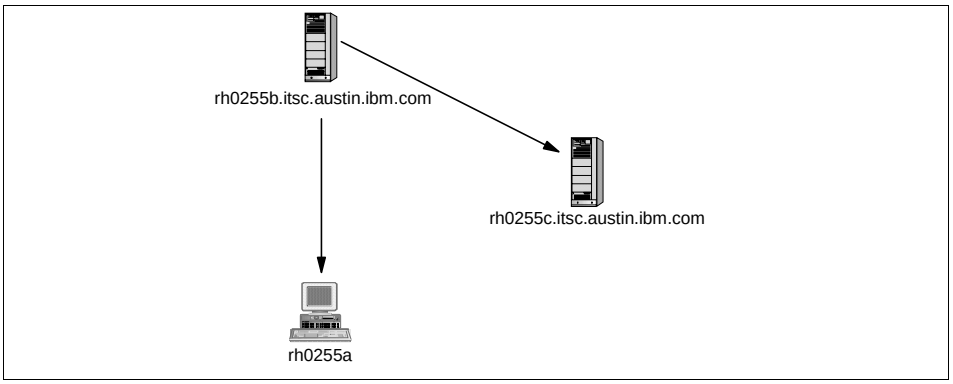


Figure 113. Repeater Source to Many Nodes Distribution

Source machine is not a repeater

In this scenario, the source machine rh0255a, that is not a repeater, wants to send a distribution to tivdev02 and rh0255c (Figure 114). The routing is returned as follows:

```
# wrpt -q rh0255a tivdev02 rh0255e
--[RPT:rh0255a [2]]
   |--[RPT:rh0255b.itsc.austin.ibm.com [6]]
   |   |--tivdev02 [5]
   |   |--[RPT:rh0255c.itsc.austin.ibm.com [4]]
```

The TMR server directs rh0255a to contact its repeater to perform the distribution. Then rh0255b can distribute to tivdev02 and rh0255c.

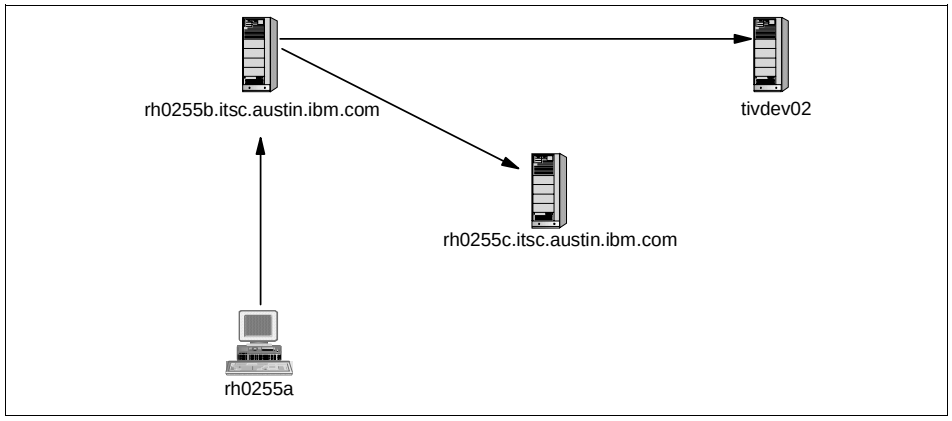


Figure 114. Non-Repeater Source to Many Nodes Distribution

Non-repeater to multiple targets all part of one repeater

Here, rh0255a wants to distribute to k124a and rh0255e. These nodes are both served by the same repeater, rh0255c. The routing is as follows:

```
# wrpt -q rh0255a k124a rh0255e
--[RPT:rh0255a [2]]
   |--[RPT:rh0255c.itsc.austin.ibm.com [4]]
   |   |--k124a [2]
   |   |--rh0255e [3]
```

This routing tells rh0255a to request rh0255c to repeat the data to the desired nodes (Figure 115).

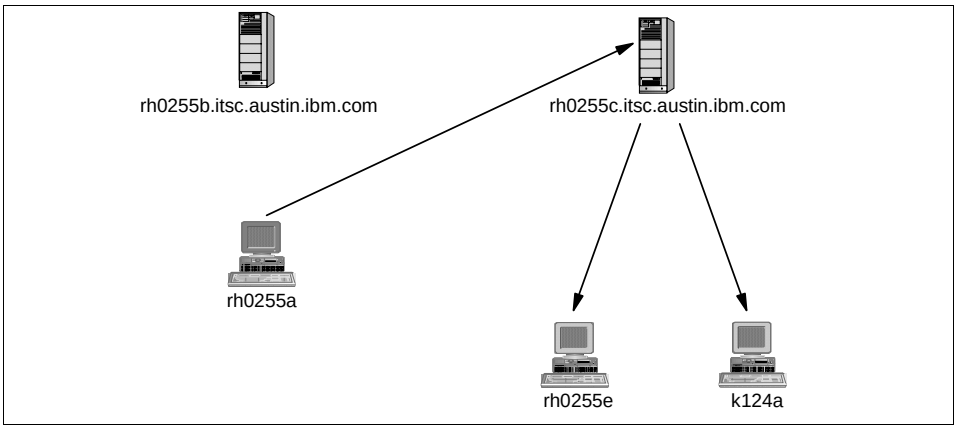


Figure 115. Non-Repeater Source to Many Nodes All Targets of One Repeater

Non-repeater to multiple targets not part of own repeater

Now, rh0255a wants to distribute to tivdev02, rh0255e, and k124a, the last two of which are not targets of rh0255c's repeater (rh0255b). The routing is as follows:

```
# wrpt -q rh0255a tivdev02 rh0255e k124a
--[RPT:rh0255a [2]]
   |--[RPT:rh0255b.itsc.austin.ibm.com [6]]
   |   |--tivdev02 [5]
   |   |--[RPT:rh0255c.itsc.austin.ibm.com [4]
   |   |   |--rh0255e [3]
   |   |   |--k124a [2]
```

This routing shows that rh0255a will use rh0255b as a repeater. Then, rh0255b will request rh0255c to perform the repeating function to rh0255e and k124a (Figure 116).

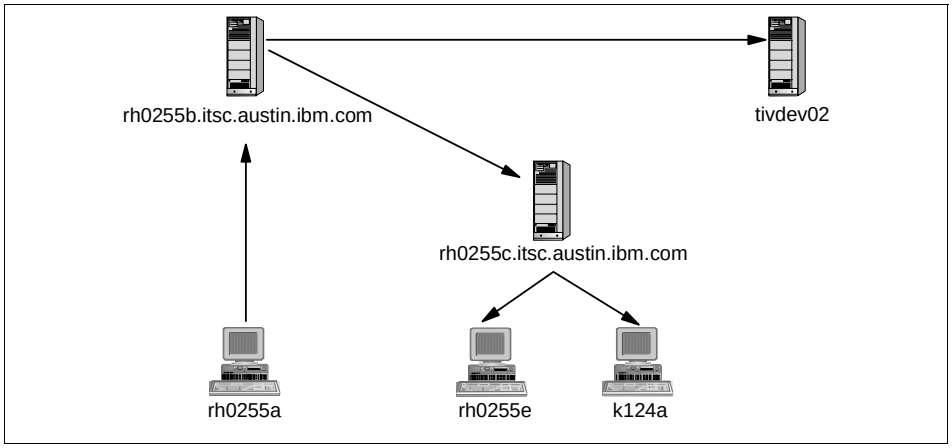


Figure 116. Non-Repeater Source to Many Nodes Not Targets of Own Repeater

Distribute to single target of one repeater and multiple of another

There is a special case when the data will not go through the designated repeater to reach a node. Suppose tivdev02 wanted to distribute to rh0255f, rh0255e, and k124a. The route will look like this:

```
# wrpt -q tivdev02 rh0255f rh0255e k124a
--[RPT:tivdev02 [5]]
   |--[RPT:rh0255b.itsc.austin.ibm.com [6]]
   |   |--rh0255f [7]
   |--[RPT:rh0255c.itsc.austin.ibm.com [4]
   |   |--rh0255e [3]
   |   |--k124a [2]
```

Because going through the repeater, rh0255b would just add an extra step to the distribution to rh0255f. tivdev02 will actually distribute directly to rh0255f (Figure 117 on page 270). If we were distributing to more than one node under rh0255b's jurisdiction, then the data would be repeated by rh0255b. So the `wrpt` output does not always exactly reflect what actually happens. You need to interpret it knowing how the repeater mechanism works.

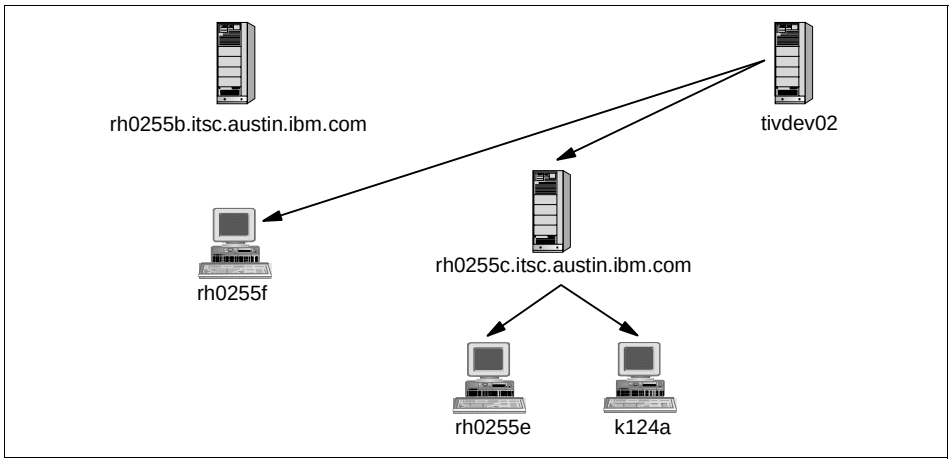


Figure 117. Distribute to Single Target of One Repeater and Multiple of Another

Non-repeater to hosts in another tmr

Here, rh0255a wishes to send data to tivdev21 and tivdev22 (Figure 118). Providing we have the correct TMR connections and resource updates, the repeater manager will return the following information for `wrpt`:

```
# wrpt -q rh0255a tivdev21 tivdev22
--[RPT:rh0255a [2]]
   |--[RPT:TMRB [1]]
   |   |--tivdev21 [3]
   |   |--tivdev22 [4]
   |
```

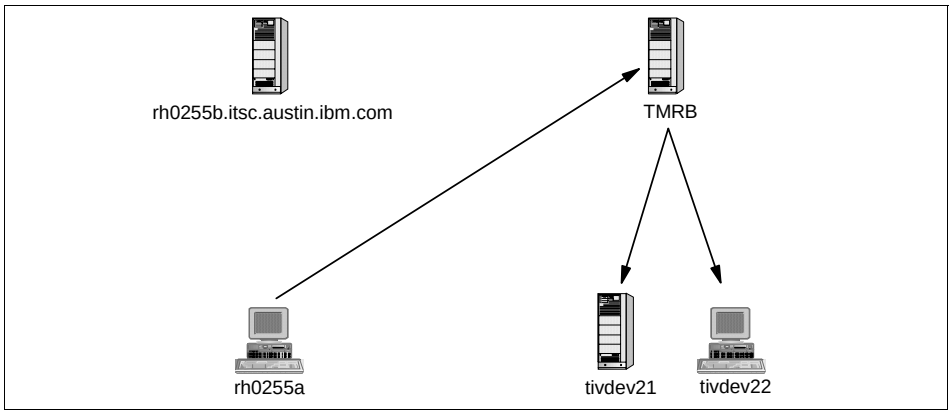


Figure 118. Non-Repeater to Targets in Another TMR

In summary, a non-repeater can only send data to one other node at a time. For distributions to multiple nodes, Mdist will find a repeater to perform the distribution. If a repeater has to send data to a single node normally served by a different repeater, then the second repeater will not be used as it just adds an extra step. Note that this means a repeater could be distributing data to a machine you might not expect it to.

7.6.3 Bulk Data Transfer and Inter-Object Messaging

Bulk Data Transfer (BDT) is a technique used when large data transfers (typically greater than 16 KB) are required between objects on separate systems. Rather than have the data go from object on system A through oserv on system A to oserv on system B to object on system B, BDT utilizes a technique know as Inter-Object Messaging (IOM). IOM 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. The terms BDT and IOM are often used interchangeably to mean the same thing. Strictly speaking, BDT is what we need to do, and IOM is how we do it.

BDT/IOM:

- Is used anytime the data is being transferred is larger than *16 KB* in size.
- Does not use the port 94 for transferring the data.
- Is not just used during multiplexed distributions.

For transferring small bits of data (less than 16 K) back and forth, the server and managed node use the already open ports (managed nodes) or re-open

the known ports (6543 for PC managed nodes and usually 9494 for TMAs). Anytime there is a chance for large amounts of data to be sent, the TMR Server and managed node will attempt to open an Inter Object Messaging (IOM) channel. This is the process:

1. TMR Server opens a new port above 1023 (subject to `set_port_range` restrictions for managed node).
2. TMR Server discovers its own IP-address:

   It checks for a /etc/wlocalhost file (or for NTs a registry entry found with the `wlocalhost` command) and if it exists, uses this text name to do a `gethostbyname` to return the IP-address.

   If this doesn't exist, it does a `gethostname` function call to find out what its own host name is, then does a `gethostbyname` on that name to find out its own IP-address.
   This is in case there are multiple adapters in the server.
3. The TMR Server then makes up a `dkey` consisting of IP-address: `port#:secretkey` and sends this to the open port of the endpoint.

> **Note**
>
> From 3.1.3, 3.2 with SuperPatch and 3.6 and above, the `dkey` now also includes the string host name from step 2.

4. The endpoint will then open some port above 1023. For managed nodes, the port number used is subject to the `set_port_range` restrictions. For TMAs and PC managed nodes, the port opened is assigned by the operating system. Then the endpoint attempts to connect to the IP-address and port# sent in the `dkey`.

> **Note**
>
> When the string host name is sent, it is used if name resolution is successful using the host name provided. This allows the endpoint to connect back using the address provided through local name resolution instead of needing to have a route back to a specific IP address.

Firewall Considerations

Every possible effort should be made in order to avoid IOM transfers to PC managed nodes and TMAs crossing Firewalls. Since the port number opened to receive data on the endpoints cannot be controlled, the Firewall should open all ports for the endpoints.

7.7 UserLink and Dynamic Host Configuration Protocol (DHCP)

This section explains how the UserLink/DHCP component works and illustrates the install process. This will help you configure and troubleshoot it.

7.7.1 Dynamic IP Addressing and Tivoli

Dynamic Host Configuration Protocol (DHCP) allows an organization to dynamically assign IP addresses to PCs in the network. When a PC connects to the network, the DHCP server assigns the PC an available address from a defined range of addresses. Each PC leases the address it receives. When the lease time expires, the IP address is disassociated with the PC and returned to the list of available IP addresses (lease periods were specified when your organization configured DHCP.)

Tivoli relies heavily on using the correct IP address for managed nodes and endpoints. In order to function well in a DHCP environment, Tivoli provides the UserLink component in the Tivoli Framework. This support is provided for the following managed resources:

- Windows NT managed nodes.
- Windows 3.x, Windows 95, or Windows NT PCs running an IP agent.
- All endpoint clients.

Note

Install the UserLink on the Tivoli server and on any managed node that will service requests from PCs. This daemon accepts requests from PCs to update their IP address or distribute file packages to them.

The UserLink/DHCP service provides IP address synchronization between the PC agent and its associated PC managed node. PC agents communicate directly with the UserLink/DHCP service passing to the service the PC's current IP address and the name of its associated PC managed node. The UserLink/DHCP service then updates the IP address maintained by the PC

managed node. An end user can also manually update the PCs IP address using the UserLink browser.

The UserLink/DHCP service can be installed on any managed node in the TMR. You can install it on the TMR server, but this is not a requirement. When you install a TCP/IP agent, you are prompted for the location of the service. Once installed, the PC agent contacts the service on the server you specified. A single UserLink/DHCP service provides DHCP support for all Windows PC agents in the TMR.

If you have Windows NT managed nodes that require DHCP support, you must install an NT TMR server in your environment. Ensure that the NT TMR server is running WINNS for name resolution; DNS cannot resolve addresses allocated for DHCP address leases (you can use DNS for non-DHCP client name resolution).

Note

The TMR server oserv needs to be configured to allow dynamic IP address support using the following command: `odadmin allow_dynamic_ipaddr TRUE`

You must first configure DHCP in your environment. The Framework does not provide DHCP; it simply enables you to use DHCP and Tivoli in your computing environment.

When a Tivoli PC managed node connects with the TMR server, the PC managed node passes its current IP address to the server. If the IP address is different from that previously known for the PC managed node, the server updates its address mappings. The TMR server can resolve IP address changes caused by DHCP as well as those caused by moving a machine to a new subnet. You can configure a PC agent to update its IP address at start up only or at regular intervals while the agent is running.

7.7.2 The UserLink/DHCP Service

The UserLink/DHCP service, also known as the usrlnkd daemon, has two purposes:

- Provides IP address synchronization for PC managed nodes and NT managed nodes using DHCP.
- Enables an end user on a Windows, Windows 95, or Windows NT PC to retrieve Tivoli Software Distribution profiles, which include file packages and AutoPacks, using the UserLink browser.

For DHCP support, the communications between the PC and the UserLink/DHCP service are straightforward:

1. When the PC is booted, or if the agent is configured to update the IP address at intervals, the PC agent contacts the UserLink/DHCP service sending the PCs IP address.
2. The UserLink/DHCP service runs the set_ip.pl script and creates the pc_name.ip file. This file is created in the $BINDIR/TAS/USERLINK directory or in the directory specified by the USRLNKD_STAGING_PATH environment.
3. The service updates the IP address in the PC managed node object.

If an end user manually updates the IP address using the UserLink browser, the same set of actions occurs.

Note

The UserLink/DHCP service is provided for PCs running the PC agent; Tivoli endpoints need not rely on this service.

7.7.3 DHCP Support for Windows NT Managed Nodes

If you have Windows NT managed nodes that require DHCP support, you must install an NT TMR server in your environment. Ensure that the NT TMR server is running WINNS for name resolution; DNS cannot resolve addresses allocated for DHCP address leases (you can use DNS for non-DHCP client name resolution).

When a request comes in from an NT managed node running DHCP, and the NT TMR server has dynamic IP address support enabled, it checks and confirms its IP address in the odlist. If the IP address is not there, the server checks the host name in the odlist and then uses WINNS to resolve the IP address. The server then updates the IP address in the odlist to the managed node's request.

7.7.4 DHCP Support for PC Managed Nodes

If you have PC managed nodes that require DHCP support, you must complete the followings steps. Prior to performing these steps, you must configure DHCP in your environment:

1. Install the UserLink/DHCP service on a TMR server or managed node in your TMR. You must install the service on at least one managed node in a TMR, though this managed node need not be the TMR server. Tivoli

recommends that you install one UserLink/DHCP service for each 200 DHCP clients. The service's performance is also dependent on the managed node's operating system, specifically, the number of available file descriptors and process IDs.

Note

Do not install the UserLink/DHCP service on a Windows NT PC managed node. The UserLink/DHCP service and the PC agent both use port 6543 to communicate with the TMR server, forcing you to shut down the PC agent to run the UserLink/DHCP service.

2. Create a PC managed node for each PC that will use the UserLink/DHCP service.
3. Install and configure the PC agent on each of the PCs for which you created a PC managed node.

If you want to enable your PC users to retrieve file packages and AutoPacks, complete the steps described above, and install the UserLink browser on each PC. The UserLink browser also enables the end user to manually update the PCs IP address.

7.7.5 Installing the UserLink/DHCP Service

The UserLink/DHCP service can be installed on any server or managed node in the TMR. A single UserLink/DHCP service can provide DHCP support for all PC agents in the TMR. Tivoli recommends that you do not install the service on the TMR server to avoid undue load on the TMR server.

The authorization roles required to install the UserLink/DHCP service are product_install or super for all the TMR. You can install the UserLink/DHCP service from the desktop or the command line.

The steps of how to install the DHCP using the Tivoli Desktop and the command line are explained in the Chapter 12 of the *Tivoli Framework Planning and Installation Guide*, Version 3.2. This chapter also explains how to change the default directory in which the UserLink/DHCP service temporary files are created.

7.7.5.1 Creating a PC Managed Node

You must create a PC managed node for each PC in your TMR. For PCs using DHCP, you should create the PC managed node before installing the PC agent on the PC. When the PC agent is started, and if it is configured to do so, it passes its IP address to the DHCP service. The service then

forwards the address to the PC managed node. Thus, the PC managed node should exist first to receive this IP address.

The steps to create PC managed nodes are explained in the *Tivoli Framework Planning and Installation Guide*, Version 3.2.

You should select both **No** radio buttons in the lower-half of the dialog. By doing so, you indicate that the PC managed node will represent a PC that is running DHCP, and that the PC is currently off-line, respectively. Figure 119 on page 277 shows the Add Hosts dialog.

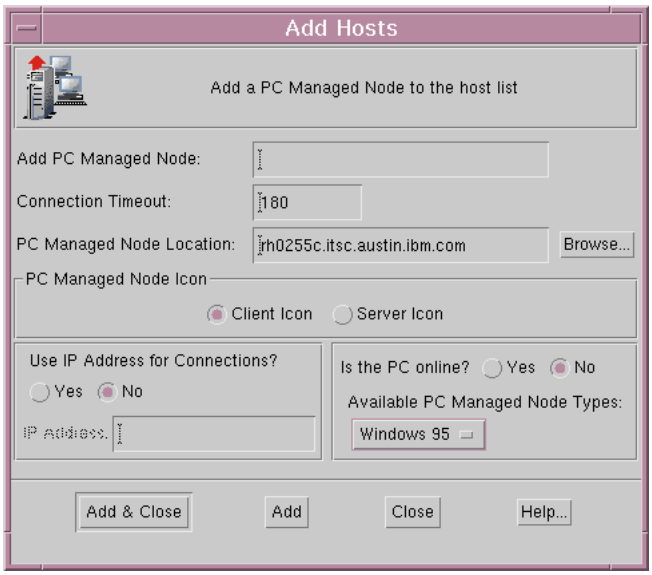


Figure 119. Creating a PC Managed Node for a DHCP PC

7.7.5.2 Installing and Configuring the PC Agent

You must install and configure the PC agent to enable communication with the PC. The steps in the following procedures describe the part of the PC agent installation that is specific to the UserLink/DHCP service. Note that to customize the agent, you must edit the TMEAGENT.CFG file using a text editor:

1. Begin the PC agent installation as described in the *Tivoli Framework Planning and Installation Guide*, in the chapter “Installing a PC Agent.”
2. Enter the name of the machine on which you are installing the agent when the following dialog is displayed:

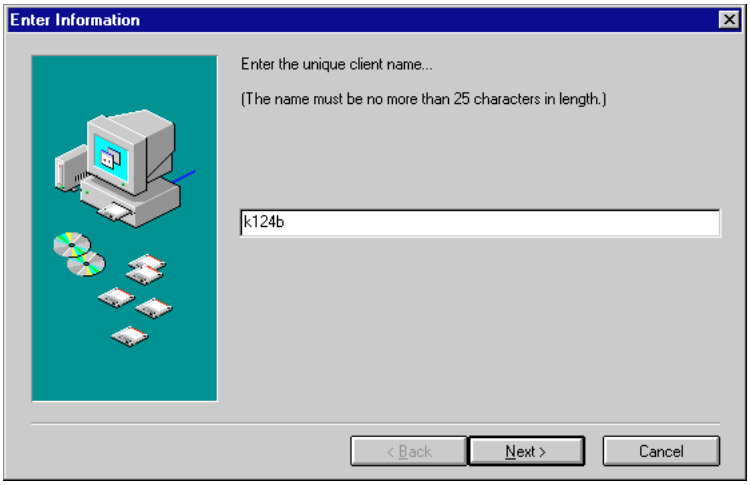


Figure 120. Unique Client Name

The name you enter in this dialog must be the same that you used when creating the PC managed node for the PC. Ensure that you type it the same as before. This updates the DefaultServer entry in the TMEAGENT.CFG file.

3. The setup program displays a series of dialogs to lead you through configuring the agent to either run under a Tivoli NetWare repeater or to use the UserLink/DHCP service for IP address synchronization.

   Select the **Yes** button when the dialog ask you to use a default server for DHCP IP Synchronization.

> **Note**
>
> If a TCP/IP agent is running under a Tivoli NetWare repeater, DHCP support is provided by the NetWare repeater.

4. Enter the server or managed node on which you installed the UserLink/DHCP service as shown in Figure 121 on page 279. You can specify either the name or the IP address.

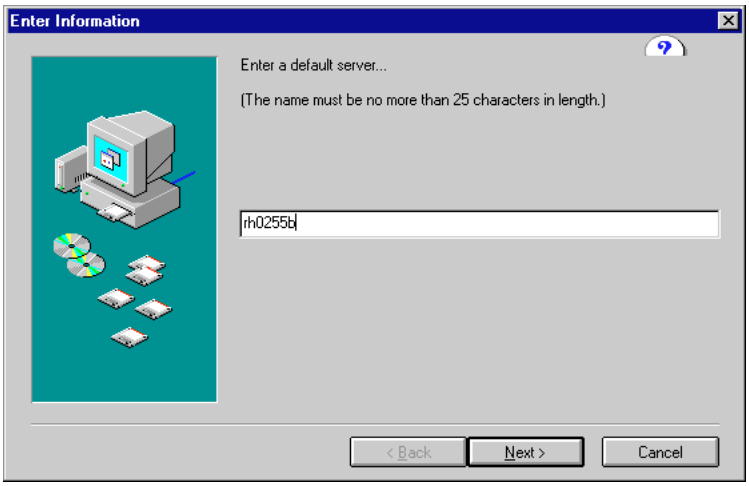


Figure 121. Default Server

5. Finish the PC agent installation.
6. Include the following entries, in the TMEAGENT.CFG file, which is installed in the C:\ETC directory on all operating systems except NetWare. For NetWare, the TMEAGENT.CFG file is installed in the ETC directory of the SYS volume.

   DefaultServer=hostname

   Identifies the name of the server or managed node on which the UserLink/DHCP service is installed. You can also enter this name during the PC agent installation.

   UpdateIPAtBootup=YES

   Updates the IP address each time the PC agent is started. By default, this entry is set to NO.

   AutoUpdateIP=YES

   Specifies that the agent should automatically update the IP address based on a time interval. The interval is set by the UpdateIPInterval entry.

   AlwaysUpdate=YES

   Specifies that updating should continue even after a successful update. If this entry is set to NO, AutoUpdateIP is disabled once the LinkStatus entry is set to GOOD.

   UpdateIPTries=number

Indicates how many times the agent should attempt to update the IP address after an initial failure.

UpdateIPInterval=`minutes`

Specifies the number of minutes between AutoUpdateIP attempts. You can enter between 5 and 144000 minutes. The default is `1440`, which is equivalent to one day.

Note 1

If you put `@h` or `@H` in the CLIENTNAME entry in the TMEAGENT.CFG file, the agent uses the host name as the client name when it runs. This enables the PC agent to automatically update its client name if the host name has changed through DHCP. You can see the change when opening the properties window of the PC managed node. This is only supported for PC agent 4.009 or a later version. This is not supported on the LAN Workplace stack.

Note 2

If you change the name of the PC managed node by selecting the **open editable properties of the managed node**, WIN95ClientObjectID is changed to ClientObjectID.

By modifying these entries, you can alter the DHCP behavior to better suit your environment. Tivoli recommends that you configure the agent to update its IP address when the agent is started and at specified intervals after start up. The interval at which the agent updates its IP address should correspond to the length of the PCs lease of the IP address.

Other entries that enable DHCP support but that *should not be edited* include:

UpdateIPDate=`date`

Indicates the last date and time that the IP address was successfully updated. This information is used to determine when the next update should occur.

UpdateIPStamp=`stamp`

Used internally by the PC agent.

LinkStatus=`status`

Indicates the status of the connection between the PC agent and the UserLink/DHCP service. This entry is set to `GOOD` when the agent has successfully passed the current IP address to the UserLink/DHCP service.

See Appendix C of the *Tivoli Framework Planning and Installation Guide*, Version 3.2 for a detailed description of the TMEAGENT.CFG file.

7. Restart the PC agent.

7.7.6 UserLink Daemon

The UserLink daemon (usrlnkd) is designed to listen for requests from PCs to change their IP address or to receive file packages. When UserLink is installed on managed nodes, the usrlnkd will run on that machine. When installing the PCs, there will be options to designate the contact host. This allows the load to be spread out within the TMR.

Note

An NT managed node cannot run both a UserLink daemon and the PC agent. Both daemons listen on the same port (6543).

The steps of the PC agent communication and the UserLink daemon are as follows:

1. The PC agent running `set_ip.pl`:
   - Sends the `pc_name` to usrlnkd.
   - The `pc_name.ip` file in `$BINDIR/TAS/USERLINK` is created.
   - The IP address in the PC managed node object is updated.
2. Queries running `gt_version.pl`:
   - Lookup all file packages, PC manager, and subscribers.
   - Send list of file packages.
   - Create two files in `$BINDIR/TAS/USERLINK` `pc_name.err` and `pc_name.ver`.

Note

The UserLink agent (used to get the software distribution packages) has a configuration file called `usrlnk16.ini` (Windows 3.x) or `usrlnk32.ini` (Windows 95 or NT) used to configure the timing for IP address updates.

3. Communication and distribution occur over port 6543:
   - Runs `pulldist.pl`.
   - Sends the name of the PC managed node and file package.

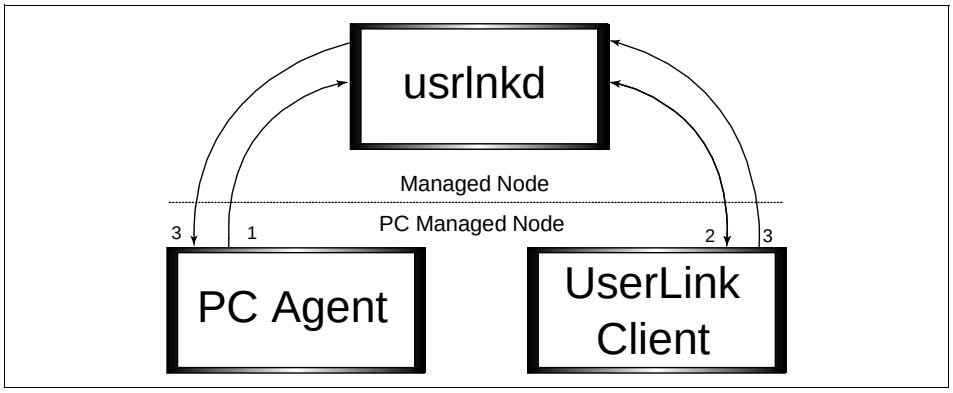


Figure 122. Communication Between PC Agent and usrlnkd

7.7.7 Retrieving Software Packages

Another purpose of the UserLink/DHCP service is to enable an end user on Windows, Windows 95, or Windows NT to retrieve Tivoli Software Distribution profiles, which include file packages and AutoPacks, using the UserLink Browser.

The UserLink browser, which you must install in addition to the service, is illustrated in Figure 123 on page 283.

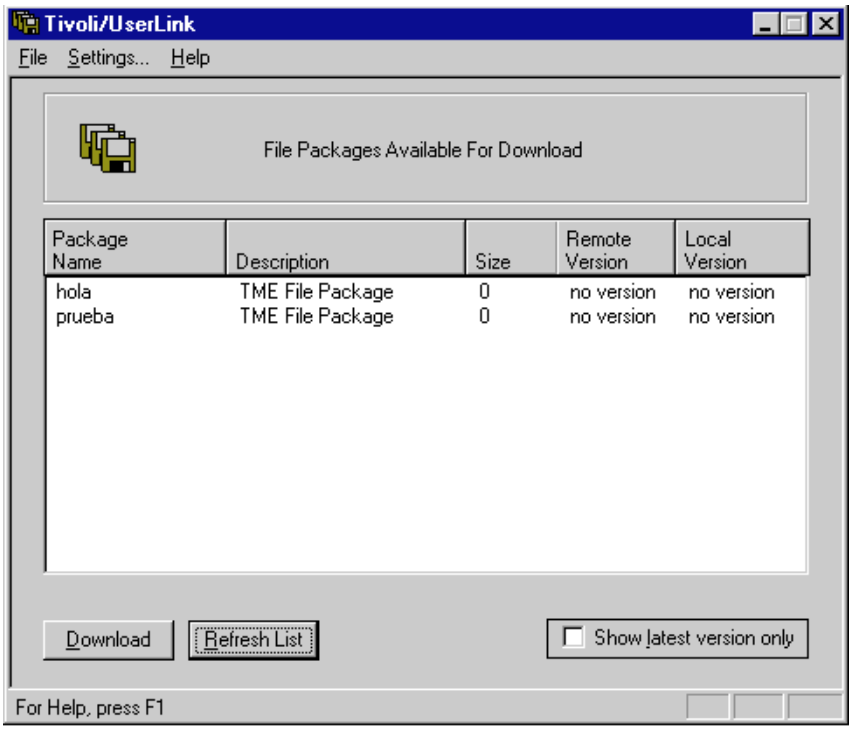


Figure 123. UserLink/DHCP Browser

The UserLink browser communicates directly with the UserLink/DHCP service to determine the available file packages and AutoPacks and to retrieve the selected ones.

When the PC user presses the **Refresh List** button on the UserLink browser, the browser queries the UserLink/DHCP service for a list of file packages and AutoPacks to which the PC managed node is subscribed. The UserLink/DHCP service returns the list of available file packages and AutoPacks to the browser. When the user selects the profile to retrieve and presses the **Download** button, the selection is passed through the UserLink/DHCP service. The file package, or AutoPack, is then distributed to the PC managed node.

The following steps describes the actions that are taken for a software retrieval:

1. The UserLink browser sends the PC managed node's name and request to the UserLink/DHCP service.

2. The service runs the `get_versions.pl` script and performs the following actions:
   - The script searches the Tivoli database for all file packages and AutoPacks and their corresponding profile managers.
   - The `get_versions.pl` script then checks the subscription list of each profile manager for the PC managed node and generates a list of available profiles for the PC managed node.
   - The script creates the `pc_name.ver` and `pc_name.err` files in the $BINDIR/TAS/USERLINK directory or in the directory specified by the USRLNKD_STAGING_PATH environment variable.

   If the UserLink/DHCP service is not installed on the TMR server, the `get_versions.pl` script runs on the managed node where the service is installed (the UserLink server) but initiates database lookups on the TMR server. The TMR server generates and sends the list of available profiles to the UserLink server. The service then creates the `.ver` and `.err` files locally.
3. The UserLink/DHCP service sends the list of profiles to the UserLink browser on the PC managed node.

   If an end user retrieves, or pulls, one of the available file packages or AutoPacks, the following occurs:

   1. The UserLink browser sends the request to the UserLink/DHCP service. This request includes the PC managed node's name and the name of the requested profile.
   2. The UserLink/DHCP service runs the `pulldist.pl` script, which runs the `wdistfp` command. You can modify this script and add the `-u` argument to the `wdistfp` command. The `-u` argument displays a dialog that illustrates the progress of the distribution based on the file package size.

      Note

      The `-u` option for the `wdistfp` command is a new feature of Tivoli Software Distribution. It allows you to create a file package, calculate its size, and store it. When the distribution to a PC occurs, it uses the size to calculate the status of how much data has been sent to the PC. To use this feature, the `-u` option must be added to the `wdistfp` command in `pulldist.pl`.

   3. The requested file package or AutoPack is distributed to the PC managed node. The PC agent receives the profile over the port 6543 and installs the files.

7.7.8 Installing the UserLink Browser

For an end user to retrieve a Tivoli Software Distribution profile or to update the PCs IP address manually, you must first install the UserLink browser. Recall that the UserLink browser is supported on only Windows, Windows 95 and Windows NT machines. For instructions on using the UserLink browser, see the on-line help provided with the browser (in the same directory in which the UserLink browser is installed).

For instructions on how to install the browser see the *Tivoli Framework Planning and Installation Guide*. This chapter explains how to install the browser either using the `setup.exe` command that is included in the CD or using a Tivoli Software Distribution Package.

7.7.8.1 UserLink Browser Version System

Creating file packages with `^#` at the end denotes file package versions.

Pushes to the agent do not update the UserLink .cfg file. Write a configuration program to update these files so that the UserLink client browser knows which file package is the latest.

The UserLink daemon keeps the version information in .cfg files on the PC.

7.7.9 Troubleshooting UserLink

Whether the end user is updating the IP address, refreshing the profile list, or pulling a software package, the UserLink/DHCP daemon will generate a log file. This file is created in the directory where the usrlnkd service resides. If the UserLink/DHCP service is not running, use the `wstartul` command on the machine where the service is installed.

There are several common problems that can occur when trying to perform UserLink queries:

- Cannot find the UserLink Server. Verify that all the installation process (especially the PC agent ones) are OK.
- UserLink daemon is not running (usrlnkd). This daemon is started automatically by the oserv at oserv boot time. It can also be started with the `wstartul` command.
- Cannot find the `get_versions.pl`. This script is found in the user link directory and is searched for relative to the usrlnkd directory path.

You can see if there is an error between the PC agent and the usrlnkd by looking at the pc_name.err or pc_name.ip files. Starting the UserLink daemon

by hand displays all connections and communications it received on standard out. Use the following to start the usrlnkd daemon by hand:

```
cd $BINDIR/TAS/USERLINK

./usrlnkd
```

This can be a useful way to find errors between the PC agent and usrlnkd daemon conversation.

Chapter 8. Tivoli Enterprise and Firewalls

This chapter is based on a White Paper written within Tivoli product support. Updates to this paper can be obtained by registered customers from the Tivoli Support Web site at:

`http://www.support.tivoli.com`

The general purpose of this chapter is to discuss the ramifications and issues associated with installing the Tivoli Framework in a firewall environment. We discuss the general areas where Tivoli 3.6 deployment is influenced by firewalls and give guidelines for those implementing Tivoli in a security conscious environment.

8.1 Background

With the Internet creating a virtual extension of your enterprise cyber-space, Network users often transcend the domains of their private Network across public Netspace. Network security is, therefore, an essential facet of every installation. Firewalls are Network access control devices that are used in securing the Network perimeter of an enterprise installation. The use of firewalls in a Tivoli Framework installation creates interesting issues during deployment.

Firewall placement influences the workings of the Tivoli installation. This paper addresses these issues in an effort to clarify the questions and myths surrounding this topic. Note that Tivoli also has a suite of products dedicated to secure systems management in an Internet environment known as Tivoli CrossSite.

8.2 Tivoli Communications

This section summarizes the different kinds of networking connections established between Tivoli components in an installation. Understanding the Network connectivity paths is key in understanding Tivoli operation in firewall environments.

All TMR Servers, managed nodes/gateways, endpoints, PC managed nodes, and NetWare Servers must have the TCP/IP protocol installed, configured, and operational on their systems. This is a fixed requirement for the Tivoli Framework to function. The only exception is IPX/SPX for Netware Managed Nodes, but Tivoli does not support operations between a TMR server and IPX/SPX Managed Nodes through firewalls.

The various types of communications can be summarized as follows:

- Inter-ORB communications
- Inter-TMR communications
- Inter-Object Messaging (IOM)
- Endpoint and gateway communications
- Applications not using Framework Services

8.2.1 Inter-ORB Communications

Inter-ORB communication is the basic interaction mechanism between the different Object Request Brokers (ORBs, also known as oservs) in the installation. Another name for this is Inter-dispatcher communication. It uses TCP connections. The communication takes place whenever requests from one managed node invoke a remote method on another managed node. The Inter-ORB communication also manages communications between a Managed Node and a TMR Server when a method request needs to be authorized and its implementation details resolved. This connectivity is also termed as the Object-Call or objcall channel.

The ORB that initiates the connection acts as a TCP client, while the ORB that accepts the request is the TCP server. The roles of TCP client and TCP server are in the context of the request and do not necessarily have any relationship to Tivoli roles of Client and Server. In other words, an ORB that is a server to one remote ORB can also invoke TCP client requests on other remote ORBs.

The ORB dispatchers that communicate over this connection have a sustained TCP connection over TCP port 94. These connections can only be disrupted due to Network faults or if a dispatcher is restarted.

8.2.2 Inter-TMR Communications

A TMR is defined as a set of managed nodes controlled by one TMR Server. In addition, multiple TMR servers can be linked together to assist in making the managed enterprise scalable. Once they are linked, managed resources can be exchanged, and the two TMRs can more or less function like one extended TMR. See 2.3.5.4, “Interconnected TMR Resource Exchange” on page 27 for more information on exchanging resources.

The communications between two ORBs of two TMR servers is known as Inter-TMR communication. From a Network communication perspective, the

Inter-TMR communication follows the exact same pattern as Inter-ORB communications.

8.2.3 Inter-Object Messaging (IOM)

This is a communication channel that allows exchange of bulk data between two object implementations. This exchange takes place independent (or out-of-band or OOB) of the objcall channel. The creation of IOM channels is during method execution on an on-demand basis. The IOM connection channels also connect using TCP. Examples of OOB communications typically run over ephemeral TCP ports above 1023. These OOB connections can be tuned using commands to the Tivoli oserv through mechanisms described later in this chapter.

IOM channels are created by a method that wants to initiate bulk data transfers greater than 16 K Bytes in size with a remote method of an object. This connection will require use of OOB communications ports. Communications less than 16 K bytes in size run over the existing TCP port 94 communications mechanism.

The local method acts as an IOM-channel TCP server. It selects a listening port and starts up a Network server and passes the connection information to the remote method. This information, known as the DKEY, gets transferred securely through the objcall channel as an argument to this remote method. The remote method at the other end, when invoked, uses the DKEY to establish the client end of the IOM TCP connection. Once the bulk data is exchanged, the IOM connection gets torn down at both ends.

Examples of users of IOM calls are Tivoli User Administration, Tivoli Software Distribution, and the Tivoli InSecure Logfile Adapter (provided with the Tivoli Enterprise Console).

See 2.3.3.1, “Inter-Object Messages” on page 19 for more information.

8.2.4 Endpoint and Gateway Communications

An endpoint running the TMA package first uses UDP to advertise its presence in the enterprise. Gateways read UDP messages destined for the gateway port. Once an endpoint is recognized as valid, it is assigned a permanent gateway. The endpoint then connects as a TCP client to the TCP server on the gateway daemon. Note that this connection is not constant; it is broken down between each login activity. The endpoint TMA server listens, typically, on TCP port 9494 and also accepts TCP client connections coming down from the gateway.

Unlike managed nodes, TMA endpoints only communicate with gateways. From a Tivoli perspective, they do not need to communicate between themselves.

8.2.5 Applications Not Using Framework Services

Some of the Tivoli applications do not always use the Framework services for Network communication and so will not honor port ranges you set. These application actions include:

- UserLink client to UserLink server
- NetWare Managed Site to NetWare Repeater
- PC Managed Node to PC Agent
- TEC non-secure event logging (posting)

8.3 Ports and Port Ranges

As previously discussed, the Object Dispatcher service listens on TCP port 94. This is a registered port, assigned to Tivoli by the Internet Assigned Numbers Authority (IANA). All Inter-ORB and Inter-TMR communications with the Object Dispatcher use this port as the destination. Object call requests cause the oserv to create TCP client connections (to the remote object dispatcher service) using ephemeral ports (TCP ports above 1023).

The object-call induced client ephemeral connections, the IOM channel server, and IOM ephemeral client connections can (optionally) locally bind to a port from a pre-selected port range. This port range selection in Release 3.6 is on a per-TMR basis. In Release 3.6.1, this port range selection is further enhanced to allow selection on a per-managed node basis. The selection is done using the `odadmin set_port_range` command. This feature of selecting port ranges plays a major role in firewall environments. The port range can be reset to no restriction using `odadmin set_port_range ""`.

It is important to understand that if a port range is set through the `odadmin set_port_range` command, that ALL TMRs must be able to communicate on either a subset of the ports defined or all TCP ports for inter-TMR communications to work properly.

When one has set the port range for the communications, it is important for the implementor to understand that this range is changeable in the future.

The MDist service used for fanning out data uses IOM channels to multiplex data transfer to a large number of destinations. Once a port range is selected,

all subsequent IOM traffic for MDIST fan-outs will use ephemeral ports from this range. MDist repeaters (including the TMR server) must have a large number of ports available to choose from. A good rule of thumb is to provide a range that is three times the number of managed nodes in the system.

This suggestion comes from:

- One port for every managed node that a node needs to interact with for inter-ORB connections.
- One port for each IOM connection that is opened at the repeater node.
- One port for padding for more simultaneous IOM connections.

There are some differences in the port selection algorithm between NT and Unix. Although implementation related, the rest of this section addresses these differences to help you to understand Tivoli behavior at deployment.

The algorithm differences stem from having to get around some of the non-standard (X/OPEN non-compliance) features of the Winsock API. On Unix, a port is considered *IN USE* if it is held in the TCP *TIME-WAIT* state and binds to these ports fail appropriately. On NT, subsequent binds to ports already in *TIME-WAIT* bind successfully but fail during the connect () call.

On Unix, when a managed node/gateway binds to a port from the range, Tivoli starts at the lowest number in the range and hunts for a port that will bind successfully. If bind () fails, the Tivoli communication moves to the next port in the port-range and retries the bind. This bind attempt continues, until the bind () finds an available port. It finally gives up the search when the upper bound in the range is reached.

On Windows NT, Tivoli always remembers the last port bound and connected successfully from this range. Subsequent bind attempts start from the next port. If the bind fails, the algorithm proceeds just like in the UNIX case, except that when the upper bound is reached, it wraps around until the previous successful port range is reached. This wrap-around is needed because we may start at a port range that is already the upper boundary. This practice has been found to be more efficient than starting from the lower boundary and moving up.

However, if the NT bind succeeds, but the connect () fails with EADDRINUSE, the algorithm makes five more attempts to bind and reconnect, each time bumping the port by one. When all five attempts fail to connect, it gives up.

The gateway daemon/service is free to listen on any port chosen during the gateway creation. This is not registered with the IANA. The gateway port selection is totally unrestricted. The default port for the gateway daemon is 9494.

The endpoint TMA also listens on a well-known port selected during installation. Like the gateway port, its selection is unrestricted and defaults to 9494 (in Release 3.6.1 this default will be changed to 9495). If the selected port is unavailable (this can happen if the daemon is terminated and restarted within the 2-segment TCP lifetime), the TMA can be configured to either fail immediately or pick an arbitrary port that is available.

After an endpoint upgrade, the TMA is restarted and bound to the same preferred port chosen during selection. This, again, plays an important role for framework installation with firewalls.

It should also be pointed out that during installation time Tivoli may require the use of TCP port 513 (rexec) to install the product properly. Additional discussions of this should be held with your Network administrator/firewall administrator to determine if this will be a problem in your specific environment. If it is, have the port opened ONLY for the period that one must install a node/client, and then the port can be closed.

Gateway TCP client connections to endpoints use ephemeral ports. In Releases 3.6 and 3.6.1, gateways reside on managed nodes. If port selection is chosen on the managed node, the gateway daemon initiated client connections will honor this selection and find an ephemeral port from this range.

Endpoint TCP client connections also use ephemeral ports. Endpoint ephemeral ports cannot be selected. There would be too much overhead managing port information for tens of thousands of endpoints in addition to the migration and login strategies currently allowed for endpoints.

Table 10 summarizes the port specific information:

Table 10. Tivoli Port Usage Summary

Type of Connection	TCP Server Listening Port	TCP Client Ephemeral Port	Protocol /Duration
Inter-ORB	94	Either chosen from selected range or assigned by OS	TCP / Sustained

Type of Connection	TCP Server Listening Port	TCP Client Ephemeral Port	Protocol /Duration
Inter-TMR	94	Either chosen from selected range or assigned by OS	TCP / Sustained
Inter-Object Messaging	Chosen from selected range or assigned by OS	Chosen from selected range or assigned by the OS	TCP / On-Demand creation for duration of bulk-data exchange
Endpoint Initial Login	Gateway server port chosen at installation (default 9494)	Ephemeral port provided by the OS at the endpoint	UDP / Initial login to determine assigned gateway
Endpoint Normal Login to gateway or upcalls	Gateway server port chosen at installation (default 9494)	Ephemeral port chosen by OS at the endpoint.	TCP Connection is not kept after login
Secure LogFile Adapter	94	None	TCP / Sustained during periods of activity
Insecure Logfile Adapter	Chosen from a selected range or assigned by OS	Ephemeral port chosen by OS at the endpoint	TCP / Sustained during periods of activity
Remote Control	Chosen from a selected range or assigned by OS or port 2501	None	TCP / Sustained during the period of a session
Gateway's client connection to endpoint (downcalls)	Endpoint server port chosen at installation (default 9494/5)	Ephemeral port chosen either from the selected port range or assigned by OS at the gateway	TCP / Sustained for the duration of a downcall or control packets

8.4 Firewall Considerations

Firewalls come in several flavors. Firewalls, typically, either consist of network layer filters to protect access to networks and hosts or as application proxies that monitor the services that are allowed to operate across the boundaries of the corporation.

One of the most important issues of using firewalls is their level of transparency to user applications while maintaining the sanctity of the Netspace. Since the Tivoli framework is a system management infrastructure, this level of transparency is influenced by the firewall placement relative to the Tivoli components.

The Tivoli Framework currently operates only with Network layer (ISO layer 3 and above) firewalls. Application proxies need to be provided by firewall vendors, such as Checkpoint FW-1, IBM SNG, Eagle Raptor, or the like.

Virtual Private Networks (VPN) is another firewall strategy that is growing in acceptance. Typically, VPNs should be transparent to Tivoli; however, no official testing has been done to validate Tivoli operations under a VPN infrastructure.

TCP/IP tunnels through firewalls using technologies, such as SSH or GRE tunnels, work with Tivoli provided that end-to-end IP and above connectivity is reliable. SSH or GRE tunnels, in many instances, provide greater security than a traditional firewall but are outside the scope of this chapter.

8.4.1 Packet Filtering

Routers often embed firewall functionality inside them. These usually do packet filtering by parsing out Network layer information from packet headers as packets flow through the router. The router is configured to monitor header fields, such as source and destination address, source and destination ports, protocol used, and so forth. Firewalls permit or deny access based on filters set for these fields.

In a three tier TME3.6 installation, firewalls may be needed between server and managed nodes that spread across disparate Networks. Placing firewalls between a TMR server and a gateway is quite common since gateways are repeaters. For readability sake, this chapter labels such firewalls as *upstream firewalls*.

The considerations for upstream firewalls are as follows:

- Port range selection on managed nodes is turned on. Obviously, for someone deploying firewalls, this is a basic requirement.
- TCP protocol must be allowed to flow through.
- Filtering by source and destination address of the managed nodes can restrict traffic to the set of managed nodes that span across the firewall. Note that not all managed nodes may need to talk to each other.

- Inter-ORB connectivity requires TCP port 94 to be permitted as a destination port. The source port must be one from the selected Tivoli port range for the TCP client (request-originating) ORBs.
- Inter-TMR connectivity, again, requires that port 94 be permitted if there are two TMR servers being connected across a firewall.
- Ports selected for IOM client connections need to be let through.
- In Release 3.6.1, port selection is available on each managed node. Consider two managed nodes that communicate with each other (for example, when a repeater on one managed node distributes data to the other managed node). The firewall port selection between them may have to be tailored to suit the port selection chosen specifically for these two managed nodes. This functionality also increases administrative overhead on a Managed Node from an Operating System maintenance perspective.
- TME installation uses network services, such as rexec and TRIP (Tivoli's version of `rexec`, for NT, see Appendix A.5.1, "Installation of the Tivoli Remote Installation Package" on page 616), which do not allow port selection for connectivity. Thus, the firewall needs to let through `rexec` type connections during the deployment. Once deployed, the firewall port ranges can be tightened.

Firewalls between gateways and TMA endpoints pose some challenges. These are labeled here (again, only from a readability standpoint) as downstream firewalls. Tivoli feels that putting firewalls between endpoints and gateways is not a best practice for a security conscious environment and should require careful discussions and investigations before implementing such an approach.

Downstream firewalls, if deployed, must accommodate all of the following factors for gateway to endpoint communications to work:

- TMA endpoints initiate login packets using UDP client connections to the gateway UDP server. UDP is used during gateway fail-over/recovery modes too. Allow UDP packets (destined for the gateway listening port) to flow through upstream from the endpoints towards the gateways.
- As mentioned before, endpoint ephemeral ports are not controlled. The only deterministic port (that can be selected) on the endpoint is the TMA daemon (service) listening port. Firewalls must not be configured to restrict the endpoint source port range.
- Gateway downstream client connections choose ephemeral ports from the port range selected for the managed node. This range ought to be large enough to allow the gateway to service all its endpoints during distributions. Remember, gateways act as repeaters for their endpoints.

- The gateway repeater should be tuned (max_conn) to ensure that the distributions do not time-out waiting for available ports.
- TME supports endpoints configured with DHCP. If the endpoint that has already registered loses its DHCP lease and either reconnects by restarting the TMA daemon or does an up-call with the new IP address, the new IP address is dynamically updated in the endpoint manager. However, firewalls between must be configured to be insensitive to this IP address change.

8.4.2 Machine Considerations - Upstream

Machines located outside (upstream) of a firewall should have specific TCP/IP routing policies set up on them to ensure additional conflicts and compromises are not inflicted on an internal (downstream) Network. These include making the external machines only have a static TCP/IP route to the internal (downstream) TMR server. The router between the Internet and the external Network should have explicit TCP deny filters for access from the Internet for the ports opened to an external Tivoli client. This router should only have explicit TCP permit filters for the machine to talk the protocol(s), permitted doing its normal work and the ports that Tivoli requires to talk to the internal TMR server.

Next, these nodes should have their default TCP/IP route set to the external/internal firewall interface. This coupled with the router access-list inclusions, and the explicit permit filters for said node provide additional and logical security for the machines to limit their port exposure from the Internet and keep these machines from being launching points to the internal Networks. This should be done to each machine external (upstream) of a firewall that runs the Tivoli products.

8.4.3 TMR Considerations - Upstream

In sites that have significant concerns over opening inbound/outbound ports to a large number of upstream (external) systems, deploying a small TMR outside of the firewall may be worth considering. This adds some additional complexity to the deployment as sufficient system, disk, and memory requirements must be accommodated on this machine. The attractiveness of placing a TMR externally is relatively clear in that only one external machine needs to communicate to one internal system. If this is done, the TMRs on both sides still need the `odadmin set_port_range` command run on them. Also the internal (downstream) TMR and the external (upstream) TMR cannot exchange resources, or the internal TMR will attempt to communicate with the external machines directly.

Tivoli Software Distribution to the external systems becomes a two-step operation:

1. The internal TMR must distribute a file package to the external machine.
2. The external TMR must have defined tasks to watch for file package arrival and then be able to run an after script to distribute to the subscribed managed nodes.

The external nodes must use the TEC Secure Logfile adapter, so that all events are sent to the external TMR for logging. Then the external TMR will send (through a notice group subscription being setup) these notices to the internal TMR for action by support groups.

In this case, the considerations from the section before this for ports and firewall/router strategies should be used in addition to port strategies outlined here.

8.4.4 TCP Connection Source Filtering

Although there is this concept of TMR server and TMR client from a Tivoli management perspective, there is no distinction as to who can initiate a TCP connection. Filtering on direction of connections, therefore, does not work very well unless, of course, the roles of TMR managed nodes is restricted.

One can, for instance, configure managed nodes and expect them only to initiate client TCP connections to remote ORBs or remote repeaters. Either the destination port used should be port 94 or for IOM, the one sent by the repeater in the DKEY. Repeaters choose ports from the selected port range. This knowledge can be used to toss out any other initiated client connections.

8.4.5 Network Address Translation (NAT)

Tivoli 3.6 supports Network address translation but with significant specific guidelines. With the Internet explosion, there is a large proliferation of Internet addresses worldwide. The usual practice is to assign a globally unique address to each host that uses TCP/IP. This assignment is administered by a central Internet authority.

Enterprises using TCP/IP for internal connectivity have been assigning IP addresses for the internal Networks without actually getting this space assigned by the Internet authorities. This causes a problem. Assigning globally unique addresses to every TCP/IP host on the planet means the Ipv4 address space will be exhausted in the near future. We will not, at this time, address Ipv6 address space, as Hosts within enterprises are being partitioned into public and private address spaces. By partitioning this way,

companies can manage their internal Networks on TCP/IP conserving the unique global addresses only for those hosts that must be on the real Internet. Hosts on private Networks that need an Internet address, therefore, have to depend on some address translation schemes to present themselves uniquely on the Internet.

In practice, a customer of an Internet Service Provider will be provided a Class C(/24 or /28) network for their external (upstream) network. This same customer may or may not have internal (downstream) a registered internal network IP address range. If not, addresses, such as 32.0.0.0/8 or 192.9.200/24 or 10.0.0.0/8, which are parts of the IANA reserved/special use IP address blocks, may be utilized inside of a customer network. RFC 1597 and RFC 1631 provide detailed information about Network Address Translation devices and IP Address Allocation for Private Internets.

Network address translation devices (NAT) act as a conduit between these two spaces. NAT devices typically create virtual address routers between the two spaces along with the additional ability of transforming IP addresses and ports. By deploying NAT on private network boundaries, unregistered hosts can be made visible on the Internet. NAT devices, therefore, also act as firewalls.

Often NAT devices are physically distinct from port filtering-based firewalls. Some NAT devices serve dual roles of address conversion and also do port mappings. There are also NAT/Firewalls that do Port Address Translation (PAT), but PAT devices are neither tested, nor supported by, Tivoli at this time.

8.4.5.1 NAT Requirements for Tivoli Enterprise

The Tivoli 3.6 general requirements for NAT support are as follows:

- The NAT mappings from private to public addresses must be static, that is, the mappings do not change dynamically. In other words, there is a 1-1 permanent mapping between private addresses to public IP addresses (no time slicing/sharing of public addresses).
- The NAT device acts as a proxy DNS server to ensure that client systems can resolve host name addresses on both sides of the NAT address spaces.
- The NAT device acts as a router to route IP traffic between the public and private networks.

NAT firewalls can be placed between any two TME components. There are further restrictions and limitations for NAT support based on its placement. Three separate cases come up:

- NAT devices between TMR server and managed nodes or between any two managed nodes.
- NAT between gateways and endpoints.
- NAT between inter-TMR servers.

The rest of this section addresses these cases individually.

8.4.5.2 NAT between Two Managed Nodes

When a managed node is installed and activated for the first time, its ORB connects to the TMR server. The server's odlist keeps track of Network addresses and other properties of all such managed nodes in its TMR. The server distributes this odlist to all managed nodes in the TMR when they come up.

Thus, if the server sees a mapped address for a NAT mapped node, all other managed nodes associate this particular (NAT-mapped) managed node's presence in the TMR only through this mapped address. This restricts the NAT-mapped managed node to ONLY be manageable from points that are across the NAT box. In Figure 124, the TMR server and managed nodes, MN1-MN3, reside on the same (north) side of the NAT device. Only these nodes may run methods on a managed node MN4 (on the south side of the NAT box). MN5 cannot run methods on MN4 (since MN5 is on the opposite side of the TMR server).

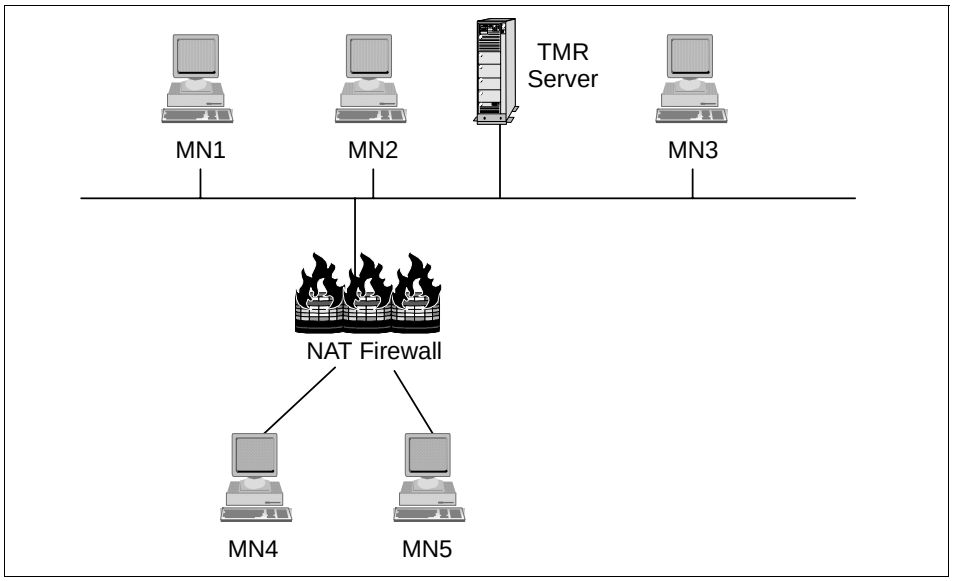


Figure 124. Managed Nodes Separated by NAT Device

IOM across NAT

For processing IOM communications across NAT devices, the DKEY is passed to the remote method. In Release 3.6, the DKEY now also contains the host name of the IOM connection server, which is usually the repeater. The IOM client connection node now resolves the host name of the repeater (assuming the NAT box does name resolution) to establish the IOM connection. Only if the name resolution fails, does the client resort to the IP address embedded in the DKEY. This new feature allows MDist repeaters and their destination targets to reside on either side of the NAT box.

8.4.5.3 NAT between Gateway and TMA Endpoint

With Release 3.6, we have the ability to implement NAT between gateway and endpoints. Earlier, gateways and endpoints had to reside in the same address space. Now, one can separate gateways and endpoint address spaces by placing NAT devices between them.

This is useful because we can now manage endpoints that reside in separate name spaces from one single TMR. This is usually the case when there is a TMR with a centralized server and gateway set up is in one domain while managing far-flung endpoints in multiple domains. In such environments, NAT is essential as a security measure and also needed to keep the respective private domain addresses unique.

The additional requirements, in this case, over and above the general requirements for NAT, are the following:

- Users need to provide fully qualified host names for gateways for endpoints to resolve.
- Users need to write a gateway selection policy script to assign gateways to endpoints.
- The NAT box must forward UDP protocol datagrams across.
- When starting up the TMA agent daemon/service, users must provide initial gateway information (`-g` option for `lcfd`) as fully qualified host names.

TMA Endpoint Login across NAT

This section describes the TMA endpoint login mechanism across a Network address translation (NAT) device. Consider the TMR server and gateways to reside in one IP name space domain called the server domain. We will call this domain name space `dev.server.com`. This domain appears north of the NAT device.

TMA endpoints running the TME3.6 Tivoli Management Agent can be in different client domains. These endpoints reside south of the NAT device and log into the server domain through the NAT device. Figure 125 on page 302 depicts the initial login sequence for an endpoint residing south of the NAT device.

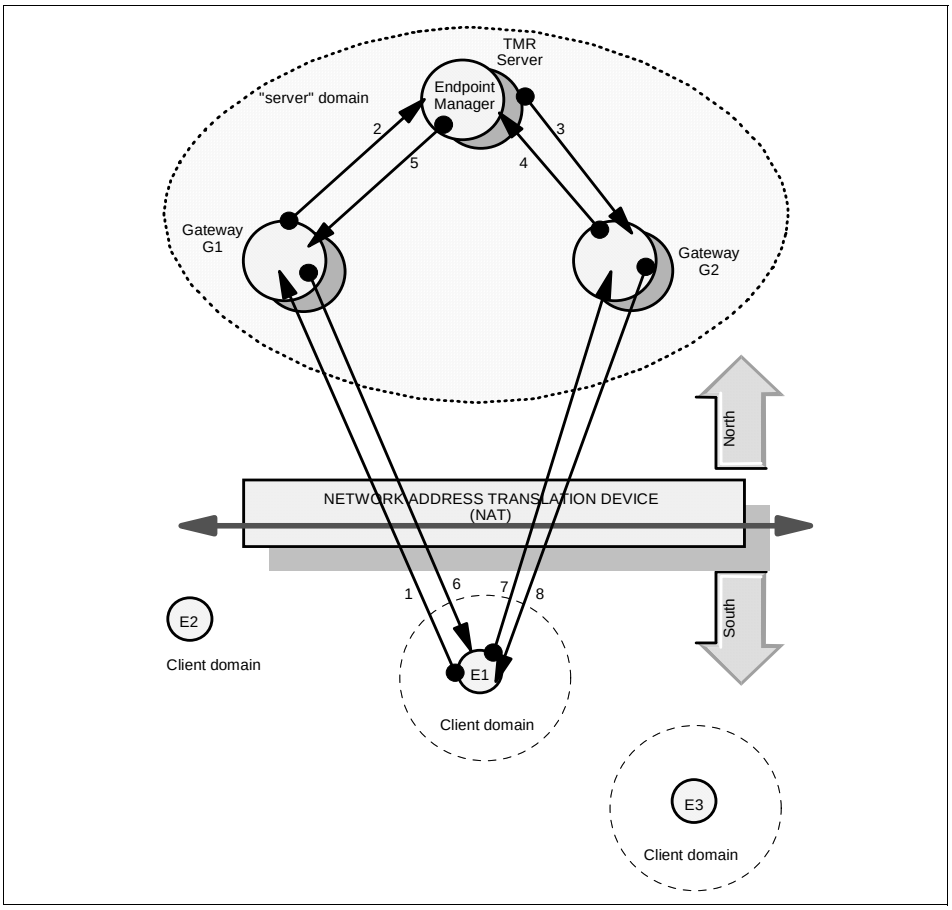


Figure 125. TMA Endpoint Login Across NAT Device

An endpoint has the Tivoli Management Agent (TMA) service (daemon) installed on it. It initiates the initial login sequence (#1 in Figure 125). The TMA service attempts to log into a gateway by sending UDP datagrams. Details about the login gateway IP address and port are provided from the following:

- The command line where the TMA application service is launched from.
- A last.cfg file in the TMA installation area that provides this information of last known initial gateways saved from prior logins.
- A UDP broadcast datagram that is received by all hosts residing on the client endpoints subunit.

In some environments, the third option may not be possible since the NAT box may not entertain broadcasts across its two domains.

On receiving the initial login request, a gateway G1 checks to see if this endpoint is already assigned to itself. If not, it forwards this request (#2) to the endpoint manager that resides on the TMR server. The endpoint manager executes two policy scripts. It first runs the allow_install_policy script to determine whether this endpoint requesting a log-in is permitted to log into the TMR. If the endpoint is eligible to login, the endpoint manager runs the select_gateway_policy script. This script echoes a list of gateway OIDs assigned to service this endpoint.

The endpoint manager runs a method on each of the gateway OIDs and picks one as the assigned gateway (#3 and #4).

This assigned (and failover) gateway IP information is then sent back to the login gateway G1 and, subsequently, to the logging endpoint (#5 and #6). Normally, this information is in the form of IP addresses.

The endpoint has now found its bearings. It logs (normal login) into the assigned gateway G2 (#7). The assigned gateway G2 already knows about this endpoint, accepts the normal login (#8), thus completing the entire login sequence.

The real IP addresses sent down (in #5 and #6) have no significance south of the NAT device. The assigned gateways in the select_gateway_policy script detail the gateway's fully qualified host name (FQHN) of the gateway along with the OID. This is done by appending the FQHN to the OID, the strings separated by the "|" (pipe) symbol. Note, since the select_gateway_policy script is tokenized on white spaces, no spaces should appear between the OID, the pipe symbol, and the FQHN of the gateway.

An example is a gateway paris fully qualified as `paris.dev.server.com` with an OID `123267682.1.529`. It will appear in the select_gateway_policy script as follows:

```
/****************************************/
#!/bin/sh
echo "123267682.1.529|paris.dev.server.com ""
exit 0
/****************************************/
```

Once a selection of gateways with the above format is listed in the policy script, endpoints will receive the FQHNs (instead of IP addresses) for

assigned (and failover) gateways during steps #5 and #6 of the login mechanism. The endpoints will retain these FQHNs.

8.4.5.4 NAT between TMR Server to Another TMR Server.

There is very little one can do today with NAT devices spread between two TMRs. The local odlist maintained by one TMR is exchanged with the remote TMR during the interconnection process.

Most likely, the managed nodes in one TMR will not be NAT-mapped within their own TMR, the odlist exchange will exchange the unmapped addresses. These are useless for inter-TMR connectivity among two dispatchers across NAT.

However, if all managed nodes of both TMRs are mapped (statically) across NAT devices spanning across the two TMRs, such that both TMR servers see the same addresses for all managed nodes they manage, some management of hosts across NAT-separated TMRs may be possible.

This has not been tested. The case study in 8.5, “Case Study 1 - Hub to Remote Through Firewalls” on page 304 covers this hypothetical scenario.

For now, if we at least map the TMR servers (again with static mappings) across NAT, then, currently, one TMR will be able to get IOM channels and MDist to operate correctly across the TMRs. The reason is because MDist repeaters exist on either TMR servers, and if the subscriptions between the two TMRs are handled correctly, users are able to push profiles across the TMR. There is an example of this in the case study.

8.5 Case Study 1 - Hub to Remote Through Firewalls

This case study describes an installation at hypothetical companies with a particular focus on connectivity through firewalls. Company details mentioned here are purely fictitious, and any resemblance to real-life companies is coincidental. The installation architecture is oversimplified to mainly focus on the firewall aspects.

Information Management Inc. (IMI) is a company that provides system management services to other companies. They are headquartered in Dallas, Texas with branches in San Francisco and New York. They provide system management and information technology services for clients First Town Bank of Boston (FTBB) and Eternal Insurance Inc. in Hartford, Connecticut.

IMI uses Tivoli to manage their customers as well as to manage their own sites. IMI offices have their own public and private IP addresses, and so do their customers.

IMI headquarters in Dallas provides a hub TMR. Local IMI managed nodes and endpoints connect to this TMR. All hosts in the Dallas office have registered IP addresses. The Dallas office configuration is illustrated in Figure 126.

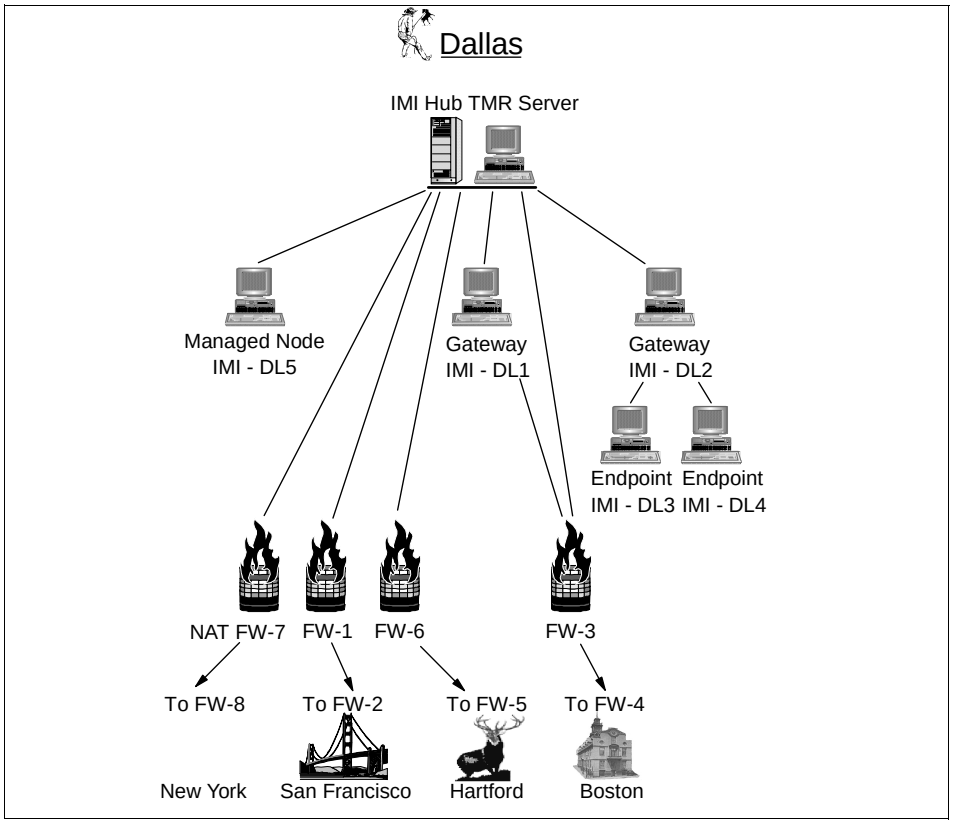


Figure 126. Hypothetical Internet Architecture - Dallas Hub

IMI is connected to San Francisco through the Internet. Managed nodes and endpoints in San Francisco have public IP addresses and are part of this TMR. The IMI office in New York, however, uses private IP addresses and has its own local TMR. New York is connected to Dallas using private leased lines.

Figure 127 illustrates the San Fransisco and New York networks.

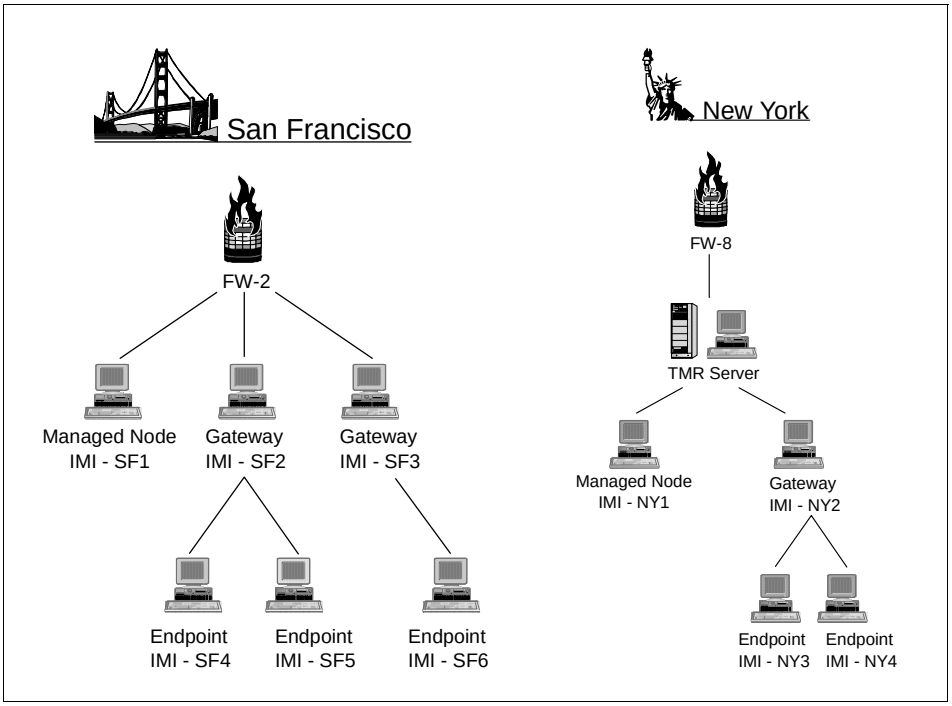


Figure 127. Hypothetical Internet Architecture - SF and NY Offices

Clients, First Town Bank of Boston (FTBB) and Eternal Insurance Inc. in Hartford (EII), both have private IP spaces. IMI also has their own endpoints that reside in the customer premises. These networks are illustrated in Figure 128.

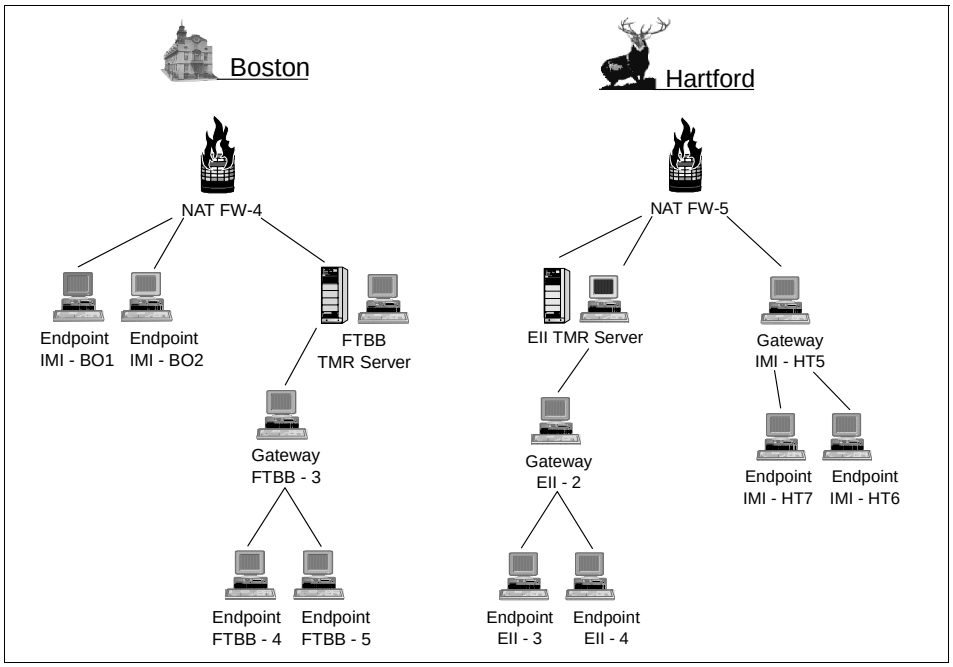


Figure 128. Hypothetical Internet Architecture - Boston and Hartford

Although IMI expects to manage each client locally, it also expects to perform some management tasks (for example, distributions) from the IMI headquarters. Each client TMR is connected to the hub TMR in Dallas. The hub and spoke TMR spaces are delineated by Network-filtering firewalls and NAT devices, as needed.

Note that for simplicity in the preceeding diagrams, network level details, such as LANs, bridges and routers, are not shown unless necessary in the context of the discussion. Network connectivity is shown only from a Tivoli perspective.

For the sake of this discussion, executing methods on endpoints is considered equivalent to invoking the method on the controlling gateway's ORB. The downcall from there is handled by the gateway.

Nomenclature for host names use the convention: *company acronym - location - number*.

Example: IMI-DL2 is an IMI owned resource located in Dallas, while IMI-BO5 is an IMI resource in Boston. The location is omitted for EII and FTBB, which, in this case study, are assumed to have single sites.

The following observations can be made for this setup:

- The IMI-HQ TMR has managed nodes and endpoints that are local (Dallas) and remote (San Francisco) to the TMR server. The SF office has public addresses. All managed nodes/endpoints in SF come into the TMR using the Network filter firewall FW-1.
- Hosts, such as IMI-DL5, can source file packs and act as repeaters that distribute to managed nodes and endpoints in SF. The gateways IMI-SF1 and IMI-SF3 act as repeaters for endpoints IMI-SF4 to IMI-SF6.
- FW-1 and FW-2 play the role of upstream firewalls that control traffic between the hub and SF. They do this by filtering packets based on source and destination addresses as well as port range selections on managed nodes IMI-TMR, IMI-DL5, IMI-SF1, and gateways IMI-SF2 and IMI-SF3.
- The New York office has private IP addresses. Firewall FW-2 (NAT equipped) maps these addresses statically to public addresses. The IMI-HQ-TMR hub server connects the NY-TMR server into inter-regions. After the connections, NY-TMR exchanges its odlist with the hub server. This odlist contains unmapped addresses for managed nodes in NY.

The address for the NY-TMR server can be changed to the mapped address by doing an `odadmin region change_region <mapped_addr>` command at the hub TMR. This allows the hub TMR to communicate with the NY TMR server. Note that this can only be done for the NY-TMR server and managed nodes IMI-NY1/IMI-NY2 cannot be managed from the hub TMR in Dallas.

Nonetheless, profile distributions between the hub TMR and NY TMR can be made possible by doing the following:

1. Create a profile manager PM-NY in the NY-TMR.
2. Subscribe managed nodes and endpoints in NY (IMI-NY1 to IMI-NY4) as subscribers to this profile manager PM-NY.
3. Create a profile manager PM-DL in the hub TMR in Dallas.
4. Subscribe PM-NY to PM-DL.
5. Distribute profiles created in PM-DL down to all levels. Distributions across regions use TMR servers as repeaters. Thus, distributions from PM-DL will use the repeaters at either end of the NAT box. Once PM-NY gets incoming records, the repeater in NY-TMR will push down to the leaf nodes IMI-NY1 to IMI-NY4.

Firewall FW-8 does address and port filtering to make sure that the only nodes communicating between the TMRs are the servers. Both, clients FTBB and EII have private address spaces, and thus, have the same restrictions

that the nodes in NY have with respect to its managed nodes and servers connecting to the hub TMR.

As such, in this environment, we only allow distributions to flow from the hub TMR to IMI customer's nodes in Boston and Hartford. All other management of these hosts have to be done from their local TMRs.

The IMI owned endpoints at FTBB Boston, though, are more manageable. Endpoints IMI-BO2 and IMI-BO-3 have the ability to login across the NAT box to the IMI gateway IMI-DL1 and merge as managed clients of the hub TMR. Other gateways in the DL TMR can act as assigned and fail-over gateways. Firewall FW-3 should not do any port filtering for endpoints IMI-BO-2 and IMI-BO-3.

The EII TMR setup is similar to FTBB except that here, the gateway for the IMI endpoints IMI-HT5/HT7, resides in the private EII address space. This gateway is mapped across the NAT box FW-5, and it is on the same side of the endpoints IMI-HT6 and IMI-HT7.

There are several reasons for doing this as opposed to what was done for IMI owned endpoints at FTBB. For instance, assume IMI has a security policy to disallow UDP packets into the hub TMR from across the Internet. Endpoints need UDP for initial login. With this restriction, endpoints, such as IMI-BO-2, cannot successfully execute their initial login across the firewalls.

Another reason is that, in the EII case, the gateway IMI-HT5 (also the repeater for the IMI endpoints) allows distribution traffic across the Internet to be minimized by being a focal distribution point for the IMI-HT endpoints. As before, firewalls FW-3 and FW-6 must not make any assumptions about ports used by IMI endpoints at their customer sites.

8.6 Case Study 2 - Dual TMR Setup with Firewalls

Figure 129 on page 310 illustrates one other successful methodology to deploy TMA 3.6 in an Internet firewall environment that many customers are using. It is included to give another perspective of TMA deployments. In this example, there are two separate TMRs due to customer resiliency issues. In the setup shown, there are no TMA endpoints logging in through a firewall, and all machines are managed nodes in the DMZ network.

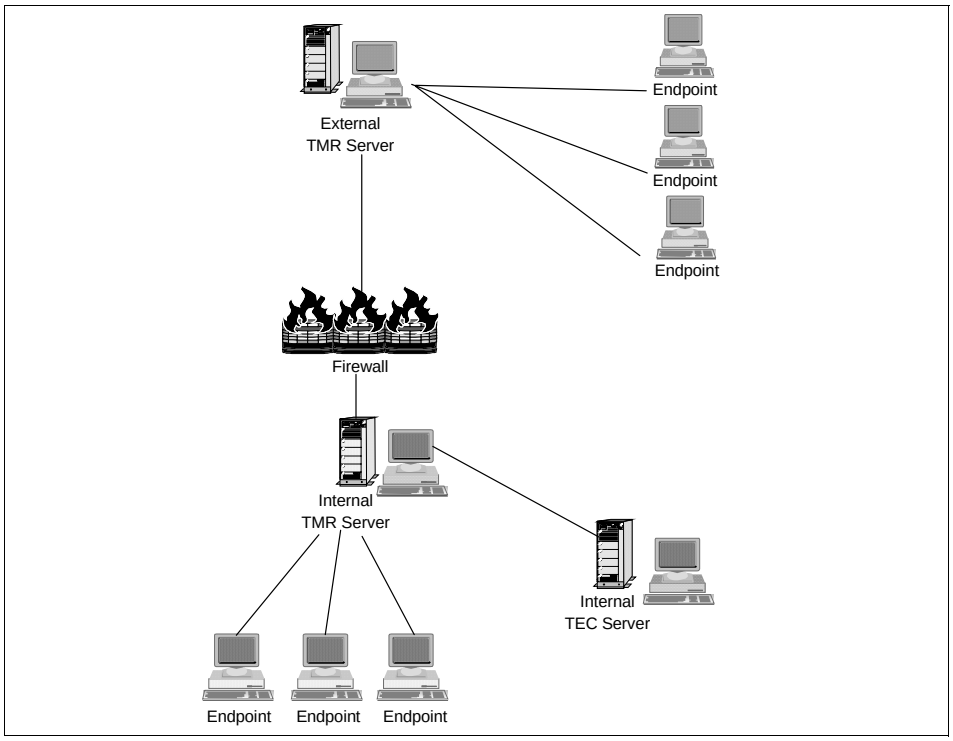


Figure 129. Dual TMR Implementation Across Firewall

One TMR is located outside the firewall and the other is inside. Note that the internal and external TMRs do not exchange resources. In this example the only purpose of the external TMR is the use of distributed monitors so no resource exchanges are needed. External TEC events are sent to the internal TMR TEC console for monitoring.

Machines being managed only send monitoring events to the (external) TMR server. They need no direct route to the internal network. Internet inbound and outbound traffic is blocked on these systems except for Tivoli and normal service ports.

The internal and external TMRs could have Tivoli Cross-Site security monitors running for additional monitoring alerts.

Each TMR server has a default route to the firewall which has ports 94 and the range specified in the TMR. These are bi-directionally enabled only between the TMR servers. The external TMR server has internet inbound and outbound traffic blocked and a static route only to the internal TMR server

plus select machines within DMZ. The internal TMR server has the default route to firewall which has ports 94 plus the TMR range bi-directionally enabled only between the TMRs. The internal TMR host is only allowed intranet traffic on the firewall and a static route only to the external TMR server (plus any internal required routes).

Chapter 9. RDBMS Interface Module (RIM)

The RDBMS Interface Module (RIM) is designed to allow Tivoli applications that collect or generate large amounts of data to store that data in third-party databases. The goal of RIM is to allow the applications to have a common set of APIs to get and store data. RIMs job is to convert the data provided through those APIs to the format used by the various database vendors.

9.1 Applications Using RIM

The Tivoli applications that store and use large amounts of data are being migrated to use external databases. At the time of writing, there are four applications that use RIM, but others are expected to follow. The current applications, and the way they use the database, are:

Tivoli Inventory — Collects and stores hardware and software information about TME managed nodes, PC managed nodes, and TMA Endpoints.

Tivoli Enterprise Console — Collects, filters, and acts upon events coming in from resources across the enterprise.

Tivoli Software Distribution — Distributes software to endpoints. Software Distribution will write information to the database with every file package distribution and use the same configuration repository as Tivoli Inventory. This is an optional interface enabled by installing Software Distribution Historical Database modules.

Tivoli NetView — Manages Network resources and shows live topology. NetView has an option to use relational databases through RIM to record event data, collected measurements, and topology snapshots.

9.1.1 Applications Moving to RIM

The following are likely to be other applications that will use RIM:

Tivoli Distributed Monitoring — This product does not rely on RIM directly but may write data to the Performance Reporter database in future releases.

Tivoli User Administration	Creates and manages user and group system accounts. The current number of attributes for each managed user can be over 100.

9.2 Installing RIM

RIM is a component of the Tivoli Management Framework. RIM is installed with the Framework, and then each application that uses it creates the appropriate RIM objects. The application installation usually consists of two steps

- Creating the user ID and user tables in the RDBMS server.
- Creating the RIM component for the application to connect to the RDBMS server.

Using Oracle 7.x on Windows NT

The RIM support for Oracle on Windows NT is linked against `ORANT71.DLL`, which is the Dynamic Link Library supplied with Oracle 7.1. For RIM support for Oracle on Windows NT to work, this file must be installed and must be in the local path. Unfortunately, the name of this library was changed in Oracle 7.2 and Oracle 7.3, and `ORANT71.DLL` does not get installed with Oracle 7.2 and Oracle 7.3. However, `ORANT71.DLL` is available on the Oracle 7.2 and Oracle 7.3 installation media.

From the Oracle 7.3 media, the `ORANT71.DLL` can be copied from the `NT_X86\V7\RSF72` directory to the `%ORACLE_HOME%\bin directory`. Two other DLL modules, `CORENT23.DLL` and `MSVCRT10.DLL`, are required for `wrimtest` to work with Oracle 7.3. These DLL modules can be copied from the Oracle 7.3.3 installation media after the Oracle 7.3.3 installation.

9.2.1 Creating Application Database Tables

The TME applications supply scripts to create the necessary tables, views, and users on the RDBMS server. Even though RIM means the application does not need to know the specifics of each database implementation, these scripts used to build the tables do need to be vendor specific. This means that each time RIM expands to support another database, there will need to be a product update that provides new scripts for that database. The tables and views that are created make up the databases for the applications that use RIM. The following is an example of the `tivoli_syb_admin.sql` script supplied by Inventory 3.6 for use with Sybase:

```
use master
go
create database inventory on master
go
alter database inventory on master = 20
go
sp_dboption "inventory", "trunc. log on chkpt.", true
go
sp_addlogin tivoli, tivoli, inventory
go
use inventory
go
sp_adduser tivoli
go
grant create table to tivoli
go
grant create view to tivoli
go
quit
```

Note

1. It is recommended that you work with an experienced DBA to set up your databases.
2. The scripts that define your application database and tables are created when you install your application, such as TEC or Inventory.
3. Define your own database volumes, for example, with Sybase, the default device is master (see above script). Use different device names for your database and log. This is more suitable for database recovery. Note that some Tivoli product scripts may specify master, and these should be changed.
4. Check that you are defining enough space for your environment. For example, check the number of machines that will be scanned by Inventory or the number of events that you expect with TEC.
5. The Tivoli user ID provided during installation is used as the database and table creator. This means, in most implementations, that the database should be created using the same account information as supplied during initial installation. If your installation needs to use a different user ID, use the `wsetrim` command to change the user ID. This command is used to change any of the RIM objects settings and will be explained in detail later in this chapter.

9.3 Understanding RIM

From the Tivoli application user's perspective, RIM is invisible. An application using RIM will gather and store data in the RDBMS, but there is no interaction or involvement by the user. For a detailed description of RIM and how it works with the Tivoli applications, refer to the *Using Databases with Tivoli Applications and RIM* redbook, SG24-5122

9.3.1 RIM Behind the Scenes

Several components work together to make communication through RIM possible. The client application uses RIM APIs to make a request to gather and retrieve data. The RDBMS_Interface translation layer receives the request from the client, looks up the RIM host, and sends the request to the RIM host. The third component is the vendor adaptor layer, which sends vendor-specific requests to the database.

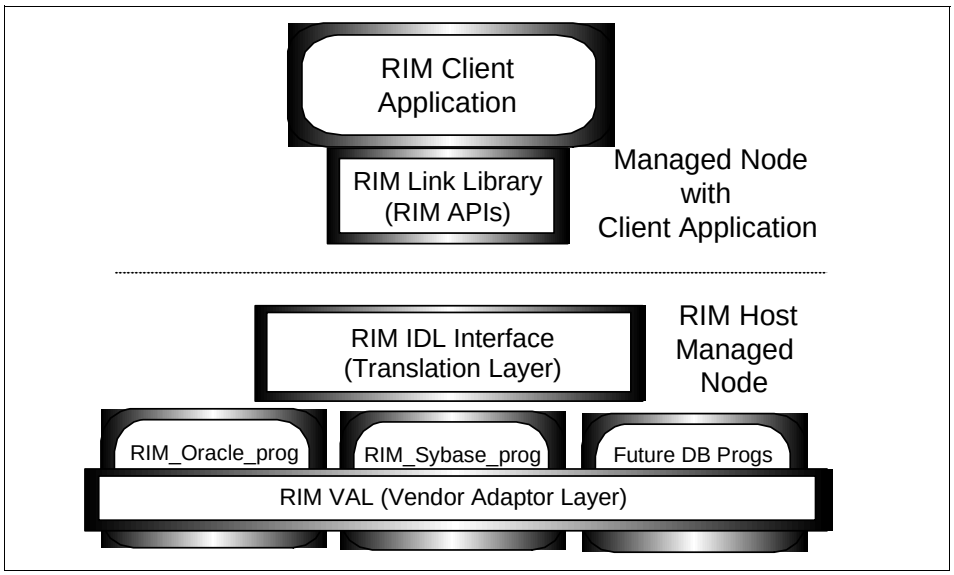


Figure 130. RIM Components

9.3.2 RIM APIs

There are several Tivoli APIs that an application can use to store and retrieve information from an external database. Using API calls allows the application to be database vendor unspecific. These APIs can be seen as IOM commands in a `wrimtrace` output on the RIM host when an application has been using RIM.

A list of RIM calls that can be seen using `wrimtrace` includes:

```
connect
iom_session
release
database
commit
rollback
execute_sql
quote_value
insert
insert2
update
update2
retrieve
retrieve2
delete
delete2
```

Refer to *Tivoli Framework Reference Guide* for more information about using `wrimtrace`. Note that `odadmin environ get RIM_DB_LOG` shows you where the log will be written, and that you have two tracing options besides `TRACE_OFF` which give database errors (ERROR) or the contents of the IOM packets (INFORMATION). When you change the trace level, be sure to kill the `RIM_databasename_PROG` process, as described in the manual.

9.3.3 RDBMS_Interface Translation Layer

The translation layer acts as the engine behind the RDBMS_Interface API. It performs two major tasks:

- Looks up the correct server with which to connect.
- Translates data passed through the API as an any into an SQL string involving operations on the data model tables, rows, and columns (database records and fields).

9.3.4 Vendor Adaptor Layer

The lowest layer of the RDBMS_Interface is the stub functions making calls to an actual vendor-specific library adopted for a given combination of RDBMS vendor and platforms. This layer consists of hooks to vendor-specific library calls. Each vendor instance of this layer is implemented as a separate dedicated C program.

9.4 RIM on Framework 3.6

This section looks at how RIM works in the Framework 3.6 release.

9.4.1 Creating RIM 3.6 Objects

RIM for Framework 3.6 includes support for four database products:

- Sybase
- Oracle
- DB2 (Unix only)
- MS SQL (NT Only)

These databases can be specified in the GUI or using the `wcrtrim` command. Details on how the `wcrtrim` and `wsetrim` commands work are included later in this chapter.

The common installation options used to define RIM objects are:

1. Database Vendor - either Sybase, Oracle, MS_SQL, or DB2.
2. RIM host (TME RDBMS access host) - A Tivoli managed node with client access to the RDBMS database server. This option can only be set through the command line. The dialog does not allow you to change or set this field. It automatically sets the RIM host option to the TMR server.
3. RDBMS User ID.
4. RDBMS Password.

Vendor specific options:

1. Database ID
2. Database Home
3. Database Server ID
4. Instance ID (DB2 only)

The following tables describe the RIM object parameters in more detail:

Table 11. RIM Installation Options

Install Option	Sybase	Oracle
Database Vendor	Sybase	Oracle (default on GUI)
TME RDBMS Access Host	RIM HostName[1]	RIM HostName[1]
Database ID	Name of the database that the application will use.[1]	$ORACLE_SID Oracle SID listed in `transnames.ora` file.
Database Home[2]	$SYBASE The value that Sybase sets in this environment variable.	$ORACLE_HOME The value that Oracle sets in this environment variable.
Database Server ID	$DSQUERY Sybase name for the server.[4]	$TWO_TASK Label in `transnames.ora` file.[5]
Database User ID	tec for Tec application and tivoli for Inventory application.	tec for Tec application and tivoli for Inventory application.
Password[6]	The password is for the database user. You'll be prompted for the password when using the `wcrtrim` command.	

NOTES:

1. Must be a managed node or the TMR Server.
2. The database name will depend on the application. By default, Inventory uses inventory, and TEC uses tec.
3. If the RIM host is the RDBMS, then this will be the default directory where the file resides. If the RIM host is not the RDBMS, then this value will be the name of the directory that you copy the file to. Use forward slash even when specifying a Windows NT home.
4. The name for the sybase server is usually in a UNIX environment variable called `DSQUERY`. This is a unique Sybase name for the server, not always the same as the machines IP address.
5. You do not need to enter a server ID for Oracle if the RIM host is the RDBMS.
6. The password is set automatically by applications, such as Inventory and TEC, so it does not appear in the dialog at install time. If you need to re-create a RIM object, make sure you know both the user and password for access to the database. Check this information with your database administrator. If password for the user ID is changed in the RDBMS, then use the `wsetrimpw` command to change the password in the RIM object.

Table 12. RIM Installation Options (Cont.)

Install Option	MS SQL	DB2
Database Vendor	MS_SQL	DB2
TME RDBMS Access Host	RIM HostName[1]	RIM HostName[1]
Database ID	Name of the database that the application will use.[2]	Name of the database that the application will use.[2]
Database Home[3]	The directory where MS SQL is installed, such as: `C:\mssql`.	$DB2HOME The value that DB2 sets in this environment variable.
Database Server ID	server name where MS/SQL Server is installed.	tcpip
Instance Home	not applicable.	$INSTHOME
Database User ID	tec for Tec application and tivoli for Inventory application.	tec for Tec application and tivoli for Inventory application.
Password[4]	The password is for the database user. You'll be prompted for the password when using the `wcrtrim` command.	

NOTES:

1. Must be a managed node or the TMR Server.
2. The database name will depend on the application. By default, Inventory uses inventory and TEC uses tec.
3. If the RIM host is the RDBMS, then this will be the default directory where the file resides. If the RIM host is not the RDBMS, then this value will be the name of the directory that you copy the file to. Use forward slash even when specifying a Windows NT home.
4. The password is set automatically by applications, such as Inventory and TEC, so it does not appear in the dialog at install time. If you need to re-create a RIM object, make sure you know both the user and password for access to the database. Check this information with your database administrator. If password for the user ID is changed in the RDBMS, then use the `wsetrimpw` command to change the password in the RIM object.

9.4.2 Client Application Communication with RIM 3.6

RIM performs the following actions using the values provided (see also Figure 131 on page 321):

1. Application 1 asks TMR A for the object ID of the required RIM host. The TMR looks in the name registry and finds that Application 1's RIM host is RIM Host X.
2. Application 1 requests an action of RIM Host X. This request is routed through TMR A.
3. RIM Host X uses the directory value for Database Home and looks at the appropriate configuration file. These are:
   - `interfaces` file for Sybase,
   - `tnsnames.ora` file for Oracle.
4. RIM Host X looks up how to contact the RDBMS server matching the name of the Server ID to an entry in the configuration file.
5. RIM Host X contacts the RDBMS server and performs the action for the database specified in the Database ID field. It accesses this database using the user ID and password that has been set for Application 1's RIM object.
6. Once the request is completed by the RDBMS server, RIM Host X passes the data through an IOM channel directly to that managed node.

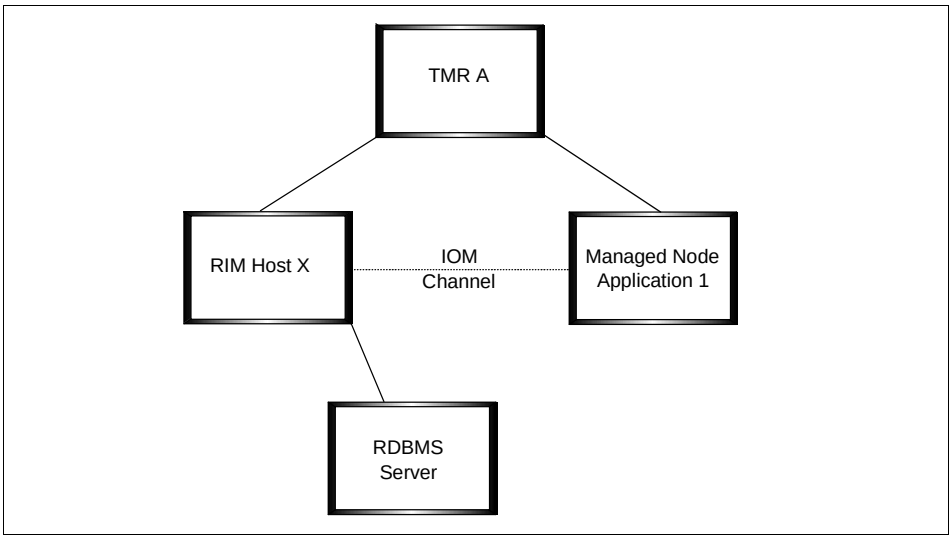


Figure 131. How an Application Uses RIM

9.5 Troubleshooting RIM

This section lists a series of activities you may wish to perform when working with RIM applications.

9.5.1 Finding the RIM Objects Defined in a TMR

The Tivoli name registry contains a RIM item that references all the RIM objects in the TMR. RIM objects from other connected TMRs will also be listed if the resource has been updated:

```
root@rh0255c:~# wlookup -ar RIM
inventory       1562489759.3.35#RIM::RDBMS_Interface#
root@rh0255c:~#
```

Figure 132. wlookup - Listing the RIM Objects in Your TMR.

Note

You can not use a RIM object from an interconnected TMR. The RIM Host must always be local within a TMR.

9.5.2 Displaying the Settings for a RIM Object

We use `wgetrim` to display RIM object settings:

```
root@rh0255c:~# wgetrim inventory
RIM Host:       rh0255e
RDBMS User:     tivoli
RDBMS Vendor:   Sybase
Database ID:    Inventory
Database Home:  /sybase/install
Server ID:      rh0255e
root@rh0255c:~#
```

Figure 133. Listing the Information for a RIM Object - wgetrim

9.5.3 Changing RIM Object Information

The `wsetrim` command is used to change RIM object information. It will only allow you to set the following parameters:

`[-n new_name]` The new name of the RIM object.

`[-d database]` The name of the database in the RDBMS server.

`[-u user]`	The name of the user that can operate on the database.
`[-H rdbms_home]`	The directory where the RDBMS vendor-specific configuration file resides.
`[-s server_id]`	The name of the RDBMS server.
`[-I instance_home]`	The directory where the DB2 instance was created.
`rim_name`	The current name of the RIM object you are changing. This parameter is always required.

Note

The RIM host name and the RDBMS Vendor options cannot be changed by using the `wsetrim` command. If you need to change either of these two options, you must delete the existing RIM object and recreate it.

The following screen shows how to change the name of the RIM object from `inventory` to `appl2` using the CLI:

```
root@rh0255c:~# wsetrim -n appl2 inventory
root@rh0255c:~# wlookup -ar RIM
appl2 1562489759.3.35#RIM::RDBMS_Interface#
root@rh0255c:~# wgetrim appl2
RIM Host:       rh0255e
RDBMS User:     tivoli
RDBMS Vendor:   Sybase
Database ID:    Inventory
Database Home:  /sybase/install
Server ID:      rh0255e
```

Figure 134. Changing a RIM Object Name - wsetrim

9.5.4 Changing the RIM Host Machine Name

You can only have one RIM host for each application in a TMR. If you wish to change the RIM host, then all RIM objects using the same RIM host have to be deleted and re-created specifying the new RIM host name.

To delete a RIM object you can use `wdel`. Running `wlookup` after `wdel` confirms that the object has been deleted.

```
root@rh0255c:~# wlookup -ar RIM
appl2 1562489759.3.35#RIM::RDBMS_Interface#
root@rh0255c:~# wdel @RIM:appl2
root@rh0255c:~# wlookup -ar RIM
root@rh0255c:~#
```

Figure 135. Deleting a RIM Object - wdel

When you re-create the RIM object, you use the `wcrtrim` command. You are creating a new RIM object; so, you will be prompted for the user's database password.

```
root@rh0255c:~# wcrtrim -v Sybase -h newhost -d Inventory -u tivoli
-H /sybase/install -s rh0255e inventory
RDBMS password:
root@rh0255c:~# wgetrim inventory
RIM Host:       newhost
RDBMS User:     tivoli
RDBMS Vendor:   Sybase
Database ID:    Inventory
Database Home:  /sybase/install
Server ID:      rh0255e
root@rh0255c:~#
```

Figure 136. Creating a New RIM Object - wcrtrim

9.5.5 Troubleshooting Example: Failure to Connect with RDBMS

A distribute failure was indicated to the administrator with the dialog shown in Figure 137 on page 325:

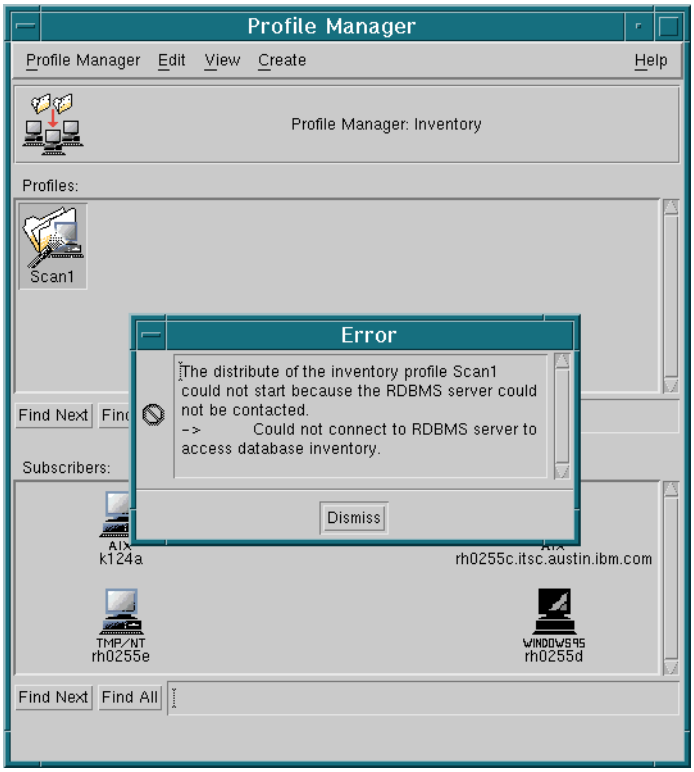


Figure 137. RIM Connection Failure Message in the Desktop

This problem was created by changing the settings of the Inventory RIM object. A `wgetrim` shows that the Database Home directory has been changed to `/tmp`. The interfaces file for Sybase will not be found there.

```
#wgetrim inventory
RIM Host:       k124a
RDBMS User:     tivoli
RDBMS Vendor:   Sybase
Database ID:    inventory
Database Home:  /tmp
Server ID:      rh0255e
#
```

Figure 138. Example of wgetrim

To check whether or not a RIM object can connect to the database, use the `wrimtest` command. If you get a Session Opened message, then RIM connected to your database. You can then execute a retrieve or any SQL

command against the database to view data. Figure 139 is an example of using wrimtest:

```
[root@itso2]/> wrimtest -l tec
Resource Type  : RIM
Resource Label : tec
Host Name      : itso2
User Name      : tec
Vendor         : Sybase
Database       : tec
Database Home  : /data/sybase
Server ID      : ITSO2
Instance Home  :
Opening Regular Session...Session Opened
RIM : Enter Option >g
Table Name > tec_t_evt_rep
Enter <Field Name> [/n] [/s /l /f /d [<Value>]Editor? [y/n] [Default n] >
1 > msg
2 > class
3 > origin
4 >
Where Clause >
Retrieve) Num of Rows [0] >
Row 0
        msg : (0) Distributed Monitoring TACF_Monitors/TACF Files on
host hptmp9-ep Wed Dec  2 10:12:00 CST 1998 CST

Status: >>> critical <<<

TACF Files (TracingFile)  () Greater than 1(Previous: 1.38 KB
Current: 1.38 Effective: 1.38)

        class : (0) Tacf_LogFileSize
        origin : (0) 146.84.32.34
RIM : Enter Option >x
Releasing session
[root@itso2]/tmp>
```

Figure 139. Example of wrimtest

This example showed using wrimtest to list data from the TEC Event Repository table (tec_t_evt_rep) showing columns msg, class, and origin. Refer to the *Tivoli Framework Reference Manual* or the wrimtest main page for details on using the wrimtest command.

Figure 140 on page 327 shows an odstat output from TEC not starting up because the RIM object is not configured correctly. Since this is a very large output, many lines have been deleted to just show sample messages. The thread in error is 13166 near the bottom:

```
n_active = 14  n_free = 186
   tid type    ptid State  StdO  StdE    Start   Err    Method
     1 SYS
     7 O+bhdoq          run     0     0 Mon14:16        1998892590.1.158#TMF_Scheduler::sch
eduler# start
    11 O+bhdoq          run     0     0 Mon14:16        1998892590.1.616#SentryEngine::engi
ne# run_engine
    86 SYS
   105 O+hdoqs          run     0     0 Mon14:17        1998892590.1.530#TMF_UI::Extd_Desk
top# uiserver

---- history ----

 13141 O+bhdoq         done     6     0 11:18:27        1998892590.1.885#Tec::Se
rver# start_server
 13142 O+hdq1-13141  done     6     0 11:18:27        1998892590.1.885#Tec::Serv
er# _set_state
13147 O+hdq1-13142  done     6     0 11:18:27        1998892590.1.881#Tec::Inst
anceManager# update_state
 13148 O+hdq1-13141  done   290     0 11:18:28        1998892590.1.885#Tec::Serv
er# get_backrefs
13151 O+hdoq1-13141  done     6     0 11:18:28
1998892590.1.179#TMF_Administrator::Configuration_GUI# refresh_member
 13152 O+hdq  1-105  done  5100     0 11:18:28        1998892590.1.865#TMF_UI::P
resentation# get_icon_info
 13153 O+hdq 1-4509  done  5100     0 11:18:28        1998892590.1.865#TMF_UI::P
resentation# get_icon_info
*13154 O    1-13141  done     0     0 11:18:28 NO_METHOD  1998892590.1.881#Tec::In
stanceManager# refresh_member
 13155 O+   1-13141  done    47     0 11:18:28        0.0.0 get_host_location
 13156 O+   1-13141  done    47     0 11:18:28        0.0.0 get_host_location
 13157 O+hdoq1-13141  done    43     0 11:18:28
1998892590.1.348#TMF_ManagedNode::Managed_Node# install_directory
 13158 O+   1-13157  done    15     0 11:18:28        0.0.0 get_oserv
 13159 O+   1-13157  done    26     0 11:18:28        1998892590.1.2 query insta
ll_dir
 13160 O+hdoq1-13141  done    24     0 11:18:29
1998892590.1.348#TMF_ManagedNode::Managed_Node# interpreter
 13161 O+   1-13160  done    15     0 11:18:29        0.0.0 get_oserv
 13162 O+   1-13160  done     7     0 11:18:29        1998892590.1.2 query inter
p
*13163 O+ho 1-13141  done   352     0 11:18:29  e=12  1998892590.1.885#Tec::Serv
er# is_dataserver_running
 13164 O+   1-13163  done    15     0 11:18:29        0.0.0 get_name_registry
 13165 O+hdoq1-13163  done   105     0 11:18:29        1998892590.1.26 lookup
*13166 O+hdoq1-13163  done   352     0 11:18:29  e=12  998892590.1.905#RIM::RDBM
S_Interface# RIM_iom_session
 13167 O+hdq1-13141  done     6     0 11:18:29        1998892590.1.885#Tec::Serv
er# _set_state
```

Figure 140. Example of RIM Call in odstat Output

Figure 141 on page 328 shows the `wtrace` output for the same error. The complete output is too large, so we have just shown the thread that actually produced the error, 13166. The `RIM_iom_session` method failed because RIM could not make a connection to the RDBMS:

```
loc-ec 13166 M-hdoq  1-13163       54  e=12
      Time run:    [Tue 01-Dec 11:18:29]

      Object ID:   1998892590.1.905#RIM::RDBMS_Interface#
      Method:      RIM_iom_session
      Principal:   root@itso2.dev.tivoli.com (-2/-2)
      Helper pid: 50532      Path:        /data/usr/local/Tivoli/bin/aix4-r1/TAS/RI
M/RIM_Sybase_prog

      Input Data: (encoded):

             {
               40 "0x32 0x34 0x35 0x34 0x39 0x38 0x36 0x38 0x33 0x30 0x49 0x33
               0x35 0x31 0x30 0x30 0x49 0x32 0x30 0x30 0x30 0x34 0x62 0x34 0x38
               0x32 0x30 0x30 0x30 0x34 0x62 0x34 0x38 0x20 0x69 0x74 0x73 0x6f
               0x32 0x00 "
             }

      Results: (encoded):
             "Exception:UserException:SysAdminException::ExException:RIM:
             :ExRIMError:RIM::ExRIMConnectFail"
             {
               "Exception:UserException:SysAdminException::ExException:
               RIM::ExRIMError:RIM::ExRIMConnectFail" "rim_errors" 1 "Could
               not connect to RDBMS server to access database %7$s." 912532709

               {
                 0
               }
               "tec" 0
             }
```

Figure 141. Example of RIM Error in wtrace Output

If we now look at the RIM trace log from the same exception, we will see a more detailed description of the problem.

```
00050532 [Tue Dec  1 11:44:50 1998] Connection ID: 0, Operation: val_connect, DB Call:
val_connect
0õÐ 8˜ ?X lH0õ¨ðT.ÆðT0õÐðTÑ.ï¸ðTÑ.çHðTDB-Library Error: Could not open interface
file.DB-Library Reports OS Error No such file or directory

00050534 [Tue Dec  1 11:45:33 1998] Trace Message - Connection ID:: Connecting to IOM
Channel
00050534 [Tue Dec  1 11:45:33 1998] Trace Message - Connection ID:: Beginning IOM Loop
00050534 [Tue Dec  1 11:45:38 1998] Trace Message - Connection ID::
IOM Command: RELEASE
row_param: <NULL>
rows: <NULL>
number1: 0
number2: 0
string1:
string2:
00050534 [Tue Dec  1 11:45:38 1998] Trace Message - Connection ID::
REPLY IOM Command : RELEASE Result : Success
rows: <NULL>
00050534 [Tue Dec  1 11:45:38 1998] Trace Message - Connection ID:: Ending IOM Loop
```

Figure 142. Example Output of RIM Tracing with wrimtrace

Here we can see the connect operation failed because it could not find a file or directory. This was because the Database Home field in the tec RIM object definition was incorrectly typed. After correcting the RIM object using `wsetrim`, the database connected successfully, as shown in the second half of the above screen.

9.5.6 RIM Specifics

When an exception is thrown, any RDBMS-specific error messages are included in the exception. Error messages are also logged to a file. Exceptions contain:

- RIM error message
- Database function that caused the error
- Return code from the database function
- Database error message

Before the exception occurs, a message is written to the RIM log. The default error log file is `/tmp/rim_db_log` but the location can be changed by creating a `RIM_DB_LOG` variable in the oserv environment by using the command:

```
odadmin environ set RIM_DB_LOG "pathname"
```

The command to trace RIM calls in versions of the Tivoli Framework from 3.2 and above is `wrimtrace rim_object <options>`. Where `rim_object` is the name of a valid RIM object, such as tec or inventory, and the valid options are:

`TRACE_OFF`	Turns tracing off
`ERROR`	Returns error codes
`INFORMATION`	Shows the SQL commands passed by RIM

To turn both `ERROR` and `INFORMATION` tracing on at the same time, concatenate them with the pipe symbol enclosed in double quotes. For example:

```
wrimtrace tec "ERROR|INFORMATION"
```

Note

In order to activate the last `wrimtrace` command, you must restart the RIM daemon. One method is to set the trace options, kill the daemon, and the next RIM action will restart the daemon. Remember to set tracing off and kill the daemon again when you have finished tracing. The RIM daemon name will be `RIM_<dbvendor>_prog` (ex. `RIM_Sybase_prog`). You may also see a `RIM_generic_prog` daemon running as well. Go ahead and kill that daemon as well.

9.6 Queries

Creating a query will allow you to perform functions, such as a search of the Inventory database and select machines, based on the attributes stored in the database when an Inventory scan was performed.

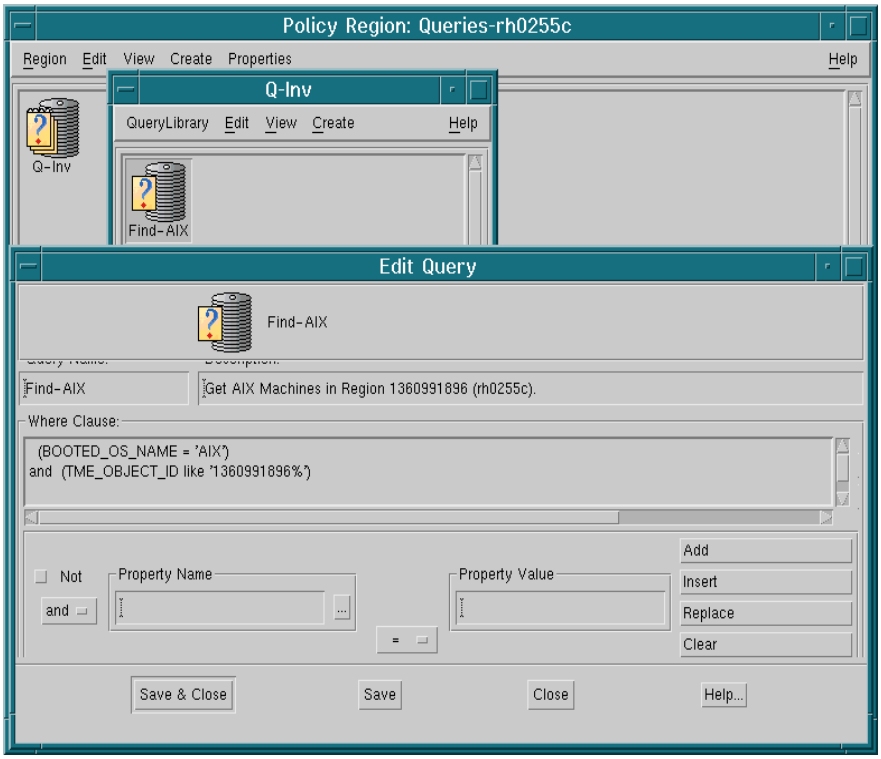


Figure 143. Creating a Query through the GUI

Figure 143 shows a query for finding the machines with the AIX operating system that exist in a specific TMR. The SQL wild card character is percent (%).

You may need this type of query when more than one TMR is using the same Inventory database. When performing the query while selecting subscribers, for either another Inventory scan or a software distribution, you will ensure that only one TMRs objects will appear in the Subscribers list.

Figure 144 on page 332 shows how to run the previous query when selecting machines for a Software Distribution:

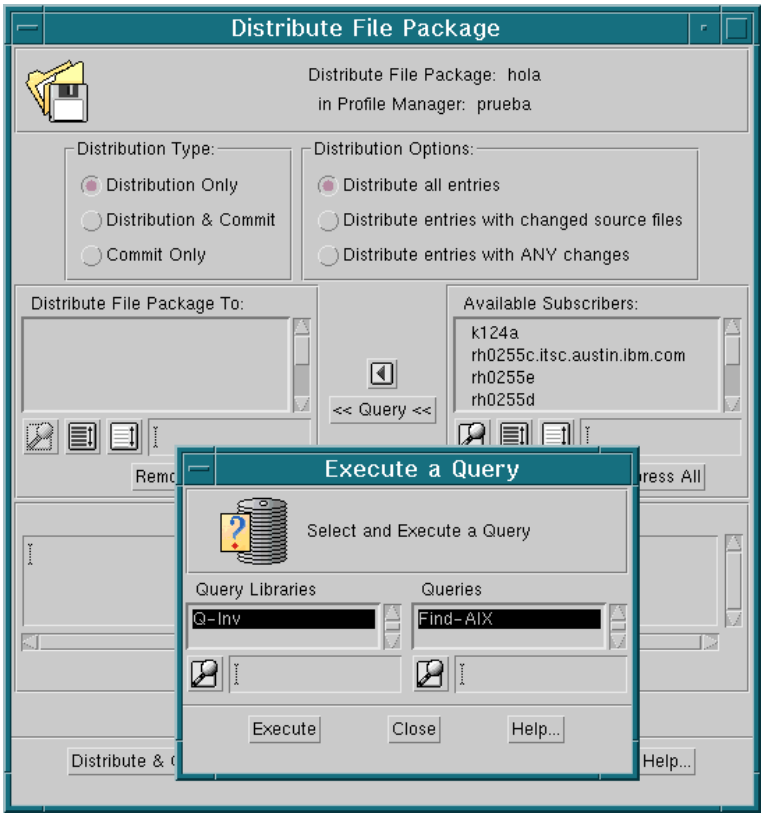


Figure 144. Running a Query to Select Subscribers for Software Distribution

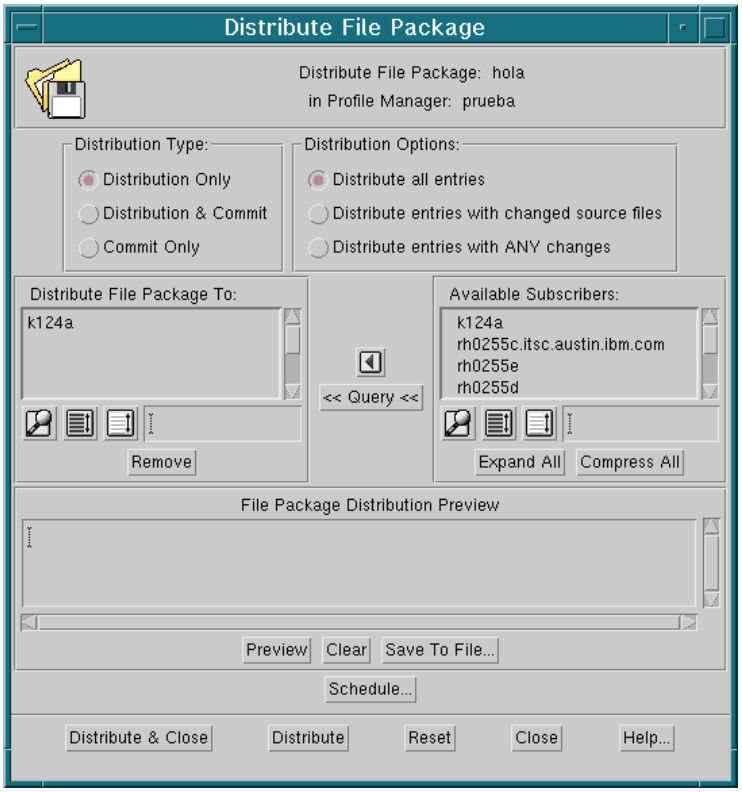


Figure 145. Subscribers Selected after Running a Query

Instead of selecting individual machines from the Subscribers list, you select the **Query** button, select the **Query Library**, and then the appropriate query. Once selected, you can Execute. Figure 145 shows that only k124a is an AIX machine in the 1360991896 region. You can execute multiple queries, each with different selection criteria. The newly-selected machines will be appended to the list for the distribution.

Note

You may experience problems if you try to create a query through the command line and then try to edit the same query through the desktop.

9.6.1 Queries with RIM 3.6

The Create Query dialog in Framework 3.6 allows you to specify the database that you want to query. As long as the RIM host knows about the

RDBMS server, you can query any database including a non-TME application's database.

9.6.2 Tivoli Roles Needed to Execute Queries

For a Tivoli administrator to be able to execute queries, they must have at least RIM_View authorization. If they will be updating information, they will also need RIM_Update authorization.

9.7 Designing Your Tivoli Environment for RIM

There are many considerations when designing an environment that will use RIM. Some brief notes follow that should help when considering your design:

- You can have one RIM host for each application in a TMR.
- You can have one or more RDBMS servers. They do not have to be managed nodes. If you want the RDBMS server to be a RIM host, then it can only perform that role for one TMR where it must be a managed node.
- Multiple RIM hosts can share one RDBMS.
- Beware of performance bottlenecks if you are considering the following:
  - Using a TMR server, or another application server as a RIM host in a busy environment.
  - Several applications sharing one RDBMS server. This could be through one or more RIM hosts.
  - Sharing the RDBMS across TMRs.
- Each RIM host, whether they are in the same TMR or not, must have direct IP connectivity to the RDBMS server.
- In Framework 3.6, the RIM host must also have direct IP connectivity to the managed node with the application making database requests. The RIM host can then open the IOM channel when returning the data for any database operations.
- Tivoli does not support a configuration that tries to share a RIM object across TMR connections.
- If you share one RDBMS server between two RIM based applications using the same database, their records will be automatically stored with a reference to their region. A query from either region will see the objects placed there by any TMR's application.

- You can have multiple RDBMS servers in a TMR being referenced by different RIM objects. In this case, you only need to re-create the RIM objects that are using the RDBMS server being updated.

Note

In a situation where TMRs are sharing the same database for Inventory data, ensure that you design your queries correctly if you do not want to see other TMR machines in the result of your queries. See Section 9.6, “Queries” on page 330.

Part 2. Tivoli Enterprise Products

Chapter 10. Software Distribution

Tivoli Software distribution is used by many customers for distributing, installing, and controlling software. There are many features that provide a great deal of function but whose complexity make this application one of the more challenging ones when it comes to resolving problems. In this chapter, we concentrate more on the standard Software Distribution file package. This is used when all that is required is to get a particular set of files from a source system into specific locations on a target system. If the installation is to a PC platform, and involves more complexity, such as the modification of initialization files, then the usual choice would be to use AutoPack. The AutoPack feature is the subject of Chapter 11, “AutoPack” on page 375.

Recent versions of Tivoli Software Distribution have included troubleshooting information in the *Tivoli Software Distribution Reference Manual*.

10.1 Differences with Software Distribution Version 3.6

There are some differences between Software Distribution 3.6 and previous releases. Besides the support for the Tivoli Management Agent (TMA) environment, these include:

- Better and more meaningful error messages.
- Individual logging of the nested file packages in the Software Distribution logs.
- Different format for Software Distribution logs.

These differences are highlighted throughout this chapter

10.2 Installation

For the new TMA environment, Tivoli Software Distribution has two main components: Tivoli Software Distribution 3.6 (server component) and the Tivoli Software Distribution 3.6 gateway package. In addition to these components, Tivoli Software Distribution 3.6 is shipped with an extensive API, the TEC Integration module, and Tivoli Software Distribution 3.6 RDBMS support.

For both the TMA and classical Tivoli environment, the Tivoli Software Distribution 3.6 Server has to be installed on the TMR server, on all designated source hosts, and on those managed nodes from which you would like to administer Tivoli Software Distribution.

The Tivoli Software Distribution Server Component is identified by a name such as:

```
TME 10 Software Distribution, Version 3.6
```

This appears in either the traditional Tivoli installation method or SIS. This component mainly consists of the file package editor, the AutoPack object editor, the GUI for launching the distribution operations (display object), the server engine, and the CLI commands.

The Tivoli Software Distribution gateway component is identified by:

```
Software Distribution Gateway, Version 3.6
```

Again, this is in either the traditional Tivoli installation method or SIS. This component must be installed on each managed node that has been designated as a TMA gateway and will serve the software distribution functions to the Tivoli Management Agents. The gateway component is the repository of all the distribution binaries that are necessary to repeat the data to the TMA endpoints. The gateway will also act as the repository of methods that are downloaded to the endpoints at distribution time.

Each of the Tivoli Software Distribution components can be installed using the Tivoli Desktop GUI, command line interface (CLI), or Software Installation Service (SIS).

You can refer to the redbook *New Features in Tivoli Software Distribution 3.6,* SG24-2045 for more information on 3.6 specifics.

10.3 Tivoli Software Distribution Internals

In this section, we will cover the overall process used by a Software Distribution and introduce the Tivoli methods that are involved.

A Software Distribution can involve the following types of machine:

- The TMR server
- The file package source host
- The TMA gateway
- The repeater
- Software Distribution targets: PC managed node, managed node, or TMA endpoint

A distribution will take place either from a managed node (including the TMR server) to another managed node, or from a managed node to a PC managed node or TMA endpoint. Between these two, there may be a number of intermediary steps including the use of repeaters and/or TMA gateways.

10.3.1 Tivoli Methods Used by Software Distribution

Every file package distribution is handled by up to five main methods:

- A push method - `fp_push`, `default_push`, or `fp_push_with_size`
- `fp_dist`
- `rpt`
- `fps_install`
- `fp_endpoint` (TMA)

The push method runs on the TMR server and is called the master method. This master method will coordinate the entire distribution process and spawn subsequent methods. The push method performs the following main steps:

1. Gets the file package and list of targets.
2. Runs the `fp_val_operation` validation policy.
3. Calls the `fp_dist` method on the source host.

Depending on the form you use to distribute a file package, the push method can be:

`fp_push` If the file package is distributed from the File -> Distribute option on the file package Properties or using the `wdistfp` command.

`default_push` If the file package is distributed from the Profile Manager -> Distribute on the profile manager window or by using drag and drop.

`fp_push_with_size`
If the size of the package was calculated, and you distribute it by using the command `wdistfp -u` to include the size in the package information. This is the push method that is spawned.

10.3.2 The Distribution Processes

When the `fp_dist` method is called on the source host with a list of targets and data, it performs the following sequence:

1. Calls the `obj_route` method on the repeater manager to determine how the file package will be routed to each target.
2. Bundles the contents of the file package into a data stream.
3. If the source host is a repeater, it distributes the data stream to all targets within its range, calling the `fps_install` method on each. If the target is a managed node, `fps_install` is called through the oserv. If the target is a PC managed node, the `fps_install` is initiated through the PC managed node's home host.
4. If a repeater, which has a target of the distribution in its range, is included in the source host's range, the source host calls the `rpt` method on the repeater and passes the data stream and list of targets to it.
5. If the source host is not a repeater, it calls the `rpt` method on the source host's repeater and passes the data stream and list of targets.

Note

If the `Always` flag is not set as an argument for the repeater, the source host will establish a point-to-point communication to the target. This is usually the case if there is only one target in the distribution list.

To see the repeater settings, enter the command: `wrpt`. In this example, `itso3` does not have the `Always` flag set, `rh2900a` and `rh2900c` have the `Always` flag set.

```
wrpt
itso3 [1]             wd-  []
rh2900a [1]           wda  []
rh2900c [4]           -da  []
```

Each repeater then:

1. Distributes the data stream to all targets in its range, calling the `fps_install` method on each (either directly through the oserv or the home host if the target is a PC managed node).
2. If a repeater, which has a target of the distribution in its range, is included in the repeater's range, it then calls the `rpt` method on the (sub)repeater(s) and passes the data stream and list of targets to it.
3. When the `fps_install` method finishes, it returns its completion status, such as whether the operation succeed or failed, to the `fp_dist` or `rpt` method (the method that called it). The `fp_dist` method does not complete until all the `fps_install` and `rpt` methods that it called have returned.

4. Correspondingly, it returns its results to the push method used. The push method completes after the `fp_dist` method completes. As the push method returns, it writes the information to any specified log file and Tivoli Notice Board.

The following figure gives an overview of the Software Distribution method:

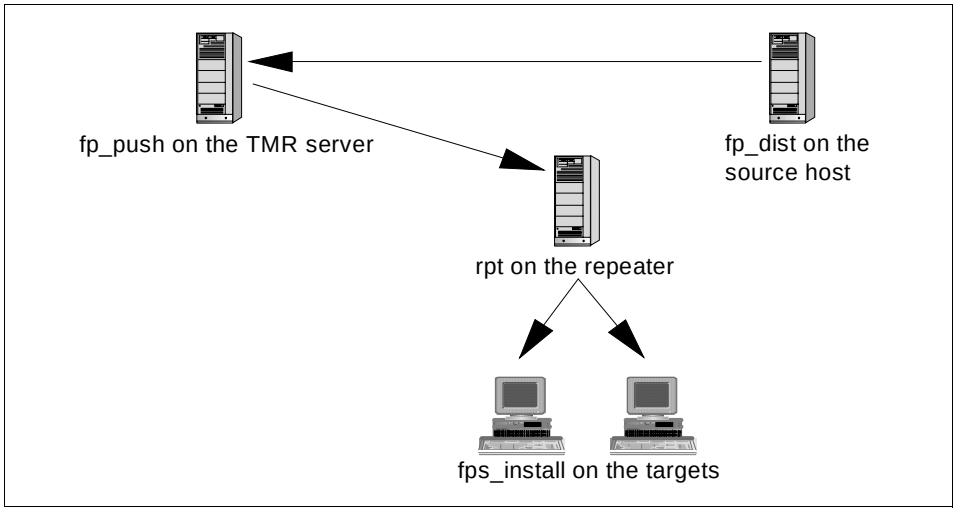


Figure 71. Software Distribution Method Overview

When you distribute a file package to a TMA endpoint, the method interaction is slightly different.

The `rpt` method now contacts the gateway cache for the existence of the `fps_install` method and checks the TMA endpoint for the existence of the appropriate executable - `fp_endpoint`:

1. Contacts the gateway cache for the existence of the `fps_install` method.
2. Checks the TMA endpoint for the `fp_endpoint` executable. If the `fp_endpoint` executable is not present in the cache on the endpoint, the gateway will initiate a downcall and load the `fp_endpoint` executable into the cache.

The redbook *New Features in Tivoli Software Distribution 3.6,* SG24-2045 has more information on Software Distribution 3.6 internals.

10.4 Repeaters and Networks

As we have seen, Software Distribution can rely on repeaters. It will use the Mdist and BDT/IOM Framework Services. Refer to section 7.6, “Multiplexed Distribution and Bulk Data Transfer” on page 259 for a more detailed description of these services.

The MDist feature can be configured by using the `wrpt` command. Several options to be used when setting up Software Distribution include:

`wrpt -L` - Displays active distribution as follows:

```
# wrpt -L
  1 fp_distribute Nov 10 17:10:14       112/0 [0-0]
#
```

The first item is the Software Distribution identifier (ID), the second is the name of the process followed by the start date and time and the last numbers are the statistics in the format: `in/est_size [out_min-out_max]`.

`wrpt -q src targets` - Displays the distribution route.

The example in Figure 146 shows the route a software distribution would take from the source host `itso3` to the final TMA endpoint `pctmp112`.

The following

```
[root@itso3]> wrpt -q itso3 pctmp112
--[RPT:itso3 [1]]
   |--[RPT:rh2900a [1]]
   |   |--[RPT:rh2900a [1]]
   |   |   |--pctmp112 [29]
   |   |
    |
```

Figure 146. Sample of Distribution Route

See also 7.6.2, “Repeaters” on page 260 for more information on repeater configuration.

`wrpt -t` - Tunes the repeater.

There are several tuning parameter associated with Software Distribution, and the recommendations in this chapter should be treated as a general rule of thumb. In general, the default repeater settings are not typically adequate for most environments and will need some tuning.

`mem_max` The amount of memory where the data is initially spooled to on the repeater. Since a hard drive is not slower than a LAN or WAN link, this value can be small and the data can be spooled to disk. A good size is 10 MB (mem_max=10000).

`disk_max` The amount of disk space to be used for spooling data once `mem_max` is full. `disk_max` needs to be at least 10 to 20 percent larger than your largest distribution.

`disk_hiwat` This parameter is the high water mark for the disk. When the repeater reaches this value, it tells the source host to slow down sending data. This should be 10 to 20 percent less than your largest distribution.

`disk_dir` This is the directory/file system that is used to spool data. This file system free space **MUST** be larger than `disk_max`.

Note

Tivoli Software Distribution 3.6 corrects some errors that were previously related to using the `disk_dir` parameter. Previously, on UNIX systems the filesystem used for `disk_dir` needed to have other write permissions. Tivoli now writes the temporary files as root. Also, if you incorrectly specify the `disk_dir` directory (maybe it does not exist), Software Distribution will use the directory/file system that is returned from the `wtemp` command to use as the `disk_dir` directory/filesystem. This could cause unexpected errors in your environment if there is insufficient space or authorization for the temporary file system.

`net_load` KB per second used in a distribution. This is the amount of network bandwidth that will be used. This should be tuned for individual WAN/LAN links. Because this parameter is present, you should try to have a repeater at a central site for each WAN type. This will enable you to tune the repeater for each link (T1, 56k, and so on).

`max_conn` This is the number of machines a repeater will distribute to simultaneously. Each connection will use approximately 1.5 MB of memory when counting requirements for spawning new processes. Keep this parameter low, around 25, unless you have plenty of memory.

`stat_intv` This specifies a timeout value for connections between a repeater and its managed nodes or PC managed nodes. The duration of the connection includes the time it takes to distribute each data packet to the target system. This

parameter, working in conjunction with the net_load, is, again, why you would want to have a repeater for each WAN type.

Each of these parameters are global. If you have more than one distribution going through the repeater at once, each value is used independently. For example, if max_conn is set to 10, and you have two simultaneous distribution using that repeater, there will be 20 connection, not 10 and 30 MB of memory will be used, nor 15 MB.

The example in Figure 147 is setting the max_conn parameter from the default value of 100 to a value of 25.

```
[root@itso3]> wrpt -t rh2900c
mem_max      = 50
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "d:\tmp"
net_load     = 500
max_conn     = 100
stat_intv    = 2

[root@itso3]> wrpt -t rh2900c max_conn=25

[root@itso3]> wrpt -t rh2900c
mem_max      = 50
disk_max     = 50000
disk_hiwat   = 50000
disk_time    = 1
disk_dir     = "d:\tmp"
net_load     = 500
max_conn     = 25
stat_intv    = 2
```

Figure 147. Example of Setting a Repeater Parameter

An additional parameter that is available on the repeater is the Repeater Manager timeout parameter. This parameter controls the amount of time each repeater node will wait for distribution status after sending the entire data stream to a target endpoint. If you use this parameter, you will need to test the timing on the slowest machine available. This is also a TMR server level parameter - that is, it is set at the TMR level and will affect all repeaters in that TMR. This parameter cannot be changed during a distribution. By default, this value is set to 0, meaning wait indefinitely. To view the existing value, use `wrpt -T`. To set this value, use `wrpt -T [seconds]`.

Note

Remember that you can also tune most repeater parameters during the active distribution:

`wrpt -k id -t repeater_name parameter=new_value`

The last option for `wrpt` of particular interest is `wrpt -A -k id -`.

This aborts an active distribution. You must enter the ID of the distribution that you obtained with `wrpt -I`.

10.4.1 Initiating BDT/IOM

See Section 7.6.3, “Bulk Data Transfer and Inter-Object Messaging” on page 271 for a description of BDT and IOM.

When the target is a managed node, the source machine determines the size of the file package. If is larger than 16 KB (configurable from Framework 3.2 onward), the source machine allocates the next available port number and opens it to listen for a file package request from the target machine. The source machine then passes a *dkey* to the target. The dkey comprises the IP address of the source host or repeater from which the file package will be distributed and the allocated port number. The managed node connects to the IOM channel specified by the IP address and port number from the dkey. Once the source host or repeater detects that the node is connected to the IOM channel, it sends the file package data stream to it. The managed node reads and processes the data stream.

When the target is a PC managed node, the same determination and port selection is made. The source machine then passes a dkey to the managed node with which the PC managed node is associated (PC managed node’s home host). The managed node then passes the dkey to its PC managed node over port 6543. Specifically, the dkey is passed to the PC agent on the PC managed node. The agent on the PC managed node connects to the IOM channel specified by the IP address and port number from the dkey. Once the source host or repeater detects that the PC agent is connected to the IOM channel, it sends the file package data stream to the agent. The agent reads and processes the data stream.

When the target is a TMA endpoint, again the same determination and port selection is made. The source host passes the dkey to the gateway that is associated with the target endpoint. The gateway will then connect to the source or repeater IOM channel specified by the IP address and port number

from the dkey. Simultaneously, the gateway contacts the endpoint to initiate the `fps_install` method.

When the source host or repeater finishes sending the file package data stream, it closes its IOM connection. Similarly, when the managed node, PC agent, or TMA endpoint finishes processing the file package data stream, it closes its connection and returns completion status.

For more information see the *Tivoli Software Distribution Reference Manual.*

10.5 Setting Timeout Values for a Distribution

Tivoli Software Distribution provides server and client level time-out parameters. These parameters enable you to set a time interval, that once reached, either the server or client will interrupt a distribution.

Setting these time-outs will help you to avoid situations caused by hung distributions. Examples of these are:

- BARC scripts that are looping or hanging. These include any scripts that could require the user to take action, such as closing a pop-up status window.
- Down nodes, including systems that are in the process of rebooting, are off-line or completely disconnected from the Network.
- Communications problems, such as breaks in the communications channel during distribution.

You need to set each of the distribution time-out parameters before distributing a Software Distribution Profile. The parameters are:

- Configuration script time-out
- Repeater Manager time-out
- High level TCP time-out
- Gateway session time-out

These parameters are described in the remainder of section 10.5. For further information, refer to the *Tivoli Software Distribution User's Guide*.

10.5.1 Configuration Script Timeout

Configuration scripts are the Before, After, Remove, and Configuration (BARC) scripts. Set the `progs_timeout` keyword in the file package definition to specify a timeout value for each BARC script running on the client. This value

should be customized for each file package. You should set the `progs_timeout` by running each BARC script in a 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 twice that time. You will only be able to change this setting by using the `wsetfpopts` command or by exporting the file package, changing the value, and re-importing it.

For example, you have a file package that has a before script that checks disk space, checks for the presence of several DLLs at the correct level, and checks for a pre-requisite software package to have been installed. You have an after script that moves files/DLLs from a staging area (and any files/DLLs that it may be replacing, those files/DLLs are moved to a replacement hold area), adds registry entries and icons to the desktop. You have a remove script that moves files/DLLs from a temporary replacement hold area, deletes any files/DLLs that were new, deletes the registry entries, and removes the icons from the desktop. On a 486 processor machine, you run each of these scripts individually and receive the following times:

Before script: 10 seconds

After script: 135 seconds

Removal script: 100 seconds

The value used for the `progs_timeout` should be 135 * 2 or 270 seconds. So, the entry in the exported file package should read:

```
progs_timeout=270
```

10.5.2 Repeater Manager Timeout

This timeout is a server-level timeout for all repeaters in a TMR. This value is stored on the repeater manager. This parameter specifies how long each repeater node will wait for distribution status after sending the entire data stream to a target managed node or PC managed node. If the target does not respond in the allotted time, the repeater logs a distribution error and terminates the connection. This timer only starts after the file package is distributed to the target system. Therefore, packet-transfer time and before script times are not included in this timeout value.

10.5.3 High-Level TCP Timeout

This is the `stat_intv` parameter available on the repeater configuration. This value sets a time after which a blocked connection between the repeater and the client system is terminated. If the repeater stops transmitting data to the client for an interval that exceeds the `stat_intv` setting, the session is

terminated. This value should be set to a value greater than the time it will take to run the longest configuration script on the target.

10.5.4 Gateway Session Timeout

This specifies a timeout value for connections between a gateway and its endpoints. This parameter provides the same functionality for the TMA endpoint as the `stat_intv` parameter does for the managed node and PC managed node. You configure this parameter by using the `wgateway` command.

10.6 File Package Definition

When you create a file package, you can customize some of its properties from the GUI. Other properties are only available by exporting the file package or by using the `wsetfpopts` command. You can export the file package definition using the GUI or by CLI:

```
wexprtfp [[-a] [-c] -f [managed_node:]filename] fp_name
```

The exported package can then be modified and imported again. There are a lot of properties, but here is a short list of some of the most important:

```
#*TFP-v2.05 Tivoli Filepack (version v2.05)
backup_fmt=
list_path=
file_cksums=n
append_log=n
src_before_prog_path=
src_before_input_path=
src_before_as_uid=
src_after_prog_path=
src_after_input_path=
src_after_as_uid=
prog_env=
log_file_uid=
log_file_gid=
log_file_mode=
progs_timeout=0
```

Table 13 provides a description of these properties:

Table 13. File Package Properties

Keyword	Description
backup_fmt	Specifies a format used to make a backup of files that exist at the target.

Keyword	Description
list_path	Specifies the directory on the target in which to write the log file containing a list of all files and directories distributed to or removed from the target. The log file is fpname.log.
file_cksums	Indicates whether to use checksums on the individual files in the file package to detect differences between the source file and the target file rather than just modified time, ownership, group membership, and file mode. This is only relevant when any_changes are indicated for distribute.
append_log	Specifies whether to append a notice to the log file when file package distribution, commit, or removal operations are performed.
src_before_prog_path	Specifies programs to be run on the source host after the file package is distributed to the targets.
src_before_input_path	Specifies files to be passed through standard input to the programs specified by the src_before_prog_path.
src_before_as_uid	Specifies the UID of the user under which to run src_before_prog_path.
src_after_prog_path	Specifies programs to be run on the source host after the file package is distributed to the targets.
src_after_input_path	Specifies files to be passed through standard input to the programs specified by the src_after_prog_path.
src_after_as_uid	Specifies the UID of the user under which to run src_after_prog_path.
prog_env	Specifies a string which will be the subject of a putenv before any configuration program is run in a UNIX machine. The string is a list of name=value.
log_file_uid	Specifies the UID of the log file specified by the log_file keyword.
log_file_gid	Specifies the GID of the log file specified by the log_file keyword.
log_file_mode	Specifies the file mode of the log file specified by the log_file keyword.
unix_on_error_prog_path	Specifies a program to be run on a UNIX target if an error stops the distribution of a file package.

Keyword	Description
progs_timeout	Sets the client-level timeout value for all configuration programs (BARC) specified in the file package definition. This timeout value will apply to each configuration script and is reset as each script begins on an individual target.

Further details of these parameters can be found in the *Tivoli Software Distribution Reference Manual*.

File Package Tip

You can create a master file package with all the properties you wish to configure. Every time you then create a new package, select the files you wish to distribute, save the changes, and clone the package with a new name. By doing this, you will not have to export the definition, set the properties, and import it again.

10.6.1 File Package Policies

Refer to the *Tivoli Software Distribution Reference Manual* for details of the numerous default and validation policies. Default and validation policies provide enforced guidelines for file package and AutoPack properties and operations. AutoPack policies start with ap_, and file package policies begin fp_.

10.7 Troubleshooting Software Distribution

Problems with Software Distribution are usually caused by:

- Configuration errors
- Network problems
- Target problems

Software Distribution provides a number of sources of information about errors:

Log file A detailed list of the success or failure of the distribution for each target. Generally, you should enable this option for every file package. The log file has more information than the associated Software Distribution notice group posting or e-mail options.

Distribution preview	This dialog can be used but is not always reliable.
Definition file	The file package definition file is a text file describing the settings of all of the file package options obtained by exporting a file package.
`wgetfpattr`	Gets the attributes for a file package.
`wrpt`	Gets/sets repeater information.

10.7.1 Troubleshooting Checklist

If the results are not as expected, the following checklist should help identify possible causes:

1. The first step in troubleshooting is to examine the log file associated with the distribution. Generally, the **Send to log file on** option should be enabled when performing a distribution because it contains more information than either the e-mail option or the post a notice option.
2. The file package definition associated with the file package should also be examined. Export the file package from the GUI or use the `wexprtfp` command. Options to check include:
   - `preproc/postproc`

     Check the filters defined here. The data runs through these just before it leaves the source and just after arriving at the target.
   - `stop_on_error`

     Specifies whether to fail a distribution to a target when any error is encountered.
   - `backup_fmt`

     Specifies both whether and where backups are for overwritten files.
   - `list_path`

     Where the list of files distributed was written.
   - `prog_env`

     The environment used by the configuration programs on the target.
   - `log_file`

     The name of the log file written.
   - `log_host`

     The host location of the log file.

3. Check the file list section of the file package definition. Ensure that shell commands are properly written and that modifiers for directories and file names are relevant. Also check the exclude files section to make sure that the files excluded are relevant.
4. Check to make sure that you can execute the configuration scripts (BARC) on a target machine without failures. You may want to manually transfer files to a machine and manually run the scripts and check for failures.
5. Disable the configuration scripts and re-enable them one at a time until a failure occurs. Check them for the following:

   File locations Are the programs where Software Distribution expects to find them? Remember, the configuration programs must be on either the source host or the target. Are the proper input files present?

   Program validity Are the programs properly written? Will they run on the target operating system(s)? If the target is a UNIX system, is the UID option set properly?

   Source host Check the before and after programs.
6. Check the methods. Look for `fps_install` in an `odstat` output (see Chapter 6, “Commands and Logs for Troubleshooting” on page 131). There should be one for each target if the target is a PC managed node. For TMA endpoints, you will need to place the gateway(s) into debug level 6 to see the upcall, downcall, and repeater information.
7. Monitor the network and watch for packet movement. The `net_load` could be set so low that it appears to be hung.
8. Create a test file package scenario using the GUI, the CLI and the original file package definition. If the file package is large (over 16 KB), create a file package smaller than 16 KB. Distribute and check the results of each of these. Do you get the same problems (and vice versa)?
9. Interregion failures usually occur because the source or destination cannot ping the other TMR server. Create the file package in the same TMR as the targets receiving them. Can the file packets be successfully distributed now?
10. Make the managed node that is the host for a PC managed node target a repeater.
11. Make sure all gateways are repeaters.

You can experiment with different sized file packages and the use of scripts to try to identify under which circumstances problems occur:

1. Send a file package less than 10 KB without any scripts.
2. Send a file package around 500 KB without any scripts.
3. Send another 10 KB package with scripts.
4. Send another 500 KB package with scripts.

10.7.1.1 Lost-n-found

Lost-n-found can also be a useful source of information. Since Version 2.02, Tivoli Software Distribution has registered callbacks for `wchkdb` to check for these file packages. The three file-related issues that are checked are:

- Does a file package have a source host that no longer exists?
- Does a file package have a log host that no longer exists?
- Does a file package have a nested file package that has been deleted (or whose parent has been deleted)?

If any of these cases apply, `wchkdb` moves the file package to `/lost-n-found`. Run the command `wls /lost-n-found` to check for these file packages. The `wmvfpobj` command can be used to move a file package back to a profile manager.

10.7.2 PC Managed Node Troubleshooting Specifics

To be a target for a distribution, a PC managed node must be running on the Network with the Tivoli PC agent. You should check that the PC managed node is valid. Some platform types can fail because there are not enough resources allocated. For example, there may not be enough sockets or enough memory for buffering (the PC agent requires at least 10 sockets). This is a problem specific to the transport stack in use.

When performing distributions to PC managed nodes, remember:

- Every line of a DOS/Windows configuration program script must have a new-line character at the end. UNIX systems do not automatically add a new-line character to end of each line of a file. You can add the character in `vi` by typing control-V (`^V`) and then control-M (`^M`).
- `_default.pif` must be set to the background.
- Tivoli log files are written to the `/tmp` (environment variable) directory.
- The Windows NT service agent writes to the event log. The Windows NT console agent writes to its own DOS window.

When working with the `tmeagent.cfg` file, refer to the *Tivoli Framework Planning and Installation Guide*.

10.8 Software Distribution and Other Log Files

There are many log files to review if problems occur with Software Distribution, some provided by Software Distribution and some provided by other components that are utilized by Software Distribution. These files are the subject of this section.

10.8.1 Software Distribution Log

From 3.6, the format for the software distribution logs has changed. These format changes now include the size of the distribution and information concerning nested file packages. In Figure 159 on page 370, we distributed a nested file package. You can now see the success/failure of each nested file package within a file package. The nest level information will be contained in the Software Distribution log even if no nested file packages are present. The nest will be 0 in this instance:

```
File Package:  "test1reg"
Operation:     install  (m=5)
Finished:      Thu Dec 10 16:02:16 1998
--------
Source messages:
<none>
--------
pctmp109:SUCCESS
size=-1, nest_level=0
size=-1, nest_level=1
--------
pctmp112:SUCCESS
size=-1, nest_level=0
size=-1, nest_level=1
================
```

Figure 148. Sample of New Software Distribution Log Format

`Size` will always be `-1` (to Tivoli Software Distribution - this is analogous to I don't know) UNLESS you do the following: On the file package you click the right mouse button and instruct Tivoli to calculate the size of the file package. After this is complete, you must use the command line to distribute the file package. The command to use to do this is `wdistfp` with the `-u` option. However, this will cause a Software Distribution progress bar to be displayed on the endpoint (TMA or PC Agent).

The designated Software Distribution log file has a brief description of any problem in the distribution. The error messages listed in Table 14 on page 357 and their causes are not meant to be all-inclusive, but to capture some of

the errors, we were able to simulate. Some of the error messages you might encounter are:

Table 14. Software Distribution Log File Error Messages

Error message from error log	Reasons for error and possible corrective actions
Cannot create tmp file (`C:\Tivoli\db\rh2900a.db\tmp\pmap24`). errno=Permission denied	You do not have the correct permissions for the directory in a repeater setting. To correct, use the `wrpt -q` command to trace the route from the source host to the endpoint. Locate the suspect repeater. Use the `wrpt -t repeater_name` command to find the setting for the `disk_dir` parameter. Check the permissions on that folder or file or file system. You must have world write permission or you will get this error message. Note: You should only see this error message if you are using NT repeaters. If you are using UNIX repeaters, the writes to the `disk_dir` directory now occur as root (before 3.6 user ID was nobody). This error can occur with either the TMA endpoint or PC managed node agent.
`recv_session`: timeout (300 seconds) waiting to receive from 146.84.32.212+9494	There was no communication from the gateway to the TMA endpoint for 300 seconds. If the BARC scripts are taking longer than this time, or if there are network issues, increase the value of this parameter by using: `wgateway gatewayname set_session_timeout numberofsecs.` This error message also occurs if the `lcfd` terminates during a distribution, or if the endpoint process is not running prior to the start of the distribution.
`decrypt_data`: HMAC does not match encrypted data	As each TMA endpoint is created, a special encryption key is also created and known to the endpoint and the TMR. If you attempt to distribute anything to an endpoint, and these keys do not match, you will receive this error. To correct, use the http interface and check the gateway that the endpoint is bound to. This interface will also show the correct Tivoli endpoint name. See also 6.3.6, “HMAC Encrypted Data Error” on page 161.
Unable to resolve method `fps_install` starting from object 1351550138.1.508	The gateway component of Software Distribution was installed on the gateway; however, the server component was not installed on the object 1351550138.1.508. To find the TMR, use `wlookup -ar ManagedNode | grep ‘1351550138.1’`. The appropriate managed node should be returned. Use the Tivoli command `wlsinst -ha` to verify Software Distribution is not installed on the server. Install Software Distribution and resubmit distribution. This error will occur with the TMA endpoint.
`ipc_create_remote` failed: unable to connect to 146.84.32.208+9494: (67) IPC shutdown	The target machine was turned off for either the TMA endpoint or PC managed node Agent. Use `ping` to check to see if the endpoint is available. Turn the endpoint back on.

Error message from error log	Reasons for error and possible corrective actions
iom_timed_send failed with code 67: communication failure	TMA endpoint was powered down in the middle of a distribution. Communication was lost. Turn the endpoint back on and redistribute. Also saw this error when there was not enough disk space on the target machine for a TMA endpoint.
Input thread abort	This error occurs when the TMR or repeater runs out of virtual memory. This error will occur with either the TMA endpoint or PC managed node agent.
Fri Dec 11 08:59:31 CST 1998 (17): system problem: Write Failed. File (/tmp/pmapQdHjMr). errno=No space left on device 1998892590.1.1320#TMF_Gateway:: Gateway#'	There was not enough physical disk space available on the repeater. Clean up the file system or drive, or use the wrpt -t repeater disk_dir=newplace to use a different file system/disk. This error will occur with either the TMA endpoint or PC managed node agent.
Write: c:/temp/smit.script: No space left on device	This error occurred when there was not enough disk space available on the PC managed node. This is a different error message than was received for disk space on a TMA endpoint.
An internal error occurred: destination dispatcher unavailable	An oserv process was unavailable along the route of the endpoint, TMA, or PC managed node. To see the route, use the wrpt -q command. Also, correlate all endpoint errors of this type and try to narrow the search for which TMR or repeater could be causing the problem. Also, verify the availability of all TMRs/repeaters with the wping command. This error will occur with either the TMA endpoint or PC managed node agent.
Could not create process: No such file or directory	A configuration script was not found. This script was either not in the specified place on the source host or on the endpoint. If you are using the PC managed node w commands (wdskspc or wseterr) on a TMA endpoint, you will need to use the wdepset commands to allow these commands to be downcalled to the TMA endpoint. See 10.9, “Using the PC Agent w Commands on a TMA Endpoint” on page 370 for instructions on how to do this.
Process did not complete in the specified timeout	The configuration script did not complete in the amount of time specified in the progs_timeout parameter in the file package for a TMA endpoint. If this is happening on a majority of endpoints, you will need to increase the progs_timeout amount.
c:/keeptkng.bat script timed out.	The configuration script did not complete in the amount of time specified in the progs_timeout parameter in the file package for a PC managed node agent. If this is happening on a majority of endpoints, you will need to increase the progs_timeout amount.

Error message from error log	Reasons for error and possible corrective actions
Script stderr: [The name specified is not recognized as an internal or external command, operable program or batch file.]	On the TMA endpoint, the end user terminated the configuration program.
`iom_timed_send` failed with code '38': ' OR code 67.	The PC managed node agent was down prior to the start of the distribution. This PC managed node is using a Windows NT repeater. Restart the agent. If this error is pervasive, contact Tivoli support.
Attempt to connect to 'w98' failed with errno 79.	The PC managed node agent was down prior to the start of the distribution. This PC managed node is using a UNIX repeater. Restart the agent. If this error is pervasive, contact Tivoli support
Attempt to connect to 'w98' failed with errno 78	The PC managed node was down prior to the start of the distribution. This PC managed node is using a UNIX repeater. Turn the PC on.
Script stderr: [The system cannot execute the specified program.]	The ID does not have permission to execute the program specified in the BARC script or the script itself could not be found. Check permissions on the directory/folder and on the program itself. This error message occurs for TMA endpoints.
'c:/keeptkng.bat' couldn't start script	The before script did not have the appropriate permissions or the script could not be found. If you are running the script from the target, make sure the script exists. If the script exits, check the permissions on the folder/directory and script. This error message occurs for PC managed node Agents.
An internal error occurred: IPC shutdown	This error was received when a gateway was restarted in the middle of a distribution. There could be other causes for this error.
Timeout (300 seconds) waiting to receive from 146.84.32.208+9494	The TMA endpoint did not respond to the gateway. In actuality, the agent was shutdown in the middle of a distribution. This error message can also be received if there is heavy Network traffic between the gateway and the TMA endpoint.
Requested resource not found	A specified dependency set was removed from a method. This error occurs on a TMA endpoint.
Resource 'c:/Tivoli/bin/lcf_bundle//bin/w32-ix86/agentcli/wdskspc.exe' not found	If you specified a dependency set, check the dependency set location on the gateway. The downcall is unable to locate the dependency set. This error occurs on a TMA endpoint.

10.8.2 Tivoli PC Agent Tracing and Other Log Files

When you are using the PC Agent with Software Distribution 3.6 or later, you can use the following to trace the methods and processes you will see running on the various TMRs and repeaters participating in the distribution.

10.8.2.1 odstat Trace for a PC Agent Distribution

This example demonstrates a distribution from a TMR (also acting as the source host) to a PC managed node through a repeater (also acting as the PC Home Host). In Figure 149 on page 361, there is an abbreviated sample of the `odstat` command output.

Refer to 6.1, “The odstat Command” on page 132 for a description of the `odstat` output. The pieces we are especially interested in are the thread ID (the first number, the parent thread ID, prefixed with a dispatcher number as in 1-14819 and the method name, last on the line). The lines in Figure 149 on page 361 picked out with bold numbers are explained in the text following the figure.

```
odstat -v

1 14819 O+hdoq1-14311  done    18     0 09:31:34        1295714281.1.1099#FilePac
kage::FpoCore# fp_push
        service: "202 attr 'push_time' dat:..6u0.:"  err=0  pid=78134
 14820 O+   1-14819  done    47     0 09:31:34        0.0.0 get_host_location
        service: ""  err=0  pid=0
 14821 O+   1-14819  done    57     0 09:31:34        1295714281.1.1095#TMF_CCMS
::ProfileManager# get_policy_region
        service: ""  err=0  pid=0
 14822 O+   1-14819  done    15     0 09:31:34        0.0.0 get_name_registry
        service: ""  err=0  pid=0
 14823 O+hdoq1-14819  done   109     0 09:31:34        1295714281.1.26 lookup
        service: ""  err=0  pid=31012
 14824 O+hdq1-14819  done    59     0 09:31:35        1295714281.1.14#TMF_SysAdm
in::Library# lookup_object
        service: "201 attr 'members' dat:"  err=0  pid=76416
14825 O+hdq1-14819  done     9     0 09:31:35        1295714281.1.1069#TMF_Poli
cyRegion::GUI# is_validation_enabled
        service: "201 attr 'classes' dat:"  err=0  pid=11574
*14826 O+hdoq1-14819  done   613     0 09:31:35  e=12  1295714281.1.26 lookup
        service: ""  err=0  pid=31012
 14827 O+hdoq1-14819  done   105     0 09:31:35        1295714281.1.26 local_loo
kup
        service: ""  err=0  pid=31012
*14828 O+hdoq1-14819  done   324     0 09:31:35  e=12  1295714281.1.1006#Tec::Se
rver# connect_agent
        service: ""  err=0  pid=18582
2 14829 O+ho 1-14819  done    24     0 09:31:35        1295714281.1.348#TMF_Manag
edNode::Managed_Node# fp_dist
.
3 14832 O+hdoq1-14829  done    12     0 09:31:35        1295714281.1.366 _get_fin
al_timeout
        service: "201 attr 'final_timeout' dat:"  err=0  pid=9648
 14833 O+hdoq1-14829  done    95     0 09:31:36        1295714281.1.26 lookup
        service: ""  err=0  pid=31012
4 14834 O+hdoq1-14829  done   292     0 09:31:36        1295714281.1.366 obj_rout
e
.
14843 O+hdoq1-14829  done  7580     0 09:33:08        1295714281.1.1100#FilePac
kage::FpoCore# get_fp_data
        service: "201 attr 'description' dat:"  err=0  pid=79154
5 14844 O a  1-14829  done   162     0 09:33:10        1998892590.1.1320 rpt
14845 O+aho1-14829  done    30     0 09:33:17        1295714281.1.348#TMF_Manag
edNode::Managed_Node# fp_operation
        service: ""  err=0  pid=15044
.
6 14851 O+hdoq1-14829  done     6     0 09:37:59        1295714281.1.348#TMF_Mana
gedNode::Managed_Node# privileged_write_to_file
        service: ""  err=0  pid=15048
 14852 M    2-12177  done     0     0 09:37:59        1295714281.1.348#TMF_Manag
edNode::Managed_Node# echo
service: ""  err=0  pid=0
7 14853 O+hdoq1-14829  done     6     0 09:38:01        1295714281.1.348#TMF_Mana
gedNode::Managed_Node# privileged_set_file
        service: ""  err=0  pid=15048
```

Figure 149. Sample odstat - TMR Server and Source Host for a PC Managed Node

```
odstat -o 1998892590.1.7-cv

8 13174 M hdoqs 1-14844   run     0     0 09:34:57        1998892590.1.1320 rpt
        service: ""  err=0  pid=29030
 13178 O+ahdoqs1-13174   run     0     0 09:34:59        1998892590.1.1320 rpt
service: ""  err=0  pid=29030
9 13182 O+ahdoq 1-13178   run     0     0 09:34:59        1998892590.1.1230#TMF_Pc
ManagedNode::Pc_Managed_Node# fps_install
        service: ""  err=0  pid=47932

ps -ef | grep skel         (run this on the repeater)
root 15214 49512   1 09:35:14  pts/0  0:00 grep skel
    root 47932 28164   0 09:34:59      -  0:00 pc_mannode_skel1
```

Figure 150. Sample odstat and ps - Repeater for PC Agent

The PC Agent distribution outlined in the above figures show the `odstat` output from the TMR/Source Host to the Repeater/PC Home Host. The following describes the `odstat` information in Figure 149 and Figure 150**:**

1. Line 1 shows the `fp_push` method on the TMR that is initiating the Software Distribution. This method is the parent of all the other distributions.The thread ID is `14819`.
2. Line 2 shows the next major method, `fp_dist`. The thread ID is `14829` and its parent thread ID is `14819`:

```
14829 O+ho 1-14819   done     24      0 09:31:35
1295714281.1.348#TMF_ManagedNode::Managed_Node# fp_dist
```

3. The third method we are interested in seeing is the `_get_final_timeout`.
4. The fourth method is the `obj_route` method. This is the child thread of the `fp_dist` method thread.
5. The fifth method is the repeater method. You will have a repeater method on the source host *for each* repeater participating in a distribution. This method will become the parent for the repeater method on the managed node whose OID begins with `1998892590.1`. The thread ID of `14844` will be seen in the `odstat` of `1998892590.1.7` (use `odadmin odlist` to find the OID of the managed node - or `wlookup -ar ManagedNode | grep '1998892590.1'`) as a parent thread of `1-14844`.
6. At this point, you will want to do a `wlookup -ar ManagedNode | grep '1998892590.1'` to discover the repeater is `itso2`. To look at the `odstat` on the other managed node, use the `odstat` command with the `-o OID` option. From the output in Figure 150 on page 362, on line 8 you can see the rpt method for this distribution. The thread ID is `13174` and the parent thread ID is `1-14844`. This is the thread ID from line 5.

7. The `rpt` method on the repeater will then invoke another repeater method, which in turn, will invoke an `fps_install` method *for each endpoint* the repeater will distribute to. You will also see a `pc_mannode_skel1` process on the repeater for each `fps_install` method running. The number of each of these methods/processes that will run simultaneously is dependent on the `max_conn` setting in the repeater.

Note

You can determine the endpoint of a distribution from the OID on the `fps_install` method. To do so use:

```
idlattr -t -g 1998892590.1.1230 label string
```

An example of a response might be:

```
"rh2900c"
```

8. After all `fps_install` methods are complete from all repeaters participating in the distribution, control is returned to the calling parent, the `fp_dist` method. After that, this method calls the methods to write to the log files, as in lines 6 and 7.
9. Finally, the distribution is complete, and any notices are written to the notice groups, and any pop-ups occur on the appropriate desktop.

10.8.2.2 Additional Sources of Information for the PC Agent

On NT endpoints, you should see a connection event, receive event, and disconnect event in the application event log with the source of TME Agent. These events will remain in the event view log until deleted. Examples are shown in Figure 151 on page 364.

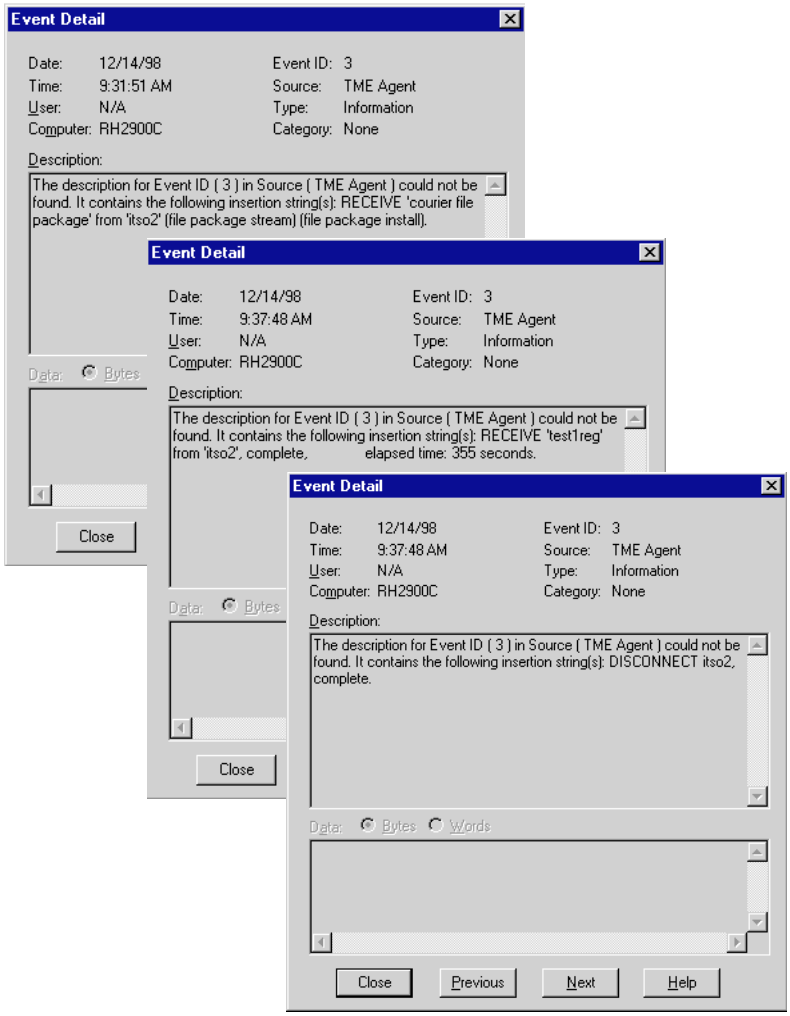


Figure 151. Event Viewer Events from the Software Distribution

On Windows 95 endpoints, there is a tivoli.log file that will include similar information. This file is located in the `%SYSTEMROOT%` directory. You can use the `wgetfile` command to retrieve these logs. When the Windows 95 Agent is restarted, this file is recreated; so, you need to retrieve this file as soon as possible after a distribution error.

Note

Three commands that can be useful on endpoints with PC Agents are:

`wdir HostName DirPath` - remotely lists a directory.

`wgetfile HostName RemoteFile LocalFile` - retrieves a file from `HostName`.

`wputfile HostName RemoteFile LocalFile` - puts a file on `HostName`.

Figure 152 is an example of the tivoli.log file:

```
# wgetfile machinename c:/windows/temp/tivoli.log /tmp/w98tivoli.log

12/15/98 11:51:03 : TME Windows 95 Agent, Version 5.000
12/15/98 11:51:03 : Release: 5.000 Build 01
12/15/98 11:51:03 : initializing
12/15/98 11:51:03 : REQUEST-WINSOCK V2.0:
12/15/98 11:51:03 : WINSOCK V1.1:
12/15/98 11:51:03 : temp: 'C:\WINDOWS\TEMP'
12/15/98 11:51:03 : msgcat: 'c:\etc'
12/15/98 11:51:03 : Listening on Port: 6543
12/15/98 11:51:03 : Hostname: rh2900m.dev.tivoli.com
12/15/98 11:51:03 : IP Address: 146.84.32.180
12/15/98 11:51:03 : statistics:
12/15/98 11:51:03 : Request Packet Size:     820
12/15/98 11:51:03 : Data Packet Header Size: 20
12/15/98 11:51:03 : Service Port:            6543
12/15/98 11:51:04 : Setting CLIENTNAME to:
12/15/98 11:51:04 : w98pc
12/15/98 11:51:26 : connection from itso3
12/15/98 11:51:26 : PEERINFO
12/15/98 11:51:26 : hostname: rh2900m
12/15/98 11:51:26 : hosttype: 0x1000
12/15/98 11:52:06 : *** Windows 95 Properties ***
12/15/98 11:52:06 : Version Major: 3
12/15/98 11:52:06 : Version Minor: 95
12/15/98 11:52:06 : Coprocessor Installed: TRUE
12/15/98 11:52:06 : Serial Ports         : 1
12/15/98 11:52:06 : Floppy Drives        : 1
12/15/98 11:52:06 : Physical Memory (Total): 74997760
12/15/98 11:52:06 :                 (Avail): 47869952
12/15/98 11:52:06 : Virtual  Memory (Total): 74997760
12/15/98 11:52:06 :                 (Avail): 47869952
12/15/98 11:52:06 : PEERINFO , complete
12/15/98 11:52:06 : DISCONNECT itso3, complete
12/15/98 11:54:34 : connection from itso3
12/15/98 11:54:34 : RECEIVE 'courier file package' from 'itso3' (file package stream)
(file package install)
12/15/98 11:55:46 : RECEIVE 'test1reg' from 'itso3', complete
12/15/98 11:55:46 : elapsed time: 72 seconds
12/15/98 11:55:46 : DISCONNECT itso3, complete
```

Figure 152. Example of a tivoli.log File from a W95 Machine

- The line with the time stamp of `11:51:03` reflects information that is put into the log file when the agent starts up.
- The lines that have the time stamp of `11:52:06` show a request for a `wpcmngnode` from `itso3`.
- The line with the time stamp of `11:54:34` shows the connection with the source host of `itso3`.
- The lines with the time stamp of `11:55:46` shows the PC managed node finished receiving the file package and elapse time information.
- If there are other errors, they are usually captured in this file.

With the PC Agent version 5.001, both the Windows 95 and NT targets have a file lstagt.bat in the `c:\etc` directory. This file contains the target directory for the last distribution performed to the machine. This is shown in Figure 153:

On `rh2900c` the contents of the lststg.bat file is:

```
set LstAgt=c:\temp
```

Figure 153. Contents of lstagt.bat File

10.8.3 TMA Tracing and Other Log Files

When you are using the TMA endpoint from Software Distribution 3.6, the following shows the methods and processes you will see running on the various TMR, repeaters, and gateways participating in the distribution.

10.8.3.1 Odstat Trace for a TMA Distribution

This example demonstrates a distribution from a TMR (also acting as the source host) to a TMA endpoint through a repeater (also acting as the gateway).

```
17739 O+hdoq1-17703  done    18     0 15:27:02        1295714281.1.1099#FilePac
kage::FpoCore# fp_push
        service: "202 attr 'push_time' dat:..6u..:"  err=0  pid=18308
 17754 O+ho 1-17739  done    24     0 15:27:03        1295714281.1.348#TMF_Manag
edNode::Managed_Node# fp_dist
        service: ""  err=0  pid=79254
 17773 O a  1-17754  done   198     0 15:28:10        1351550138.1.585 rpt
17966 O+hdoq1-17754  done   957     0 15:31:16        1295714281.1.26 lookup
        service: ""  err=0  pid=23374
 17967 O+hdoq1-17754  done     6     0 15:31:16        1295714281.1.348#TMF_Mana
gedNode::Managed_Node# privileged_write_to_file
        service: ""  err=0  pid=81764
 17968 O+hdoq1-17754  done     6     0 15:31:16        1295714281.1.348#TMF_Mana
gedNode::Managed_Node# privileged_set_file
        service: ""  err=0  pid=81764
```

Figure 154. Odstat - TMR and Source Host

```
45 M hdoqs1-17773  done   198     0 15:33:22        1351550138.1.585 rpt
        service: ""  err=0  pid=0
    46 O+hdoq   1-45  done    95     0 15:33:22        1351550138.1.26 lookup
        service: ""  err=0  pid=0
    47 O+hdoq   1-45  done    90     0 15:33:22        1351550138.1.365 rpt_regi
ster
        service: "201 attr 'tune_1351550138.1.585' dat:"  err=0  pid=45
    48 O+      1-47  done    15     0 15:33:22        0.0.0 get_name_registry
        service: ""  err=0  pid=0
.
.
55 O+ahdoqs   1-45  done   198     0 15:33:26        1351550138.1.585 rpt
        service: ""  err=0  pid=0
    56 O+hdoq   1-55  done    95     0 15:33:26        1351550138.1.26 lookup
        service: ""  err=0  pid=0
    57 O+hdoq   1-55  done    90     0 15:33:26        1351550138.1.365 rpt_regi
ster
        service: "201 attr 'tune_1351550138.1.585' dat:"  err=0  pid=0
    58 O+hdoq   1-55  done    12     0 15:33:27        1351550138.1.365 _get_fin
al_timeout
        service: "201 attr 'final_timeout' dat:"  err=0  pid=0
```

Figure 155. Odstat - Repeater/Gateway for the TMA Endpoint

There are several differences you will see here. The first is that the source host no longer runs the `obj_route` method, but rather, does a simple lookup for the gateway. The gateway then does some lookup information and starts the repeater, but at this point, this is all we can see from the `odstat` output from TMR/source host and gateway/repeater. For more detailed information about which machines are participating in the software distribution, we will have to put the gateway into debug level 6 for verbose upcall, downcall, and repeater information. An example of setting the debug level is shown in Figure 156 on page 368.

```
bash$ wgateway rh2900a-gateway set_debug_level 0
bash$ wgateway rh2900a-gateway describe
Object      : 1351550138.1.585#TMF_Gateway::Gateway#
Hostname    : rh2900a
Port        : 9494
Timeout     : 300
bash$ wgateway rh2900a-gateway set_debug_level 6
bash$ wgateway rh2900a-gateway describe
Object      : 1351550138.1.585#TMF_Gateway::Gateway#
Hostname    : rh2900a
Port        : 9494
Timeout     : 300
Debug level : 6
bash$ wgateway rh2900a-gateway restart
```

Figure 156. Setting the Gateway Debug Level

Note

If the debug level is `0`, the `wgateway describe` command will not show a debug level.

You must restart the gateway if you change the debug level.

From the gateway log located in `$DBDIR`, you can see additional information concerning the upcalls, downcalls, and MDist parameters in use during the software distribution. An example of this is in Figure 157 on page 369. The lines beginning with bold numbers are explained in the text following the figure.

```
1 1998/12/14 15:33:26 +06: mdist: distribution ID = 1, method = fps_install, size = 0
1998/12/14 15:33:26 +06: mdist: Registering Repeater Manager: 1351550138.1.365
1998/12/14 15:33:27 +06: mdist: TMF_rptm_mgr::rpt_register called, tuning parms:
2 1998/12/14 15:33:27 +06: mdist: mem_max = 10000
3 1998/12/14 15:33:27 +06: mdist: disk_max = 50000
4 1998/12/14 15:33:27 +06: mdist: disk_hiwat = 50000
5 1998/12/14 15:33:27 +06: mdist: disk_time = 1
6 1998/12/14 15:33:27 +06: mdist: disk_dir = /Tivoli/db/rh2900a.db/tmp
7 1998/12/14 15:33:27 +06: mdist: net_load = 500
8 1998/12/14 15:33:27 +06: mdist: max_conn = 2
9 1998/12/14 15:33:27 +06: mdist: stat_intv = 180
1998/12/14 15:33:27 +06: mdist: Opening cache file:
C:\Tivoli\db\rh2900a.db\tmp\pmap3
1998/12/14 15:33:27 +06: mdist: in_spool_thread started: TID = 454140
1998/12/14 15:33:27 +06: mdist: out_spool_thread started: tid = 4544b0 client =
[1351550138.1.585]
10 1998/12/14 15:33:27 +06: downcall: Method body
/bin/w32-ix86/TAS/MANAGED_NODE/fp_endpoint found.
11 1998/12/14 15:33:27 +06: downcall: dependency /bin/w32-ix86/agentcli/wdskspc.exe
found.
12 1998/12/14 15:33:27 +06: idmap: user ($root_user,w32-ix86) -> Administrator
1998/12/14 15:33:28 +06: idmap: group ($root_group,w32-ix86) -> root
13 1998/12/14 15:33:28 +06: new_session: 19060027, connecting to
146.84.32.208+9494...
1998/12/14 15:33:28 +06: reader_thread: received data: session=19060027, type=9,
len=52
1998/12/14 15:34:46 +06: mdist: in_spool_thread finished: TID = 44e690
13 1998/12/14 15:34:46 +06: mdist: objcall_wrapup with timeout = infinite.
1998/12/14 15:34:46 +06: mdist: in_spool_thread finished: TID = 454140
1998/12/14 15:36:27 +06: reader_thread: received data: session=19060027, type=5,
len=134
1998/12/14 15:36:27 +06: destroying session 19060027
14 1998/12/14 15:36:27 +06: mdist: Finished out_spool to
1351550138.188.508+#TMF_Endpoint::Endpoint#
1998/12/14 15:36:27 +06: mdist: Result length for
1351550138.188.508+#TMF_Endpoint::Endpoint# = 64
1998/12/14 15:36:27 +06: mdist: out_spool_thread finished: tid = 4544b0 client =
[1351550138.188.508+#TMF_Endpoint::Endpoint#]
1998/12/14 15:36:27 +06: mdist: out_spool_thread finished: tid = 44e8c0 client =
[1351550138.1.585]
```

Figure 157. Gatelog Sample with Debug Level 6

- Line 1 in Figure 157 shows the `fps_install` method being invoked. The `distribution ID = 1`, corresponds to the information you will receive if you issue the `wrpt -L` command to look at the current distributions.
- Lines 2 - 9 show the repeater configuration for this distribution.
- Line 10 shows the downcall from the gateway to the TMA endpoint to ensure the `fp_endpoint` method is present on the endpoint.
- Line 11 shows the downcall for the `wdskspc.exe`, such as the `fp_endpoint` method. Reference 10.9, “Using the PC Agent w Commands on a TMA Endpoint” on page 370 for information on making the PC Agent w commands accessible from the TMA endpoint.

- Line 12 shows the mapping of the `$root_user` for the w32-ix86 interpreter is the Administrator ID.
- Line 13 shows the connection to the TMA endpoint that is the target of this distribution.
- Lines 14 through to the end show the connections to the TMA endpoint.

10.8.3.2 Additional Sources of Information for TMA Endpoints

In addition to the `odstat`, gateway log and software distribution log, there is an additional file present on the TMA endpoints in the LCF_DATDIR directory. By default the LCF_DATDIR directory is `c:\Tivoli\lcf\dat\1`. The file's name is tmesdist.log. This file contains information on the last software distribution that occurred to that endpoint. The information that is contained in the `tmesdist.log` is very similar to the information in the Software Distribution log. Occasionally, there is additional information. Also, if for some reason the Software Distribution log has been deleted or not received, you can get this log from the TMA endpoints. A sample of a tmesdist.log file is in Figure 158:

```
temp script: nt_before: c:/Tivoli/lcf/dat/1/Tiv169.bat
size=-1, nest_level=0
starting program: c:/Tivoli/lcf/dat/1/Tiv169.bat
'c:/Tivoli/lcf/dat/1/Tiv169.bat' script complete: status = 0
```

Figure 158. Output from TMA Endpoint tmesdist.log File

The corresponding Software Distribution log file is shown in Figure 159:

```
Source messages:
<none>
--------
pctmp109:SUCCESS
temp script: nt_before: c:/Tivoli/lcf/dat/1/Tiv169.bat
size=-1, nest_level=0
starting program: c:/Tivoli/lcf/dat/1/Tiv169.bat
'c:/Tivoli/lcf/dat/1/Tiv169.bat' script complete: status = 0
```

Figure 159. Log File from Software Distribution

10.9 Using the PC Agent w Commands on a TMA Endpoint

The TMA endpoint does not have the w commands that were previously used by the PC Agent available to it. Some of these commands were:

`wdskspc` Used to check the amount of available disk space prior to sending the file package to the target.

`wseterr` Used to set error codes in configuration scripts. This is used to signal Software Distribution to stop distribution on errors.

You can make these commands available to the endpoints in a number of ways. You can do a software distribution and distribute the commands to the TMA endpoints. The disadvantage to this method is that, if an executable changes, you will have to redistribute this executable to all the TMA endpoints.

A preferable option would be to use the `wdepset` command and download the commands to the cache area located on the TMA endpoint. If the executable should change, the gateway will reload the executable.

Note

For Windows executables, you will also need to discover which (if any) DLLs are also required.

You can only use the `wdepset` command to make commands available on the TMA endpoint. The `wdepset` command is available through the Application Developers Environment (ADE). Therefore, you will need to have ADE installed on all the TMRs in your environment where you wish to use `wdepset`. ADE installs like any other application.

The steps for making these commands available are:

1. Decide on the name for this dependency set. In this example we are using a dependency set name of `wcommands_depends`.
2. Select the binaries you wish to have placed on the TMA endpoint. If you are only using the `wdskspc` and `wseterr` commands, only download those commands. The binaries for these commands reside on the gateways, usually in the `$BINDIR/../lcf_bundle/bin/interptype/PCagent_clis`. It is advisable to check a gateway or two to validate this is indeed the case for your installation.
3. Determine a target location on the endpoint for these executables. In this example, we are going to place these executables out of cache. That is, this is a dependency file that is not deleted if the endpoint cache becomes full. If you wish to leave this in the endpoint cache, please reference - *Tivoli ADE - Application Development for the Lightweight Client Framework (3.6)*. The location can be in relation to the LCF_DATDIR directory. The LCF_DATDIR directory is an environment variable set by

the endpoint upon start-up. It can be viewed by specifying `-d3` (debug level 3) as a start-up parameter to the Tivoli Endpoint on the Windows NT Services GUI (Settings -> Control Panel -> Services) and then viewing the `lcfd.log` file. This location can also be a relative patch. We recommend placing it under the folders/directories where the TMA endpoint code was installed.

4. On the TMR server, you will now issue the command to create the dependency set, `wdepset`. The usage is as follows:

```
wdepset -c wcommands_depends -a w32-ix86 \
bin/w32-ix86/PCagent_clis/wdskspc.exe +p ../../ \
-a w32-ix86 bin/w32-ix86/PCagent_clis/wseterr.exe +p ../../
```

   Where `wcommands_depends` is the dependency set name and `w32-ix86` is the interpreter type (Windows NT). This is followed by the executable source path beyond `%BINDIR%/../lcf_bundle` and the executable is placed out of cache in the destination path beyond `%LCF_DATDIR%`.

5. After the dependency set is created with your executables, you will need a nested dependency set which will have the original `courier_lcf` dependency and your dependency, as follows:

```
bash$ wdepset -c all_swd_deps -a depset @DependencyMgr:courier_lcf \
> -a depset @DependencyMgr:wcommands_depends
(None)
depset:
        1351550138.1.762#Depends::Mgr#
        1351550138.1.1411#Depends::Mgr#
```

6. Once the nested dependency set is created, it must be associated with the Software Distribution methods so that the executables will automatically be downloaded, if needed, to the TMA endpoint prior to distribution. This is done by using the `wchdep` command. This command associates a dependency set with a particular method header. The method headers that are used are the `fps_install` and `fps_uninstall` methods. The class name for these methods is TMF_FP. An example of the usage is:

```
bash$ wchdep @Classes:TMF_FP @DependencyMgr:all_swd_deps fps_install
bash$ wchdep @Classes:TMF_FP @DependencyMgr:all_swd_deps fps_uninstall
bash$ wgateway rh2900a-gateway dbcheck
```

7. Finally, the gateway method cache must be synchronized with the TMR server. To do this use, the `wgateway` command. You must synchronize each gateway with the TMR server, as follows:

```
bash$ wlookup -ar Gateway
hptmp9-gw       1351550138.93.19#TMF_Gateway::Gateway#
rh2900a-gateway 1351550138.1.585#TMF_Gateway::Gateway#
rh2900c-gateway 1351550138.4.21#TMF_Gateway::Gateway#
bash$ wgateway hptmp9-gw dbcheck
bash$ wgateway rh2900a-gateway dbcheck
bash$ wgateway rh2900c-gateway dbcheck
```

8. The `wdskspc` and `wseterr` commands are now available for use from the TMA endpoint.

10.9.1 Removing the Dependency Set for Software Distribution

As Software Distribution is enhanced, you may need to delete the additional dependency set from the software distribution methods. To do this:

1. Use the `wchdep` to restore the `courier_lcf` as the dependency set on the `fps_install` and `fps_uninstall` methods and synch the gateway(s). The following is an example of how to do this:

```
bash$ wchdep @Classes:TMF_FP @DependencyMgr:courier_lcf fps_install
bash$ wchdep @Classes:TMF_FP @DependencyMgr:courier_lcf fps_uninstall
bash$ wgateway rh2900a-gateway dbcheck
```

2. Delete the dependency set from the Dependency Manager, as follows:

```
bash$ wdepset -d @DependencyMgr:all_swd_deps
bash$ wdepset -d @DependencyMgr:wcommands_depends
```

Chapter 11. AutoPack

The AutoPack is a resource type provided for Software Distribution since release 3.1. To distribute an application to a PC platform, you would generally have to include some form of configuration of the target system after the application itself was copied over. For example, there may be INI files to update or Windows registry entries to change. With the standard Software Distribution file package, you could achieve most of this through before and after scripts. AutoPack is a simpler way to achieve the same result by using a snapshot method of capturing configuration data. Note that AutoPack does not apply to the UNIX platforms.

While an AutoPack is a profile and resides in a profile manager much like a file package, AutoPacks enable you to distribute software without doing anything outside of its normal process. There is no need to create and run configuration programs. With AutoPack, all scripts necessary to install the software are built-in. The AutoPack utility identifies and contains instructions to make the necessary system changes when installing a software package. When you distribute an AutoPack, Software Distribution performs a scriptless installation on the target PC.

11.1 Introduction

AutoPacks, like file packages, are profiles that reside in a profile manager. The AutoPack tool can be used to install software for the following subscribers:

- PC managed nodes.
- Windows NT managed nodes.
- NetWare managed site clients.
- TMA Endpoints

There are three components that make up the AutoPack utility:

TMR server It must have Software Distribution AutoPack Version 3.1 and higher installed, and is needed to create AutoPack profiles.

ACC AutoPack Control Center. This is installed as a separate application on a PC, and is needed to create the AutoPack .PAK file that will be distributed.

AutoPack Agent This is installed on PC endpoints (Windows NT managed node or PC managed node) and automatically installed on

a TMA endpoint. It is needed to receive and unpack AutoPack packaged files.

11.1.1 PC Operating System Type Considerations

The following must be taken into consideration when creating and distributing AutoPacks:

- If the target is a Windows NT managed node, Software Distribution must be installed on the target's TMR server or an interconnected TMR server. Also, the AutoPack agent must be installed on the target.
- If the target is a PC managed node, Software Distribution must be installed on the target's TMR server or an interconnected TMR server. The PC agent and AutoPack agent must be installed on the target.
- If the target is a NetWare managed site, Software Distribution must be installed on the target's TMR server or an interconnected TMR server and the Tivoli NetWare Repeater (TNWR). The PC and AutoPack agents must be installed on the NetWare managed site's clients.
- If the target is a TMA endpoint, the agent is installed automatically when the TMA endpoint is first installed. Note that a managed node or a PC managed node can also be a TMA endpoint.

The following table illustrates which machine types can receive an AutoPack distribution based on where the AutoPack was created:

Table 15. AutoPack Distribution Source and Targets

AutoPack File Created On	**Distribution Targets**			
	Win 3.1	*Win 95*	*NT 3.51*	*NT 4.0*
Win 3.1	YES	NO	NO	NO
Win 95	NO	YES	NO	NO
NT 3.51	NO	NO	YES	NO
NT 4.0	NO	YES	NO	YES

11.2 Notes on Installing AutoPack

The *Tivoli Software Distribution Release Notes* specify the Framework and Software Distribution release levels that are necessary to install and use AutoPack. The following are the components and locations of Software Distribution that must be installed in order to use the AutoPack utility:

1. Install the Tivoli Software Distribution 3.1 (or higher) on the TMR server and on the managed nodes that have desktops.
2. Install the AutoPack Control Center on any PC that will be used to create AutoPack files. The installation is performed by running the SETUP.EXE program, which can be found in the PC/ACC directory on the Tivoli Software Distribution CD-ROM.

> **Note**
>
> The PC used to create the AutoPack images does not need to run any other Tivoli software.

3. Install AutoPack agent on any PC (Windows NT managed node or PC managed node) that will be receiving an AutoPack. This installation is done by creating an AutoPack profile for the AP_AGENT.PAK file found on the Software Distribution installation media. Once the profile is created, it can be distributed to any PC endpoint receiving an AutoPack. Note that a TMA endpoint already has the AutoPack agent installed by default.

11.3 AutoPack Control Center

An AutoPack profile comprises an AutoPack file that is created using the Autopack Control Center. An AutoPack file contains all files necessary to install an application including scripts that perform system configuration changes needed to install the selected application. You install the AutoPack Control Center on a PC in your Tivoli environment.

> **Note**
>
> You can install the AutoPack Control Center on any Windows 3.x, Windows 95, or Windows NT machine. The PC need not have the PC agent installed, and, in fact, Tivoli recommends that the PC have minimal software installed.

The AutoPack Control Center (ACC) should be installed on a pristine PC. A pristine system is recommended because ACC works by comparing a system before and after the application or data you wish to distribute has been installed. Many applications use the same .DLL files. Some application installations detect that the needed .DLL file is already installed on the PC and, therefore, do not try to reinstall it. If you create an AutoPack in this type of environment, the AutoPack will not pick up the .DLL because it was not installed. An attempt to install the .DLL application on a new PC that has no

other applications installed on it could fail because the .DLL would not be copied.

The AutoPack Control Center will allow modification of the definition file (.DEF) for the AutoPack at creation time. This allows for the addition of the necessary DLL at a later time.

The steps of installing the Autopack Control Center are explained in the *Tivoli Software Distribution User's Guide* and in the *Tivoli Software Distribution Autopack User's Guide* (from 3.6).

Note

Older versions of the AutoPack agent cannot parse an AutoPack file that is generated by the new AutoPack Control Center. If the following error appears in the PC agent error log after distribution, install a new version of the AutoPack agent on the client system and redistribute the AutoPack profile:

```
12/15/97 10:43:09 [E] Expecting 'target_dir =' at \
line 3, but found target_dir=C:
```

11.4 The AutoPack Agent

The AutoPack agent facilitates the distribution and installation of application files on the PC. Before the AutoPack agent can be installed, the PC must be a Windows NT managed node or a PC managed node (The PC agent must be installed and connected as a PC managed node) or the PC must be installed as a TMA endpoint that automatically installs the AutoPack agent. With the senior or super authorization role, the AutoPack agent can be created from the Tivoli desktop as follows:

1. In the Policy Region dialog, add the AutoPack and Profile Manager resources to the policy region's list of managed resources.
2. Create a profile in which the AutoPack will reside.
3. Set the AutoPack's properties in the Set AutoPack Properties dialog.

The steps of installing the AutoPack agent are explained in *Tivoli Software Distribution User's Guide* or the *Tivoli Software Distribution Autopack User's Guide*.

11.5 AutoPack Properties and Operations

An AutoPack is a Tivoli Software Distribution profile that is similar to a file package but provides a simpler, more convenient way to create installable images for Windows, Windows 95, and Windows NT PCs managed by Tivoli. An AutoPack profile comprises an AutoPack file that you create using the AutoPack Control Center, which is a Software Distribution product that is installed on a PC. The AutoPack file contains all files necessary to install almost any application, such as Microsoft Word or the Netscape Navigator, including scripts that perform system configuration changes needed to install the selected application.

After you create and edit the properties of an AutoPack, you can distribute the AutoPack to its subscribers (Windows NT managed nodes, PC managed nodes, NetWare managed site clients, TMA endpoints, and other profile managers). When you distribute the AutoPack, the application files contained in the file are distributed to the target machines.

11.6 Creating an AutoPack

1. Install the AutoPack Control Center on a Windows, Windows 95, or Windows NT machine. This PC should have minimal software installed.
2. Run the AutoPack Control Center to create an AutoPack file. Creating this file entails:
    1. Taking a pre-install, or baseline, snapshot of the PCs drive and system configuration. AutoPack scans up to two drives (system root and destination drive) taking a snapshot of the file system and collecting the contents of certain system files that are stored in the baseline directory.
    2. Installing the desired software package. User installs the desired application or applications using the application install method, FTP, or any other method.
    3. Taking a post-install snapshot. AutoPack scans the PC a second time collecting the same file system and configuration file snapshots and creates a file package definition file.
    4. Specifying distribution options for the AutoPack file.
3. Build the AutoPack file. Once any desired changes are made to the definition file (.DEF), change file (.CHG), or replace file (.REP), the AutoPack (.PAK) file is created.

4. Create an AutoPack profile on a managed node using the Tivoli Desktop or command line. You cannot distribute an AutoPack file directly from the PC on which it was created; it has to reside on a managed node, and a system would not be pristine if it were a managed node. Thus, you must copy the AutoPack file from the PC to a managed node and associate it with an AutoPack profile. See “Profile Setup” in the *Tivoli Software Distribution User’s Guide* or the *Tivoli Software Distribution Autopack User’s Guide* for information on creating, cloning, setting subscribers for, and deleting AutoPacks.
5. Set the properties of the AutoPack, specifying where the AutoPack file will reside as described in this chapter. In this step, the file is actually copied from the PC where it was created and saved with the AutoPack profile.

Note

You cannot create an AutoPack file to perform OS upgrades and installations.

For information about Setting AutoPack properties, calculating the size of an AutoPack, distributing an AutoPack, and removing an AutoPack, see the *Tivoli Software Distribution User's Guide* or the *Tivoli Software Distribution Autopack User’s Guide*.

Note

Removing an AutoPack removes all distributed files and directories on the target machines and reverses any system changes performed by the AutoPack.

11.6.1 Pre-Scan

The first step to creating an AutoPack is the pre-scan. The scan begins with the user choosing the target hard-drive where the application will be installed. When the scan starts, it will scan the target hard-drive and the system drive (where the operating system is installed). The list of files that the scan will search through can be modified through the ACC dialog or through the AUTOPACK.INI file located in the `%SYSTEMROOT%` directory.

Note

Users should make changes to the `Files to Monitor` and `Files to Exclude` lists instead of the AUTOPACK.INI file.

The pre-scan gathers the following information and stores it in *.1ST files found in the baseline directory on the PC:

- AUTOEXEC.1ST - Current contents of the autoexec.bat file on the system drive.
- CONFIG.1ST - Current contents of the config.sys file on the system drive.

> **Note**
>
> If the target directory is different from the system drive and the AutoPack Control Center discovers an autoexec.bat and/or config.sys on both drives, it will gather both versions.

- FILESYST.1ST - A list of all of the directories and files on the installation and system drives.
- INIFILES.1ST - Current contents of all of the .INI files in the system root directory.
- REGISTRY.1ST - Contents of the HKEY_LOCAL_MACHINE\SOFTWARE and HKEY_CLASSES_ROOT registry hives (for Windows 95 and Windows NT scans).

> **Registry Tip**
>
> Check which registry hives your version of AutoPack is monitoring. You can edit the autopack.ini file in the [RegistryMonitor] section to include, for example, `HKEY_LOCAL_MACHINE\System\CurrentControlSet`. AutoPack will then automatically detect any newly-created services and will create them when the AutoPack is installed on another system.

- DESKTOP.1ST, STARTMENU.1ST - All of the desktop icons and contents of the start menu (for Windows 95 and NT 4.0).
- PROGRAMS.1ST - Contents of program groups (common on Windows NT).

11.6.2 Software Installation

The second step in creating an AutoPack is to install the software on the PC. This can be any number of software packages or could even be FTPing the necessary files to the PC.

11.6.3 AutoPack Build

There are three parts to the post-installation step. First, the post-scan creates *.2ND files which correspond to all the *.1ST files created during the pre-scan

step. The second part generates the files that will be used to create the AutoPack image:

- .DEF AutoPack definition file. This file is the Software Distribution definition file that is used to create the .PAK file. It contains the after and removal script paths and arguments as well as a listing of the application files to install.
- .REP AutoPack replace files. This is a list of the files that were replaced during the software installation.
- .CHG Contains a description of all the system files that need to be changed.

You can change any of the basic distribution settings. Once complete, the *.PAK file is created. This is the only file that needs to be kept, although, if the three configuration files (*.DEF, *.REP, *.CHG) are kept along with the installed files, then any necessary changes can be made and a new *.PAK file can be created at any time.

11.6.3.1 The .CHG File

The .CHG file contains all of the system and registry changes that must take place on the target PC. It includes the following:

- arguments - Lists the target directory, staging directory, log level, and prep machine's information.
- file_system - Includes changes to the `%SystemDrive%` directory, which includes shared file systems.
- win_registry - Includes keys and values added to the registry for Windows 95 and Windows NT machines. The monitored keys include HKEY_LOCAL_MACHINE\SOFTWARE and HKEY_CLASSES_ROOT.
- win-inifile - List all .INI files in the `%SystemRoot%` directory, the %WinDir% directory for Windows and Windows 95 machines, and in the Files to Monitor property sheet.
- win_programs - Includes program manager changes, such as shortcuts, icons, and groups, and the Start Menu/Programs folder on NT 4.0 and Windows 95 machines.
- win_explorer - Lists Windows Explorer changes including shortcuts and folders for Windows 95 and NT machines.

Note

Briefcases in Microsoft Office are not supported.

- config_sys - Includes the config.sys file on the C: and %SystemDrive% drives. The AutoPack Control Center will always monitor this file even if it is removed from the list.

> **Note**
>
> Only the DOS 5.0 format of this file is supported.

- autoexec_bat - Includes the autoexec.bat file on the C: and %SystemDrive% drives. The AutoPack Control Center will always monitor this file even if it is removed from the list.

> **Note**
>
> Only the DOS 5.0 format of this file is supported.

11.6.3.2 The .DEF File

The .DEF file lists all of the properties of the AutoPack as displayed in the AutoPack Contents dialog. This file can be modified before the *.PAK file is created. This allows some of the system-specific values to be changed. For example, you can create one AutoPack that installs a software application on C: and another AutoPack that installs the same software application on D:. The file list in the *.DEF file can also be modified if there were .DLL files that were not installed because they already existed on the machine.

11.6.3.3 The .REP File

The .REP files list the files that were replaced during the installation and will be replaced in the target PCs during the AutoPack distribution. This file will be empty if no files were replaced during the installation.

11.6.3.4 The .PAK File

The AutoPack file is created using sapack (executable for pack software, this file comes with the Tivoli installation), the .DEF file, the .CHG file, and the application installed files and directories. AutoPack uses a new version of sapack that supports staging and destination locations for files.

11.6.3.5 The .ERR File

If there is a problem creating the .PAK file from the generated files, then a .ERR file is created containing any errors that were generated.

11.6.4 AutoPack Properties

After creating the AutoPack .PAK file in the AutoPack Control Center, a profile representing the AutoPack has to be created on the TMR server. After the profile is created, the properties must be set providing the source host and location of the AutoPack file. The profile properties can be set up in one of three ways:

- The AutoPack file is on a managed node, and the node name and path are used in the AutoPack properties dialog. The image can be created on another system and copied by hand to the specified managed node.
- The file is on a PC managed node and copied to a specified managed node. In this case, provide the PC name and path where the image is stored and the managed node name and path where the image is to be copied.
- The file is on a managed node and needs to be copied to a specified managed node.

11.7 Distributing AutoPack Profiles

Having created the AutoPack profiles and set up the desired properties, we can now use them to install and remove software.

11.7.1 AutoPack Install of Software

After the AutoPack properties are set, it can be distributed to any supported PC endpoint or to a profile manager that has PCs as subscribers. The AutoPack is distributed as any other Software Distribution file package (meaning same Software Distribution methods are running).

> **Note**
>
> The PC endpoint must already have the AutoPack agent installed.

When the endpoint receives the AutoPack, it will lay down the application files as listed in the .DEF file. It then runs the `wsysupd.bat` script that runs the `wsyschg.exe` program passing it the keyword `after` (%1) and the path to the .CHG file (%2) that is the input file for the script. This after script will make all the necessary changes to the system file.

11.7.2 AutoPack Removal of Software

The software installed through AutoPack can be removed from the PC endpoints by using the **Remove from hosts...** option on the AutoPack. When

this option is used, the request is sent to the PC endpoint, and the `wsysupd.bat` script is run as a removal script passing the keyword `remove` and the path to the .CHG file as arguments to the `wsyschg.exe` program. This will remove all the changes that were made to the system files and remove all the installed files and directories.

11.8 AutoPack Policy

There are two default and three validation policy methods provided for the AutoPack profile.

11.8.1 Default Policy

- ap_def_autopack_file - Sets the default path for new AutoPack profiles.
- ap_def_autopack_host - Sets the default host for new AutoPack profiles.

11.8.2 Validation Policy

- ap_val_autoapck_host_file - Validates the set source host and source path for new or modified AutoPack profiles.
- ap_val_name - Validates the name given to an AutoPack profile.
- ap_val_operation - Validates the following:
  - Distribute/Remote - If the AutoPack profile is being distributed or removed from PC endpoints, then the validation policy is given the name of the profile and a list of targets.
  - Copy - If the AutoPack is copying from a PC managed node or NT managed node to a target managed node, then the validation policy is given the name of the profile, the host name and path of the source machine, and the host name and path of the target machine.

11.9 Troubleshooting AutoPack

Since the introduction of Software Distribution Version 3.1, the AutoPack Control Center for Windows has included on-line help. This contains a detailed description of all of the features of the AutoPack Control Center. It is a good source for troubleshooting information as well as a user's perspective of the tool - you should also review the information now contained in the troubleshooting section of the *Tivoli Software Distribution Autopack User's Guide*.

11.9.1 Common Problems

The following is a list of common AutoPack problems:

- Locked files preventing the package contents from being written on the target.
- Operations that require re-booting need to be identified. Some may still be possible to achieve.
- Operating system upgrades are generally not possible due to version considerations, locked files, and so on.
- Installing Exceed, the xclient product, is known to have problems.

These problems require additional actions that can not be done by AutoPack.

11.9.2 Where to Find Error Information

When creating the AutoPack file, there is one log file created:

.ERR File The AutoPack Control Center's log file. It will contain any errors that occurred while the AutoPack file was being created.

When installing or removing AutoPack files, three log files are generated:

<log file> When setting up the AutoPack file, there is an option to log to a file. To set this up, the user provides the host name and path of where to write the log file. This file will contain errors relative to the distribution from the source machine to the endpoints.

autopack.log Records messages or errors when installing or removing AutoPack files.

tivoli.log The PC agent log file.

Chapter 12. Distributed Monitoring

Tivoli Distributed Monitoring is used to monitor various local system resources in any TMR server, managed node, NetWare Server, or TMA endpoint in the Tivoli framework and generate Network-wide events and alarms. A Proxy facility also allows the extension of monitoring to systems outside the Tivoli Management environment.

One or more monitors, which will be distributed to the target machines, are defined within a profile - known as a SentryProfile. This resides, just like other profiles, in a profile manager to which targets can be subscribed (see Figure 160).

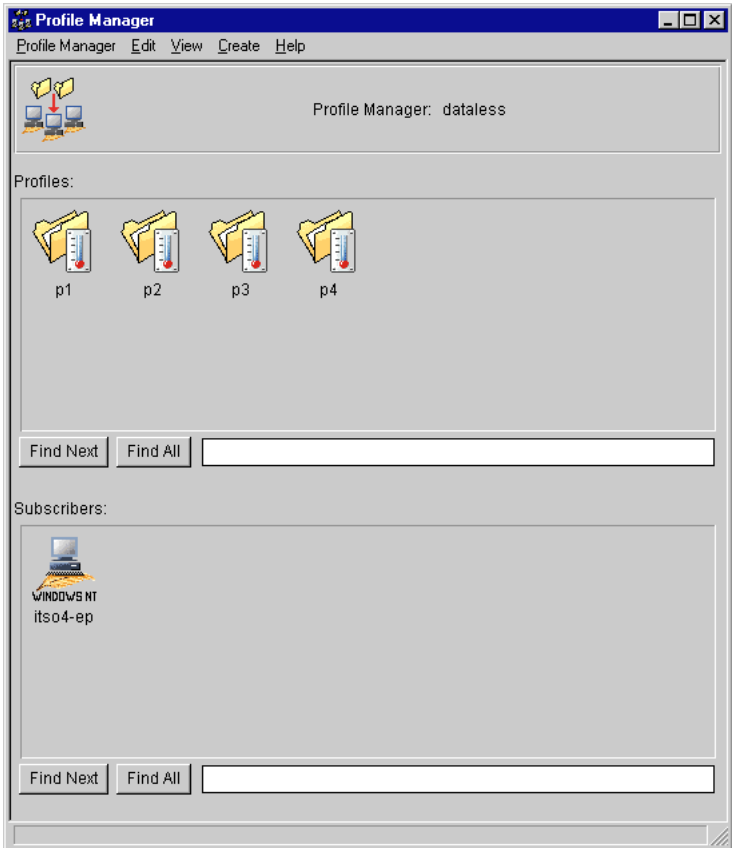


Figure 160. Distributed Monitoring Profiles in a Profile Manager

Each of the individual monitors are based upon a standard monitor, which is provided in a set of *Monitoring Collections*. A Monitoring Collection is

basically a set of programs, each designed to request specific information from the system on which it is run, that return a value to the monitor. Along with the programs, a set of definitions ensures that the monitor is given the correct number and types of arguments when the monitor is created. Many monitors can be added to a single SentryProfile, which is the smallest single entity that can be distributed to one or more endpoints.

Each SentryProfile can be associated with an *IndicatorCollection*. This is a common object where the result of an execution of any monitor from the SentryProfile at any endpoints is logged. Optionally, an icon on the Tivoli Administrator GUI can be updated in order to provide a visual indication of a critical situation or a pending problem.

When defining a monitor, a monitor is selected from a monitoring collection, and information for controlling the behavior of the monitor is supplied. This control information describes the hows, the whats, the whens, and so forth, but not the whys and the wheres. The wheres are determined when the monitor is distributed, and the whys are a matter of decision in each individual implementation.

If the built-in monitoring collections do not exactly meet the requirements for monitoring, custom monitor collections can be developed and added to the Distributed Monitoring environment, but the same deployment considerations apply as for the monitoring collections supplied by Tivoli. However, the use of monitoring collections custom-built from scratch is rare. Instead, customized monitors are very easily built using standard shell-monitors from the Universal monitoring collection provided by Tivoli.

Monitoring can be asynchronous, event-driven, or based on polling at defined intervals. Basically, all monitors in the standard monitoring collections are designed to support exception handling - creating an alarm if the actual value meets certain criteria. The standard monitors do not report historical data or react to average values. To obtain this functionality, custom script collections should be developed.

At the endpoints, a Sentry Engine controls the monitoring process itself, determining when each monitor should be fired and launching automated responses. Each monitor from every SentryProfile that has been distributed to the endpoint is represented in the Sentry Engine’s working area - the *Engine Database* - as a *monitoring probe*. These components are shown in Figure 161.

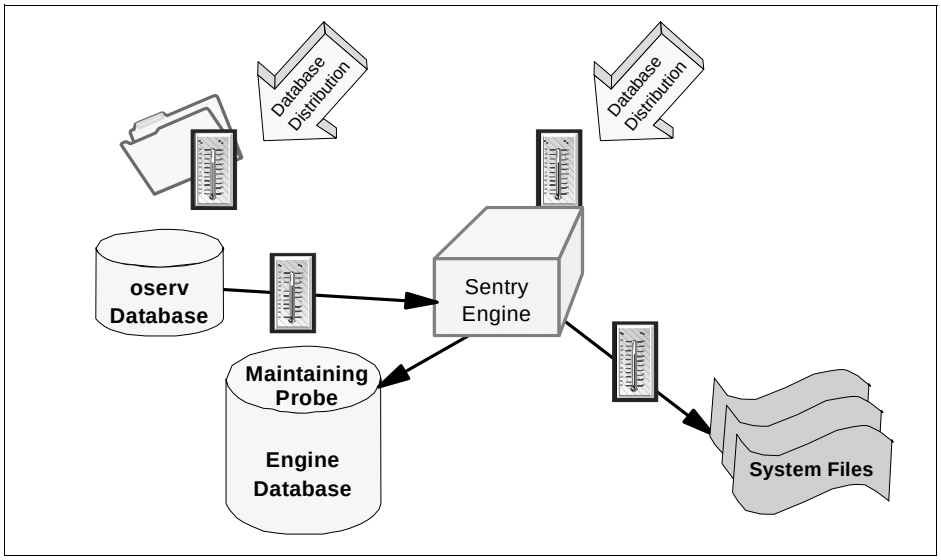


Figure 161. Distributed Monitoring (Sentry) Entities

Distributed Monitoring uses the familiar management by subscription paradigm of the Tivoli Framework; and policies governing the creation, modification, and distribution can be applied.

Tivoli Distributed Monitoring supports customisations to X/Open message catalogs to provide more detailed log information. Also supported are Proxies to allow monitoring of resources hosted on other targets including non-Tivoli-enabled equipment and the use of additional environment variables.

Also a graphing capability using a standard browser is available. Refer to the *Tivoli Distributed Monitoring User's Guide* or the redbook *A first look at TME 10 Distributed Monitoring 3.5,* SG24-2112.

12.1 New Features in Distributed Monitoring Version 3.6

In Tivoli Distributed Monitoring version 3.6, support for TMA endpoints has been implemented. Since TMA endpoints can be implemented as a supplement to a managed node, it is preferred to install the TMA on a managed node that just needs to be monitored. If there is a need to create or modify SentryProfiles at a managed node, the full Distributed Monitoring application should be installed on the managed node.

12.2 Installation Considerations

Deploying Distributed Monitoring requires detailed planning. Not only planning of the actual monitoring process - what to monitor, where to monitor, when to monitor, how often to monitor, how to react to the monitor responses, and so forth - but also planning of what binaries need to be available and where and when.

Along with the Distributed Monitoring application itself, the product is shipped with a number of built-in monitoring collections - groups of programs that are used for monitoring specific resources within all or specific types of monitored endpoints. These are made available to all types of endpoints by installing them on the TMR Server.

12.2.1 The Distributed Monitoring Application Install

Distributed Monitoring is installed on TMR servers and managed nodes just like any other Tivoli application using either Software Installation Services, the GUI, or the `winstall` command. No specific installation activities, except for installing the TMA endpoint itself, are required to enable monitoring on TMAs.

Distributed Monitoring, and all the monitoring collections used, have to be installed on the TMR Server. Netware Monitoring Collections, if used, additionally need to be installed on all gateways.

To distribute the SentryProfiles (activate monitoring) to any other managed node besides the TMR server itself, Distributed Monitoring should be installed on each managed node that is to be monitored. This ensures that the Distributed Monitoring binaries are present on each managed node and ready for use. Another way to enable monitoring at a managed node is to install a TMA on the managed node. The managed node is then enabled for monitoring without using disk space locally for the Distributed Monitoring binaries. In order to avoid problems, only one implementation of Distributed Monitoring should be active on a managed node at any one time. The use of both the full managed node and the TMA implementations of Distributed Monitoring on the same node should be restricted to migration usage.

To activate monitoring on TMA endpoints, Distributed Monitoring has to be installed on the gateways that the TMAs are connecting through. By installing Distributed Monitoring on the gateway, the binaries used by the TMA become available to the TMA and are downloaded from the gateway when needed. To be absolutely certain that the TMAs can get the required binaries, disregarding the state of the preferred gateway, Distributed Monitoring should

either be installed on any gateway that accepts logins from monitored TMAs or allow_install and login policies for the TMA endpoints should be developed to check for the need for, and existence, of a proper Distributed Monitoring installation on the current gateway.

Since gateways can currently only be implemented on managed nodes, the Distributed Monitoring installation process is similar to that on the managed node.

By installing Distributed Monitoring on a gateway, the managed node that hosts the gateway automatically becomes a candidate for monitoring.

12.2.2 TMA Endpoint Distributed Monitoring Install

No specific installation process is needed to activate Distributed Monitoring on a TMA endpoint. The files and directory structures that are required to run Distributed Monitoring on a TMA are downloaded from the gateway at the time the first SentryProfile is distributed to the TMA. This is handled in the following way:

1. The Sentry Endpoint method - dogendpoint - is invoked as a normal download method. After the first invocation, it will be in the method cache on disk at the endpoint the first time it is automatically downloaded by the endpoint process (`lcfd`). By default, the endpoint method is cached in the directory:

   `($LCF_DATDIR)\CACHE\BIN\$(INTERP)\TME\SENTRY`

2. Dogendpoint discovers that the Sentry Engine (dm_ep_engine) and the GNU-tools for the required platform that it depends upon are not installed. It invokes an upcall to retrieve them from the endpoint gateway. Note, that although these modules are being downloaded and run in a very similar way to ordinary endpoint methods, they will not be stored in the method cache. Instead, they are stored in a subdirectory structure off the current dat-directory ($LCF_DATDIR) of the TMA.

Note

The reason for this is to make sure that all required files are available to the engine when a monitor is triggered. In the usual TMA environment, all required files are downloaded from the gateway (if not found in the TMA cache) and flushed from the cache when space is required to hold other files. The implications of downloading files for monitoring in a cache-constrained environment could lead to very lengthy monitor transactions and even inaccurate responses from the monitors.

Figure 162 shows the Distributed Monitoring directory structure for a TMA endpoint:

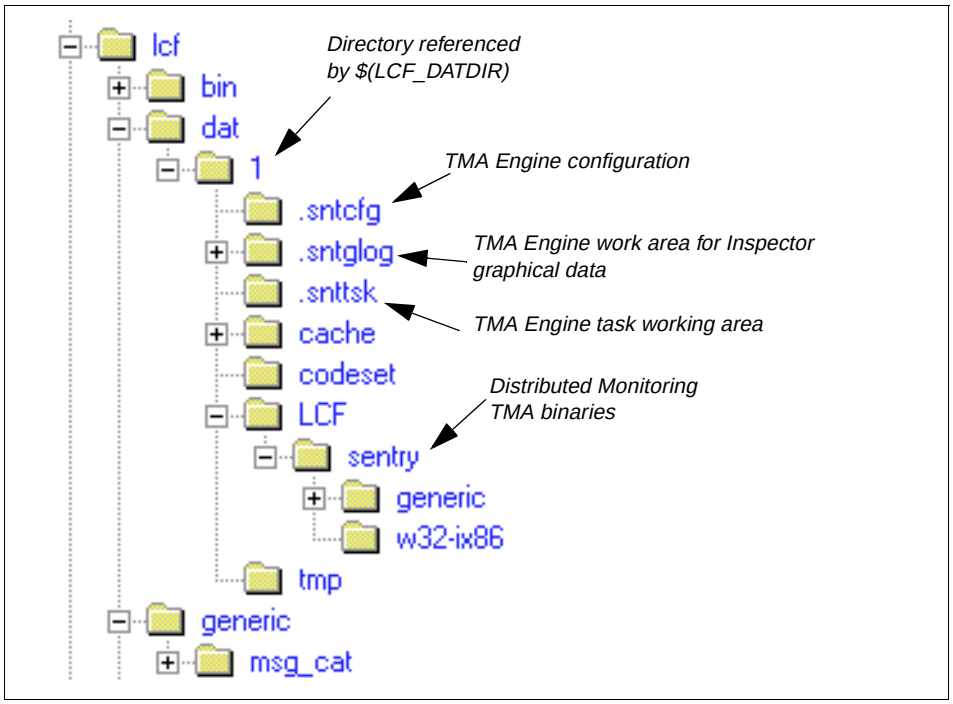


Figure 162. Directories Used by TMA Endpoints for Distributed Monitoring (3.6)

A positive side effect of this implementation is that the Distributed Monitor engine on TMAs does not rely on resources from the gateway. Once launched from the TMA, the engine is independent from the TMA itself. This enables the Distributed Monitoring engine to perform its duties from the point when the TMA is started until the engine is stopped (using the `wstopeng` command), the system is shut-down, or the engine-program (dm_ep_engine.exe) is killed.

12.2.3 Monitoring Collections

The monitoring collections are only installed on the TMR server. When installed, each monitoring collection is added to the framework using an object of the type MonitoringCapabilityCollection.

Whenever an engine needs to run a monitor from a collection that is not currently loaded, the monitor is requested from the TMR server.

12.3 Getting Started with Distributed Monitoring

When working with Tivoli Distributed Monitoring, the following four basic components are used:

- Distributed Monitoring profiles (SentryProfiles)

  A Tivoli Profile designed to hold monitor collections. The SentryProfile holds information common to all monitors in the collection, some of which are:

  - IndicatorCollection to use
  - Defaults for distribution
  - Default monitoring schedule

- Monitors

  A record within the SentryProfile defining the characteristics of a monitor. The monitor definition includes:

  - Monitor to use - from MonitorCollection
  - Resource to be monitored - if applicable
  - Threshold levels
  - Monitoring schedule
  - Responses

- Monitoring Collections

  Collections of standard monitors grouped by capability. In addition, user-supplied monitoring probes, such as shell scripts and binaries, can be incorporated as well as monitors provided by other Tivoli applications. Scripts and executables can be distributed as part of the monitor setup process.

- Indicator Collections

  List of indicators used to determine the state of a Tivoli Distributed Monitoring profile and log information on each SentryProfile in the collection.

When monitoring, four default severity levels are provided: Critical, severe, warning, and normal. In addition, user defined severity levels can be added. Based on the severity level of a monitor value, various automated actions can be invoked. The following is a list of standard responses:

- GUI pop-ups.
- Visible changes to icons in IndicatorCollections.
- E-mail that can be sent to any valid e-mail address (not limited to Tivoli administrators).

- Tivoli notification to a designated notice group.
- File logging - local or on a remote managed node.
- Local or remote user-defined response agents (programs or tasks).
- Posting of Tivoli Enterprise Console (TEC) events. When passing events on to TEC, the severity level can be mapped to those used by TEC. From Distributed Monitoring 3.6 onwards, a primary and a secondary EventServer can be specified.

12.4 Defining Monitors

Prior to defining monitors in Distributed Monitoring, the following three prerequisites have to be in place:

1. Profile managers should be created. To hold the SentryProfiles database type, profile managers should be used, and use of dataless profile managers should be for TMA subscribers only.
2. The managed resources type SentryProfile should be added to the managed resources of the Policy Region in which the profile manager is created.
3. The current administrator should be granted one or more of the following Tivoli roles:

   super in order to create SentryProfiles and set policies

   admin to edit and distribute SentryProfiles, maintain subscribers, and create and manipulate IndicatorCollections

   user browse SentryProfiles and IndicatorCollections

Creating monitors in Distributed Monitoring requires two steps:

1. Add monitors
2. Edit monitors

Through the Tivoli desktop, users add and edit Tivoli Distributed Monitoring monitors with the dialogs shown in Figure 163 and Figure 164 on page 396.

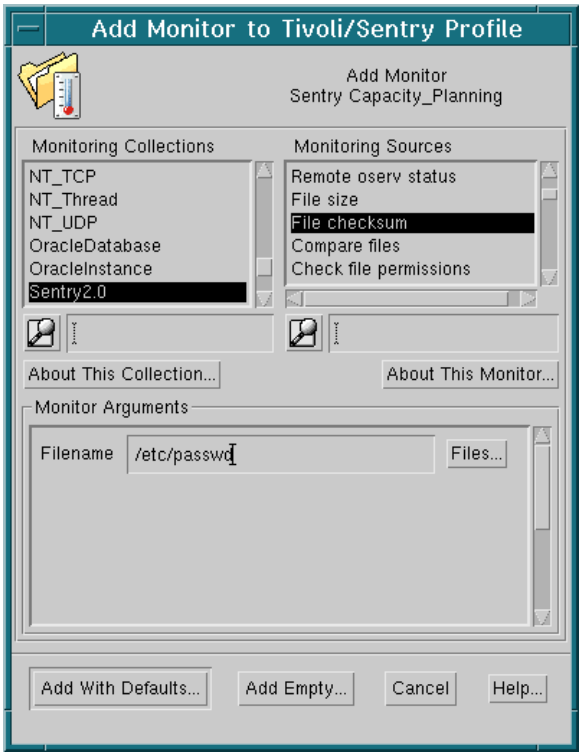


Figure 163. Select a Tivoli Distributed Monitor

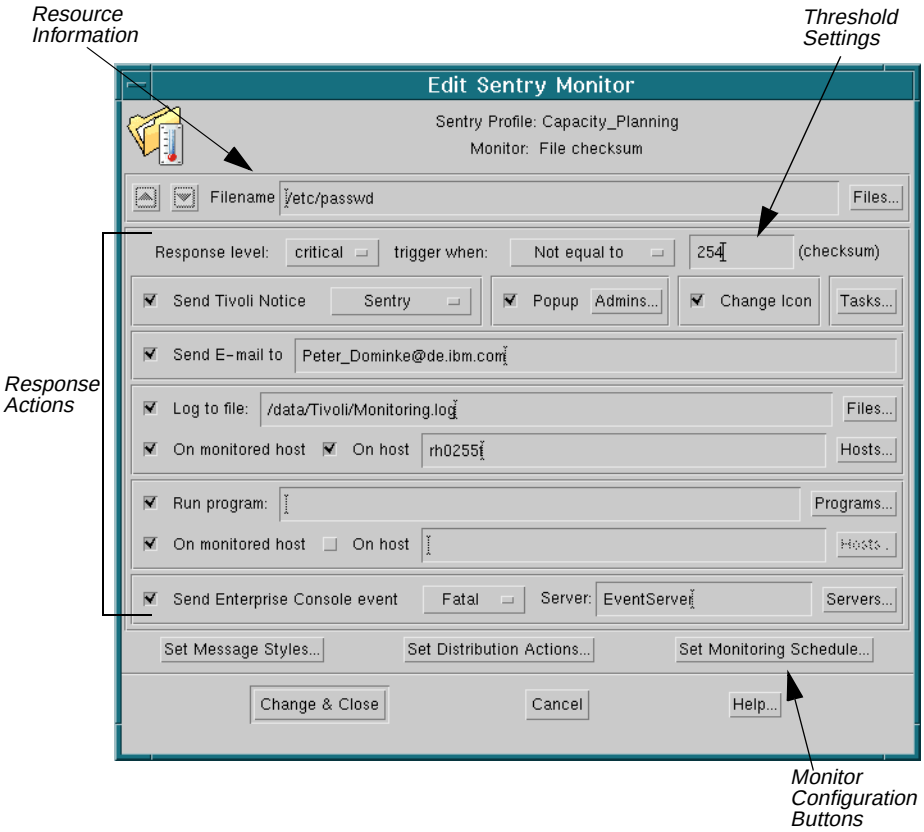


Figure 164. Edit a Tivoli Distributed Monitor

12.5 Customizing Tivoli Distributed Monitoring

In addition to the built-in monitors found in the monitoring collections, custom monitors can be added when required. This can be accomplished in one of two ways:

- The simplest way is by supplying, and optionally distributing, in-line monitoring scripts used with the *User Monitor-Numeric Script* and *User Monitor-String Script* monitors of the Universal monitoring collection.
- The complicated way is by adding custom made monitoring collections. This should only be used to apply Distributed Monitoring capabilities to subsystems and customer application systems.

The User Monitor Script monitors are time-driven. They are fired, like most other monitors, in accordance with a schedule defined for each monitor. In some cases, it may be more adequate to define monitors that are truly event-driven. These would be fired whenever a specific event occurs. To support those types of adapters, the Universal monitoring collection supplies two special monitors: *User Monitor - Asynchronous string* and *User Monitor - Asynchronous numeric*, which can be fired from programs or custom scripts using the `wasync` command - whenever the specified event is detected. A typical use of this feature is reporting specific events to Distributed Monitoring from within Tivoli (and non-Tivoli) tasks and jobs, such as scripts used for backup, software distribution, and so on.

12.5.1 User-Defined Monitors

Tivoli Distributed Monitoring supports the easy addition of user-defined custom monitors. A user can supply the path to a binary executable or shell script that will return a numeric or string value that can be evaluated. This allows users to create monitors that specifically return the information that might be unique to their environment. Custom monitors can be registered - using the `waddcust` command and made available to any new Tivoli Distributed Monitoring node that is created although this is not the same as adding monitors through collections created with the Monitoring Capabilities Subscription Language (MCSL). See the next section for more information on collections.

Custom monitors and asynchronous monitors support the use of comments that will be passed to responses for added information. The types of responses that will utilize these comments are pop-up dialogs, TEC events, Tivoli notices, e-mail, log files, and custom responses.

When using custom monitors, the script or executable must be local to the Tivoli Distributed Monitoring engine on the endpoint. Tivoli Distributed Monitoring provides a way of copying the custom scripts at the time the SentryProfile is distributed to the endpoints. The Distribution Actions dialog is shown in Figure 165 on page 398.

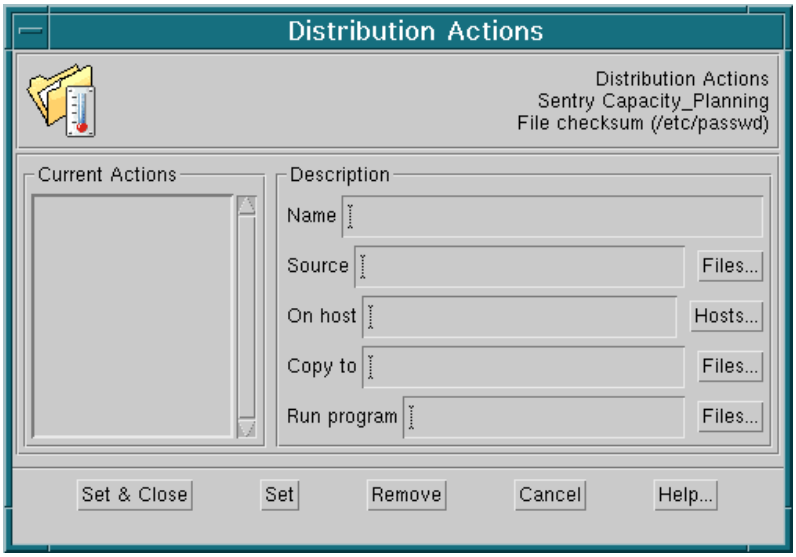


Figure 165. User-Customized Distribution Actions for Monitors

12.5.2 Asynchronous Monitors

Asynchronous monitors respond to events sent to the Tivoli Distributed Monitoring engine tagged with a given channel name. Channel names are determined by users and may be registered with the `waddchan` command to be available from the desktop.

Registering Channels

It is recommended that all asynchronous channels are registered even though it is not mandatory. When defining an asynchronous monitor using the GUI, the channel name can simply be entered in the input-box, and no checks are made for the existence of the channel. If the channel was added using `waddchan` prior to the monitor definition, it will appear in the pulldown-list of available channels.

When using the channel in a `wasync` command, no checks are made to ensure that the channel is registered. If the syntax of the `wasync` command is correct, no error messages will be issued to indicate that the channel specified does not exist.

In short, it is the responsibility of the administrator to ensure that the channel name used in monitors and `wasync` commands correspond. The `waddchan` command is only used to ease monitor definition using the GUI, and for documentation purposes, using the `wlschan` command.

Tivoli Distributed Monitoring supports two different timing options for resource monitoring: Polled and asynchronous. Polled timing is the most common. It uses a scheduler to determine when a resource will be monitored. One example is monitoring the host status every two hours.

The second form of timing is asynchronous. This is used by creating a channel that the Tivoli Distributed Monitoring engine will listen for activity from. When some outside source detects a predefined threshold, a signal is sent to a specified Tivoli Distributed Monitoring channel to trigger the event.

There are three ways to report events to Tivoli Distributed Monitoring asynchronously:

- Use the `wasync` command - This command can be used on the command line or through a script:

  ```
  wasync -c WhizzoEvent -s 1 -i ‘Event Info’.
  ```

wasync note

In Tivoli Distributed Monitoring 3.6, the use of the `wasync` command is restricted to users who are authenticated in the Framework with an ID that maps to root_user.

Use the `widmap` command to check for authorization mappings.

- Use the IDL interface structure - This method has to be used within a C program.
- Use a C function call that bypasses the Object Request Broker (ORB) - This method has to be used within a C program and communicates directly with the Tivoli Distributed Monitoring engine process.

Note

See the appendices of the *Tivoli Distributed Monitoring User's Guide* for the syntax of these system calls.

12.6 Tivoli Distributed Monitoring Proxies

A Distributed Monitoring Proxy is basically an object of the SentryProxy class residing on a managed node. It is used to monitor, and report status, from a resource residing outside the current Tivoli environment or to monitor resources hosted by a machine/device different from the hosting managed node. An example of using SentryProxies to manage resources outside the hosting managed node is to monitor the status of the IP-interfaces on another managed node, thus, providing reporting of IP-problems for the monitored node even if the node's own IP-interface is broken.

When a proxy is defined, the managed node on which it resides is determined as well as the name of the proxy. Profiles are distributed, as normal, using the proxy as subscriber; and whenever the monitors report status, the name of the proxy is used as *reporting node*.

The actual monitor running on the proxy-endpoint is not aware that it is, in fact, running on a proxy-endpoint. Therefore, only customized scripts capable of performing the actual monitoring of the remote resource should be used with proxies. This is why a filter can be applied to a proxy-endpoint to keep monitors from specific monitoring collections from running on the proxy-endpoint.

Tivoli Distributed Monitoring supports the use of proxies to pass additional environment variables to custom sentries/responses and to allow monitoring of non-Tivoli-managed endpoints.

12.6.1 Distributed Monitoring Environment Variables

Typical Tivoli Distributed Monitoring endpoints (managed nodes) have a set of environment variables that are passed and used by custom monitor and custom response-scripts. The following is a list of environment variables that

are available to both typical monitors and to Tivoli Distributed Monitoring proxies:

Engine specific variables:

- ACCEPTABLE_VALUE
- CHANGE
- COMPARED_TO
- CRITERION
- DELTA
- EFFECTIVE_VALUE
- HOSTNAME
- INTERNAL_ID
- INTERP
- LASTSTAMP
- MONITOR
- MONITOR_ID
- NAME
- o_dispatch
- PATH
- PREV_VALUE
- PROBE
- PROBE_ARG
- PROFILEOID
- RELATION
- RESPONSELEVEL
- TIME
- TZ
- UNITS
- USERINFO
- VALUE

Implicit endpoint variables:

- ADMIN - Name of administrator who most recently distributed the Tivoli Distributed Monitoring profile
- ENDPOINT - Name of endpoint object. With typical Sentries, this name and OID will be the same as host name and host OID. With Tivoli Distributed Monitoring proxies, this name and OID can be different.
- ENDPOINT_CLASS
- ENDPOINT_OID
- HOST
- HOST_OID
- OPERATOR - Access ID used by administrator for authorization

The above environment variables are available to all types of Tivoli Distributed Monitoring endpoints, managed nodes, SentryProxies, and TMA endpoints.

Refer to the *Tivoli Distributed Monitoring User's Guide* for further details on using environment variables in custom monitor or response scripts.

12.6.2 Distributed Monitoring Proxies

Tivoli Distributed Monitoring endpoints can be managed nodes, TMA endpoints, or proxies. When a Tivoli Distributed Monitor is distributed to a proxy, there is still just a single Tivoli Distributed Monitoring engine on the Tivoli managed node that is updated, but the environment variables associated with the monitor will be different to allow different endpoints to be monitored. For example, custom scripts would be able to have one script use the $ENDPOINT or $HOST environment variables to distinguish between monitored targets.

Note

If a monitor is distributed to both a managed node and a SentryProxy hosted by the same managed node, two copies of the monitor will be active. In case of an asynchronous monitor, both monitor copies will be triggered when the wasync command is run to report an event.

Tivoli Distributed Monitoring proxies also support the creation of additional environment variables that can be used by custom monitors and responses.

12.7 Distributing Monitors

Having completed the profile definition, the time has come to distribute the monitors defined in the profiles to the endpoints on which the specified monitors should run.

Understanding and correctly using profile managers are key elements in the success of your deployment of Distributed Monitoring.

It is very important to understand the concepts of database and dataless Profile Managers as well as the distinction between database mode and dataless mode endpoints. This is described in 2.4.6, “TMA Endpoints” on page 45.

12.7.1 Local Profile Copies

A Tivoli administrator can change the system or user-configuration files distributed in the original profile. However, local overwrites can complicate your deployment. If possible, use a hierarchy that does not require local changes. An administrator can make local changes to only those profiles stored in the Tivoli Configuration Change Management System (CCMS) database. These profiles are:

- Adapter Configuration Facility Profile
- GroupProfile
- SecurityProfile
- SentryProfile
- UserProfile

12.7.1.1 Local Copies of Profiles in Database Profile Managers

Using database profile managers, a copy of each profile (except AutoPack, FilePackage, and InventoryProfile profiles) is stored at each level of the distribution hierarchy. Therefore, after distribution from a database profile manager, each managed node, including the TMR Server, maintains a local copy of all profiles in its client database.

The local profile copy is kept in the local oserv database and is listed along with the original SentryProfiles in a `wlookup -ar SentryProfile` command.

```
wlookup -ar SentryProfile | sort
NT-monitoring   1998892590.1.1433#Sentry::All#
NT-monitoring@ep-sub    1998892590.1.1439#Sentry::All#
NT-monitoring@itso2     1998892590.1.1440#Sentry::All#
TACF Monitors   1998892590.1.1338#Sentry::All#
TivoliSentryDefaults#Guyincharge        1295714281.1.613#Sentry::All#
TivoliSentryDefaults#unixspoke  1998892590.1.614#Sentry::All#
universal       1998892590.1.1441#Sentry::All#
universal@ep-sub        1998892590.1.1442#Sentry::All#
universal@itso2 1998892590.1.1443#Sentry::All#
universal@rh2900b       1998892590.2.69#Sentry::All#
```

The above output from the `wlookup` command shows that local profile copies of the SentryProfile Universal exists on `ep-sub`, `itso2`, and `rh2900b` and local copies of the NT-monitoring SentryProfile are located on `ep-sub` and `itso2`.

> **Note**
>
> Remember, that after modifying a local profile copy, it has to be redistributed in order to become active. This may seem obvious when dealing with profile copies in profile managers, but is less obvious, yet still true, when working with local profile copies owned by managed nodes.

12.7.1.2 Local Profile Copies and Dataless Profile Managers

Using dataless profile managers, the profile information is written directly to the system or application files. Therefore, subscribers of dataless profile managers do not maintain a local copy of the profile. TMA endpoints do not have an object database; so, they cannot have a local copy of a profile. Managed nodes do, indeed, have an object database, but distribution from dataless profile managers bypasses the database and writes directly to the system files.

12.7.2 Distributing Distributed Monitoring Profiles

Having understood the basics of database and dataless Profile Managers and the way they operate when in relation to database and dataless endpoints, managing the distribution of SentryProfiles seems to be a fairly simple and straight-forward task, but the story does not end here.

For each SentryProfile, the default behavior of the distribution is stored along with the profile itself. These values can be modified by the administrator from within the Profile Properties dialog using the **File-Distribution Defaults...** option. The default options are shown in Figure 166:

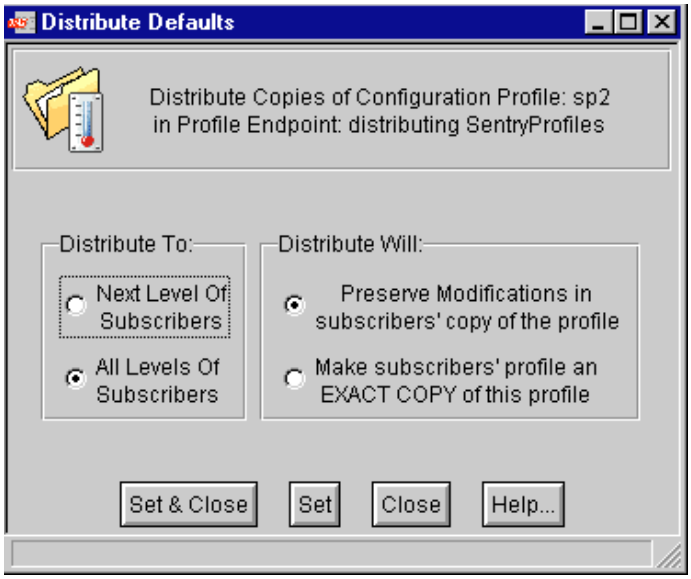


Figure 166. Distributed Monitoring Profile Distribution Defaults

The Distribute To options control the scope of the distribution. The two possible values are:

Next level Copies the profile only to the selected subscribers. In a Tivoli 3.6 environment, distributing to database endpoints, managed nodes, and profile managers sends the profile to the database of the subscriber. For managed nodes, this means that nothing is loaded into the SentryEngine at distribution time. To make the selected SentryProfile active on a managed node instantly, the *All Levels* option should be used.

All Levels Copies the profile to the selected subscribers and their subscribers (if any). This option only has an effect on managed nodes and profile managers subscribing to the profile manager.

It should be noted that since the Distribute To options controls how profiles are being copied down the subscriber-hierarchy made up by profile managers, the Distribute To option has no effect when distributing from a dataless profile manager.

The way the distribution of monitors within a SentryProfile to the endpoints handles local profile copies is controlled by the Distribute Will option. This takes one of two opposite values:

Exact Copy Makes the local profile copy an exact copy of the one distributed and overwrites all local modifications

Preserve Modifications Updates only monitor parameters in the local copy that haven't been changed since the original distribution.

Note

Be aware that the effects of the Distribute Will option, as used with the database profile manager, does apply to the dataless profile managers where local profile copies are not used.

The Distribute Will options will, when used with distributions from a *dataless* profile manager, have the following effects:

Exact Copy Updates the system files on the selected endpoint(s) with the information from all SentryProfiles within the profile manager hosting the profile being distributed.

Preserve Modifications Updates the system files on the selected endpoint(s) with the information from the profile being distributed if, and only if, any monitor within the profile being distributed has been modified compared to the last distribution of the profile to the selected endpoint(s), even if the monitoring probe representing the selected profile at the endpoint(s) has been deleted.

The actions performed when using the Distribute To and Distribute Will options can be summarized, as in Table 16:

Table 16. Actions Taken When Distributing SentryProfiles

	Distributing from Database Profile Manager	Distributing from Dataless Profile Manager
All Levels	Copies the profile to the selected subscribers and all the subsequent subscribers including SentryEngines on managed nodes.	Monitors are loaded into the engine database instantly if they are distributed for the first time or have been modified since the last distribution.

	Distributing from Database Profile Manager	Distributing from Dataless Profile Manager
Next level	Copies the profile to the selected subscribers only.	Monitors are loaded into the engine database instantly if they are distributed for the first time or have been modified since the last distribution.
Exact copy	Makes an exact copy of the profile being distributed in the local database of the endpoint overwriting any modifications that might have been applied to the local profile copy. Distributing a profile with exact copy will force the Sentry Engine to reload the entire database and all monitors.	Overwrites information in the system files with information from all SentryProfiles in the actual Profile Manager being distributed.
Preserve Modifications	Merges the distributed profile with existing records updating only the information that has not been modified locally. No monitors will be loaded into the engine database after distributing with Preserve Modifications.	Distributes only the profile being distributed if, and only if, the profile has been modified. Since no local modifications exists, the resulting monitoring probes are exact copies of the distributed profile.

When distributing a profile, the `push` method is run from the TMR server. When the endpoints have received the distributed monitors, these are merged into the system files and/or local oserv database and the sentry engine database is reloaded using the `engineUpdate` method.

12.7.3 Distributing Profiles Using the GUI

Using the GUI to distribute the SentryProfile, the following should be noted:

- When using drag-and-drop to distribute a single SentryProfile, the default distribution actions, set within the selected profile, are used.
- When distributing a profile manager, all the profiles, including SentryProfiles within that profile manager, will be distributed to all subscribers using the default distribution options for each profile.

12.7.4 Distributing Profiles Using the Command Line

When using the `wdistrib -l` command, the parameters map to GUI parameters as follows:

Table 17. wdistrib Parameters for SentryProfiles

	Exact copy	Preserve Modifications
From database profile manager	overall	maintain
From dataless profile manager	overall_no_merge	over_opts

12.8 The Distributed Monitoring Sentry Engine

On all endpoints where Distributed Monitoring is running the Sentry Engine is the central component. The engine is responsible for firing the individual monitors that have been distributed to the endpoint invoking any response actions, such as running a script, a program, or returning the result to an Indicator Collection, a task, a logfile, a TEC Server, and so on.

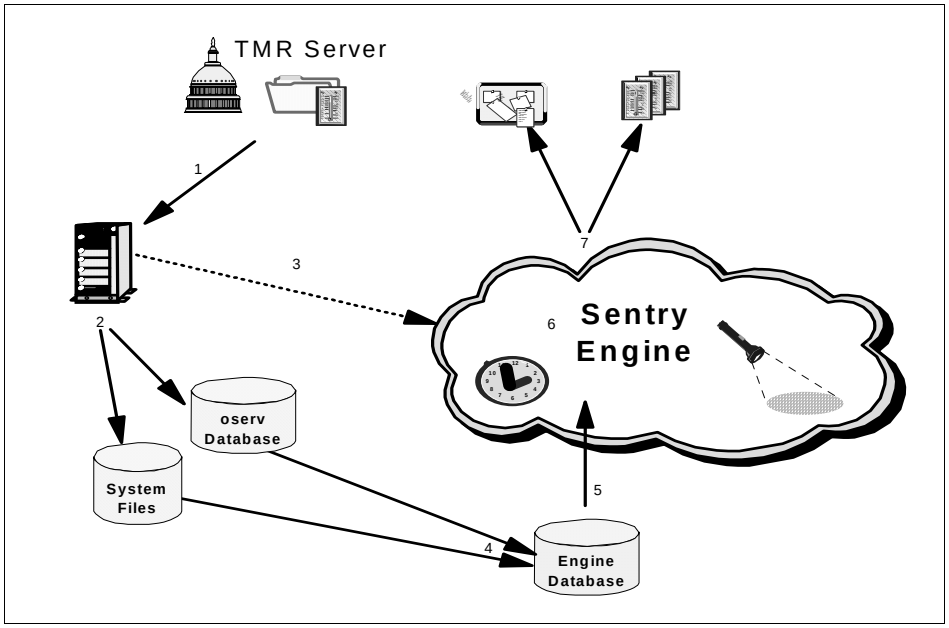


Figure 167. The Sentry Engine

Referring to the diagram in Figure 167, the functions of the managed node SentryEngine are:

1. The endpoint receives the SentryProfile.
2. Information from the SentryProfile is stored in local profile copies or system files.
3. If needed, the SentryEngine is started; otherwise, it is advised that a new SentryProfile has been received.
4. The engine database is reloaded from the local profile copies and system files.
5. The SentryEngine reads the database and builds the ready-list.
6. When the timer indicates that a monitor should be fired, the monitor inspects the required resources and reports back to the SentryEngine.
7. The SentryEngine determines if a response-action should be initiated running a task, updating an Indicator Collection, adding a notice to a NoticeBoard, and initiates the required action.

Working with Tasks

When using tasks as response actions in a monitor that are run on managed nodes, it is vital for the execution of that task that it has been distributed to the managed node on which the monitor is run.

To distribute the task, write the following from the TMR Server:

```
wdisttask -s tasklibrary LOCAL
wdisttask -d tasklibrary taskname
```

Remember, that distribution of tasks only completes successfully if ALL managed nodes in the TMR are running at the time of distribution. The same is true for creation of new tasks in task libraries with a distribute-attribute of LOCAL.

On TMA endpoints, the tasks will be made available through the framework.

The Sentry Engine is started automatically as a boot method when the Tivoli subsystem initiates on an endpoint. For TMR servers and managed nodes, this happens when the oserv program starts. On TMA endpoints, the start of the lcfd program also starts the Sentry Engine. If the Sentry Engine, for some reason, is not running, it will be started when any engine-command (`wlseng`, `wlsprb`, `wclreng`, and others) is executed against the Sentry Engine. This implies that the Engine will also be started if a SentryProfile is distributed to the endpoint, or the endpoint is unsubscribed from a profile manager containing SentryProfiles.

When the Sentry Engine starts, control information describing all the individual monitors that has been distributed to the endpoint is loaded into memory. This is normally referred to as the *engine database*. As long as the Sentry Engine is running, or until the engine database is cleared using the `wclreng` command, the Sentry Engine maintains this control information for each individual monitor adding responses and new firing times.

While running, the Sentry Engine periodically dumps the state of the engine database to the `$DBDIR/.sntcfg` or `$LCF_DATDIR/.sntcfg` directory. This is known as checkpointing, and is primarily performed in order to preserve the responses from previous monitor runs. This enables the Sentry Engine to be able to compare responses with previous runs if relative conditions, such as Increase of 5%, are specified.

The Sentry Engine exits when the engine is stopped, either explicitly by issuing the `wstopeng` command, or by stopping the object dispatcher (oserv).

Dependent on the type of endpoint, there is a slight difference in the way the engine database is instantiated. This is due to the fact that managed nodes hold local profile copies, and TMA endpoints do not. Since the Sentry Engine on a managed node can operate in both database and dataless mode, its engine database is loaded from both the local oserv database and the system files in the `.sntcfg` directory. TMA Endpoints load the engine database just from the `.sntcfg` directory.

Having loaded the engine database with monitor probes, the engine then loads the monitors from the Monitoring Collections as they are needed.

12.9 Troubleshooting Distributed Monitoring

There are a number of things that can be checked if Distributed Monitoring is not working as expected. This section includes some checks and tips for troubleshooting.

Troubleshooting Tivoli Distributed Monitoring falls into two parts:

- Troubleshooting the profile distribution
- Troubleshooting the Sentry Engine

12.9.1 Troubleshooting Distributed Monitoring Profile Distribution

Before troubleshooting the distribution process itself, it might be handy to be able to tell which profiles have been distributed to which subscribers and when. This information can be gathered from database type endpoints for each profile manager using the `wgetsub -l` command.

```
wgetsub -l @sentry
ep-sub  Sentry2.0:   NT-monitoring_1998892590.1.1433#Sentry::All#,
                            last distribute Wed Dec 09 13:18:35 1998

itso2   Sentry2.0:   universal_1998892590.1.1441#Sentry::All#,
                            last distribute Wed Dec 16 12:02:53 1998

itso3   Sentry2.0:
                     universal_1998892590.1.1441#Sentry::All#,
                            last distribute Wed Dec 09 15:03:26 1998

rh2900b Sentry2.0:   universal_1998892590.1.1441#Sentry::All#,
                            last distribute Wed Dec 16 13:45:50 1998
                     NT-monitoring_1998892590.1.1433#Sentry::All#,
                            last distribute Wed Dec 16 12:07:14 1998
```

Figure 168. Using wgetsub for Profile Manager with SentryProfiles

The information in Figure 168 shows that the profile `universal` was distributed to endpoints `itso2`, `itso3`, and `rh2900b`, and the profile `NT-monitoring` was distributed to `ep-sub` and `rh2900b`.

There is no way to gather the same information from dataless endpoints, but for all endpoints, the `wlseng` and `wlsprb` commands show which monitors are active in the SentryEngine.

In order to pinpoint problems related to profile distribution, the finer details of the distribution process will have to be fully understood. This is the subject of the next topic.

12.9.1.1 How SentryProfile Distributions Work

Distribution of a profile manager for Distributed Monitoring takes place in two phases:

1. Using MDist (a framework service), the profile is sent from the TMR Server to the dogEndpoint program on the endpoint using an IOM channel. If the endpoint has a local database, such as managed nodes, the local database is updated with the new profile.
2. A second IOM channel is opened between the dogEndpoint process and the local sentry or endpoint engine process, and data is fed to the engine.

After the engine receives the data, it updates the files in the `$DBDIR/.sntcfg` or `$LCF_DATDIR/.sntcfg` directory.

Refer to 7.6.3, “Bulk Data Transfer and Inter-Object Messaging” on page 271 for details on how IOM sessions are initiated and controlled.

If other applications, such as software distribution, work fine to the endpoint, the first phase of the transfer should also work for distributed monitoring. In case of a problem, it is the second phase with errors when the endpoint is attempting to connect back to itself.

If you repeat what the IOM transfer is attempting to do, you will generally find the problem. The `hostname` command will find the same name it's looking at then resolve this hostname to an IP address. See if you can ping the address. The problem is normally hostname resolution on the endpoint; it cannot resolve its own name, or it cannot talk to itself through the loopback adapter. If loopback is down, the transfer will fail.

12.9.1.2 Troubleshooting Distributions

Troubleshooting distribution of SentryProfiles is very similar to that of other profiles, so the output from the `odstat` and `wtrace` commands are the basic sources of information. The troubleshooting involves three basic steps:

1. Verifying IP connectivity and name resolution - including local name lookup
2. Tracing the IOM sessions - external as well as internal
3. Tracing attributes

When setting up tracing with `odadmin trace`, it is essential that both `objcalls` and `services` are specified and that the order is `objcalls` first and then `services`. The services trace parameter enables tracing of manipulation of object-attributes and is vital for gathering complete trace information for distribution of SentryProfiles.

```
odadmin trace objcalls
odadmin trace services
```

In a Tivoli environment using TMA endpoints, it is important to remember that there are more than two players active during distribution. The gateway plays a very active role in the process, and what's happened on the gateway is just as important as what's going on at the TMR server and at the TMA endpoint. However, this is only true when distributing to TMA endpoints. Dataless endpoints on managed nodes do not use the gateway, and, therefore, it does not make sense to trace the gateway when experiencing distribution problems on a managed node.

Trace Levels

When tracing the gateway, a debugging level of `9` should be used. Use the `wgateway` command to set the debug level and remember to restart the gateway to activate the changes:

```
wgateway <gateway set_debug_level 9
wgateway rh2900b-gateway restart
```

On endpoints, the contents of the /tmp/dm36.ll file should be `4` to indicate full debugging of the Sentry components. To accomplish this, the following command can be used:

```
echo 4 > /tmp/dm36.ll
```

When troubleshooting distributions, always look for the push method in the `odstat` output from the TMR server or managed node hosting the profile. Then, follow the transaction hierarchy looking for errors or peculiarities such as `NO_METHOD`.

The example in Figure 169 on page 414 shows an `odstat` output where it is attempting to distribute a Sentry profile to a TMA endpoint after having stopped the gateway.

```
* 4692 O+hdoq 1-4509  done   935     0 17:47:29  e=12  1998892590.1.517#TMF_LCF::EpMgr# push_copy_in
  4693 O+    1-4692  done    15     0 17:47:29        0.0.0 get_name_registry
  4694 O+hdoq 1-4692  done    97     0 17:47:29        1998892590.1.26 lookup
  4695 O+hdq 1-4692  done    46     0 17:47:29        1998892590.1.4 lookup_id
  4696 O+hdq 1-4692  done   148     0 17:47:29        1998892590.1.4##2@Sentry::All describe
  4697 O+hdq 1-4692  done  2136     0 17:47:29        1998892590.1.4##2@Sentry::All _get_type
  4698 O+    1-4692  done    60     0 17:47:30        1998892590.1.1162#Sentry::All# _get_profile_organizer
  4699 O+hdoq 1-4692  done   111     0 17:47:30        1998892590.1.26 lookup
  4700 O+hdoqs 1-4692  done   577     0 17:47:30        1998892590.1.517#TMF_LCF::EpMgr# get_endpoint_key_value
  4701 O+    1-4692  done    25     0 17:47:30        1998892590.1.1162#Sentry::All# _get_label
  4702 O+hdoqs 1-4692  done     6     0 17:47:30        1998892590.1.955#TMF_UI::Extd_Desktop# execute
  4703 O+hdoq 1-4692  done   551     0 17:47:30        1998892590.1.1158#TMF_CCMS::ProfileManager# default_push_profiles
  4704 O+ho  1-4703  done   461     0 17:47:30        1998892590.1.1162#Sentry::All# default_push
  4705 O+hdq 1-4704 done 461 0 17:47:30  1998892590.1.1162#Sentry::All# push
  4706 O+hdoq 1-4705  done   219     0 17:47:31        1998892590.1.1158#TMF_CCMS::ProfileManager# get_record
  4707 O+    1-4705  done    15     0 17:47:31        0.0.0 get_name_registry
  4708 O+hdoq 1-4705  done    88     0 17:47:31        1998892590.1.26 lookup
  4709 O+    1-4705  done    81     0 17:47:31        1998892590.0.0 get_identity
  4710 O+    1-4705  done    31     0 17:47:31        1998892590.1.179#TMF_Administrator::Configuration_GUI# _get_label
  4711 O+hdoq 1-4705  done  1128     0 17:47:31        1998892590.1.1158#TMF_CCMS::ProfileManager# get_record
  4712 O+hdoq 1-4705  done     6     0 17:47:32        1998892590.1.1158#TMF_CCMS::ProfileManager# set_records
  4713 O+ho  1-4705 done 461 0 17:47:32  redirect push
  4714 O+hdoq 1-4713  done  2054     0 17:47:32        1998892590.1.1158#TMF_CCMS::ProfileManager# get_record
  4715 O+hdoq 1-4713 done 461 0 17:47:32  1998892590.1.1158#TMF_CCMS::ProfileManager#
push_with_actions
* 4716 O+    1-4715  done   698     0 17:47:32        1998892590.1.1158#TMF_CCMS::ProfileManager# _get_subscribers
  4717 O+    1-4715  done    15     0 17:47:32        0.0.0 get_security_objid
  4718 O+    1-4715  done    29     0 17:47:32        1998892590.0.0 idmap_list_maps
  4719 O+    1-4715  done    57     0 17:47:32        1998892590.0.0 idmap_list_entries root_group
  4720 O+    1-4715  done   102     0 17:47:32        1998892590.0.0 idmap_list_entries root_user
  4721 O+    1-4715  done    15     0 17:47:33        0.0.0 get_name_registry
  4722 O+hdoq 1-4715  done    95     0 17:47:33        1998892590.1.26 lookup
  4723 O+hdoq 1-4715  done   236     0 17:47:33        1998892590.1.366 obj_route
  4724 O+    1-4723  done    15     0 17:47:33        0.0.0 get_name_registry
  4725 O+hdoq 1-4723  done   690     0 17:47:33        1998892590.1.26 get_all
  4726 O+hdoq 1-4723  done   237     0 17:47:34        1998892590.1.26 get_all
  4727 O     1-4723  done    18     0 17:47:34        1295714281.1.517 get_rpt_format
  4728 O+hdoqs 1-4723  done   168     0 17:47:34        1998892590.1.517 get_rpt_format
  4729 O+hdoq 1-4728  done   260     0 17:47:34        1998892590.1.26 region_get_all
  4730 O+hdoq 1-4723  done   690     0 17:47:34        1998892590.1.26 get_all
  4731 O+hdoq 1-4723  done   237     0 17:47:34        1998892590.1.26 get_all
  4732 O+hdoqs 1-4723  done   168     0 17:47:34        1998892590.1.517 get_rpt_format
  4733 O+hdoq 1-4732  done   260     0 17:47:35        1998892590.1.26 region_get_all
  4734 O+hdoq 1-4715  done    95     0 17:47:35        1998892590.1.26 lookup
4735 O+hdoq 1-4715  done    12     0 17:47:35       1998892590.1.366 _get_final_timeout
* 4736 O a 1-4715 done 0 0 17:47:35 NO_METHOD 1998892590.2.21#TMF_Gateway::Gateway# rpt
* 4737 O   1-4715 done 0 0 17:47:38 NO_METHOD 1998892590.2.21#TMF_Gateway::Gateway#
fps_cancel
  4738 O+    1-4713  done    25     0 17:47:38        1998892590.1.1158#TMF_CCMS::ProfileManager# _get_label
  4739 O+    1-4713  done    15     0 17:47:38        0.0.0 get_name_registry
  4740 O+hdoq 1-4713  done   128     0 17:47:38        1998892590.1.26 lookup
  4741 O+bhdq 1-4713  done     6     0 17:47:38        1998892590.1.88#TMF_SysAdmin::InstanceManager# log_notice
  4742 O+hdoqs 1-4692  done     6     0 17:47:39        1998892590.1.955#TMF_UI::Extd_Desktop# execute
  4743 O+hdq 1-4509  done  8719     0 17:47:39        1998892590.1.75#TMF_Message::Catalog# get_message_catalog
  4744 O+hdq 1-4509  done  2639     0 17:47:39        1998892590.1.75#TMF_Message::Catalog# get_message_catalog
```

Figure 169. Output of odstat from SentryProfile Distribution

Prior to the push method, a number of methods concerned with the GUI interactions used for distributing are run. The push itself is run in thread-ID (TID) 4705. It launches a number of sub-processes to ensure that all pre-requisites are available before performing the distribution.

In this case, the interesting sub-process hierarchy is 4113-4715-4736/4737, which ends with a NO_METHOD condition indicating that the rpt method on the gateway at dispatcher 2 (the full oid is 1998892590.2.21) could not be found.

Further investigation will now have to be performed on the managed node hosting the 1998892590.2.21 gateway.

The next example in Figure 170 shows the methods involved on the gateway when the SentryEngine on a TMA initiates. The SentryEngine had deliberately been stopped (using the wstopeng command) and was then forced to restart using the wlseng command. The odstat output shows the three basic methods used by the wlseng command at the endpoint (1998892590.22.508+) which are:

tmr_examine	Inspects the engine database for timer data
mp_examine	Inspects the engine database for monitor probe data
ra_examine	Inspects the engine database for response data

```
 3328 O             done   296     0 10:58:26        1998892590.1.4##6@LCFData::ep_tnr_info_s describe
  3329 O              done   115     0 10:58:26        1998892590.1.26 lookup
  3330 O              done 18178     0 10:58:26        1998892590.1.75#TMF_Message::Catalog# get_message_catalog
  3331 O              done  1146     0 10:58:26        1998892590.22.508+#TMF_Endpoint::Endpoint# tmr_examine
---------- records removed
  3337 M hdq          done  4790     0 10:58:55        1998892590.2.581- GetCollList
  3338 O+    2-3337  done    15     0 10:58:56        0.0.0 get_name_registry
  3339 O     2-3337  done   109     0 10:58:56        1998892590.1.26 lookup
  3340 O     2-3337  done    64     0 10:58:56        1998892590.1.14#TMF_SysAdmin::Library# lookup_object
  3341 O     2-3337  done    33     0 10:58:56        1998892590.1.540#TMF_SysAdmin::InstanceManager# get_prototype
  3342 O     2-3337  done   111     0 10:58:56        1998892590.1.26 lookup
  3343 O     2-3337  done   188     0 10:58:56        1998892590.1.26 lookup
  3344 O     2-3337  done    97     0 10:58:56        1998892590.1.26 lookup
  3345 O     2-3337  done    57     0 10:58:56        1998892590.1.4 lookup_id
  3346 O     2-3337  done   296     0 10:58:56        1998892590.1.4##6@LCFData::ep_tnr_info_s describe
  3347 O     2-3337  done   111     0 10:58:56        1998892590.1.26 lookup
* 3348 O     2-3337  done   373     0 10:58:56  e=12  1998892590.1.517#TMF_LCF::EpMgr# add_boot_method
  3349 O     2-3337  done  4811     0 10:58:57        1998892590.1.26 region_get_all
  3350 O              done  1327     0 10:58:58        1998892590.22.508+#TMF_Endpoint::Endpoint# mp_examine
  3351 M hdq          done 54077     0 10:59:01        1998892590.2.581- GetColl
  3352 O     2-3351  done   134     0 10:59:01        1998892590.1.26 local_lookup
  3353 O     2-3351  done 32253     0 10:59:01        1998892590.1.834#SentryMonitoringCapability::Collection#
_get_collection
  3354 M hdq          done 20259     0 10:59:03        1998892590.2.581- GetColl
  3355 O     2-3354  done   126     0 10:59:03        1998892590.1.26 local_lookup
  3356 O     2-3354  done 13877     0 10:59:03        1998892590.1.822#SentryMonitoringCapability::Collection#
_get_collection
* 3357 M hdq          done     0     0 10:59:04   e=6  1998892590.2.581- GetTask
  3358 O+    2-3357  done    45     0 10:59:04        0.0.0 get_host_location
  3359 O+hdoq 2-3357  done    30     0 10:59:04        1998892590.2.7#TMF_ManagedNode::Managed_Node# install_directory
  3360 O+    2-3359  done    15     0 10:59:05        0.0.0 get_oserv
  3361 O+    2-3359  done    13     0 10:59:05        1998892590.2.2 query install_dir
  3362 O              done  1329     0 10:59:06        1998892590.22.508+#TMF_Endpoint::Endpoint# ra_examine
  3363 M hdq          done 150431     0 10:59:07        1998892590.2.581- GetColl
  3364 O+    2-3363  done    15     0 10:59:07        0.0.0 get_name_registry
  3365 O     2-3363  done  4811     0 10:59:07        1998892590.1.26 region_get_all
  3366 O     2-3363  done   129     0 10:59:07        1998892590.1.26 local_lookup
  3367 O     2-3363  done 116856     0 10:59:07        1998892590.1.858#SentryMonitoringCapability::Collection#
_get_collection
  3368 M hdq          done    19     0 10:59:11        1998892590.2.581- ChangeIcon
  3369 O     2-3368  done  4811     0 10:59:11        1998892590.1.26 region_get_all
  3370 O     2-3368  done 25530     0 10:59:11        1998892590.1.816#SentryMonitoringCapability::Collection#
_get_collection
-------- a number of _get_collection entries removed
  3407 O     2-3368  done 41478     0 10:59:27        1998892590.1.851#SentryMonitoringCapability::Collection#
_get_collection
* 3408 O+bhdq 2-3368  done     0     0 10:59:28   e=6  1998892590.2.581- QueueConsumer
```

Figure 170. Output from odstat Showing Restart of Sentry Engine

12.9.2 Troubleshooting Monitor Execution

Monitors are distributed from the server and run on each individual system. In debugging Tivoli Distributed Monitoring problems, there are many pieces of information required to narrow down a problem.

For managed nodes, the very first place to look is in the Notice Board. By default, all failing monitor runs report their status to the SentryStatus group of the Notice Board. Remember to add this group to the Notice Board Subscriptions for each of the administrators that will handle Distributed Monitoring.

.

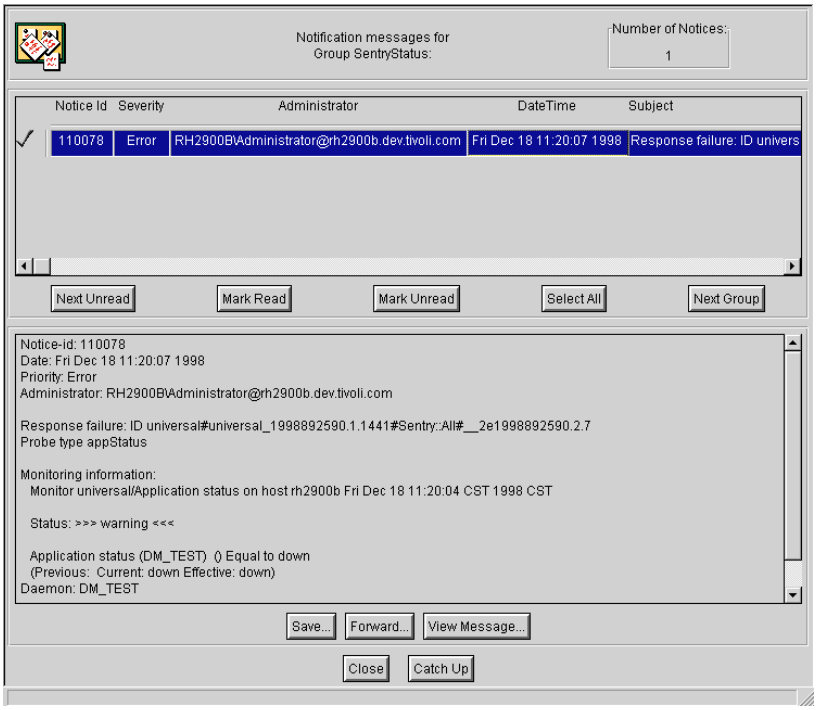


Figure 171. Response Failure Reported in the SentryStatus Notice Group

The notice in Figure 171 shows a notice from a monitor monitoring the state of application DM_TEST. The detailed information tells us that problems were encountered when issuing the response with the warning severity.

Adding notices to the Notice Board when problems occur in firing responses is only implemented for managed nodes - not for TMA endpoints. The reason for this is that the Notice Board and the oserv process of the TMR Server would be overloaded if the same problem occurred on thousands of

endpoints simultaneously, which could happen if a monitor generating the problems is distributed to the endpoints.

Tivoli Distributed Monitoring supplies a number of commands that can be used to get detailed information on the SentryEngine itself, the different monitoring probes running in the engine, and commands to manipulate monitors and monitoring probes.

In addition, from Tivoli Distributed Monitoring Version 3.6, the Sentry Engine on a TMA endpoint maintains a logfile, `$LCF_DATDIR\dm36.log`, which shows the activity of the engine. The level of detail is controlled by the file /tmp/dm36.ll.This file holds only a number indicating the log level similar to the log level parameter of the lcfd engine itself. The log levels are:

0 No information
1 Minimal logging
2 Tracing and moderate output
3 Detailed information and tight loops
4 Data

Level 4 generates a large amount of data and should only be used in certain cases. For general troubleshooting, level 2 or 3 is recommended. Figure 172 on page 418 is an example of a file generated with log level 3 for the initialization of the SentryEngine of a TMA endpoint:

```
Dec 09 15:37:34 1 DM3.6 Sentry Engine Log
Dec 09 15:37:34 2 DM3.6 Initialized private marshaling code
Dec 09 15:37:34 2 DM3.6 Initialized data store
Dec 09 15:37:34 2 DM3.6 Loading collection list...
Dec 09 15:37:34 2 DM3.6 Connecting to '127.0.0.1+9494'
Dec 09 15:37:34 2 DM3.6 Loaded capabilities
Dec 09 15:37:34 2 DM3.6 Beginning the engine run loop previous time =913239420
Dec 09 15:37:34 2 DM3.6 Beginning the engine run loop current minute =913239420
Dec 09 15:37:34 2 DM3.6 ready list is empty!
Dec 09 15:37:34 2 DM3.6 call to wait for connection seconds =26
Dec 09 15:37:35 2 DM3.6 recieved message of =32
Dec 09 15:37:35 2 DM3.6 message code is 23
Dec 09 15:37:35 2 DM3.6 returned from connection thread call
Dec 09 15:37:35 2 DM3.6 Beginning the engine run loop current minute =913239420
Dec 09 15:37:35 2 DM3.6 ready list is empty!
Dec 09 15:37:35 2 DM3.6 call to wait for connection seconds =25
Dec 09 15:37:35 2 DM3.6 recieved message of =31
Dec 09 15:37:35 2 DM3.6 message code is 24
Dec 09 15:37:35 2 DM3.6 returned from connection thread call
Dec 09 15:37:35 2 DM3.6 Beginning the engine run loop current minute =913239420
Dec 09 15:37:35 2 DM3.6 ready list is empty!
Dec 09 15:37:35 2 DM3.6 call to wait for connection seconds =25
Dec 09 15:37:35 2 DM3.6 recieved message of =31
Dec 09 15:37:35 2 DM3.6 message code is 25
Dec 09 15:37:35 2 DM3.6 returned from connection thread call
Dec 09 15:37:35 2 DM3.6 Beginning the engine run loop current minute =913239420
Dec 09 15:37:35 2 DM3.6 ready list is empty!
Dec 09 15:37:35 2 DM3.6 call to wait for connection seconds =25
Dec 09 15:38:00 2 DM3.6 connect exception: Timeout after 0 secs.
Dec 09 15:38:00 2 DM3.6 returned from connection thread call
Dec 09 15:38:00 2 DM3.6 Beginning the engine run loop current minute =913239480
Dec 09 15:38:00 2 DM3.6 Adding probe 0 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 1 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 2 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 3 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 4 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 5 to the ready list
Dec 09 15:38:00 2 DM3.6 Adding probe 6 to the ready list
```

Figure 172. The dm36.log File - SentryEngine Startup (Log Level 3)

As was shown in Figure 172, first the engine database is initialized, then the Collections are loaded, and then the engine starts the loop checking for monitors on the ready list.

Figure 173 on page 419 shows an extract from the dm36.log firing a monitor probe, `Task 2`, and checking for responses. In this case, the response action was to update an icon in an IndicatorCollection, which had been removed. Therefore, the response `Upcall (Icon)` failed.

```
Dec 09 15:38:02 2 DM3.6 Beginning the engine run loop current minute =913239480
Dec 09 15:38:02 2 DM3.6 running a probe from the ready list
Dec 09 15:38:02 2 DM3.6 Task 6
(<NT_System><SysUpTime><p3#p3_1351550138.1.1159#Sentry::All#__2e1351550138.119.0+>)
NOT ready
Dec 09 15:38:02 2 DM3.6 Task 5
(<NT_System><FileRdOperPerSec><p4#p4_1351550138.1.1160#Sentry::All#__0e1351550138.119
.0+>)
NOT ready
Dec 09 15:38:02 2 DM3.6 Task 4
(<NT_IP><GramsPerSec><p1#p1_1351550138.1.1157#Sentry::All#__0e1351550138.119.0+>)
NOT ready
Dec 09 15:38:02 2 DM3.6 Task 3
(<NT_EventLog><Secevent><p2#p2_1351550138.1.1158#Sentry::All#__1e1351550138.119.0+>)
NOT ready
Dec 09 15:38:02 2 DM3.6 Task 2
(<NT_EventLog><Syswarnevent><p2#p2_1351550138.1.1158#Sentry::All#__0e1351550138.119.0
+>)
ready
Dec 09 15:38:02 2 DM3.6 probe 2 ran, checking for responses
Dec 09 15:38:02 2 DM3.6 Connecting to '127.0.0.1+9494'
Dec 09 15:38:03 1 DM3.6 Upcall (Icon) failure: Wed Dec  9 15:48:52 CST 1998 (2):
operation 'requested resource not found' failed
Dec 09 15:38:03 2 DM3.6 Task 1
(<TME_Monitors><ObjCallsmade><p3#p3_1351550138.1.1159#Sentry::All#__1e1351550138.119.
0+>)
NOT ready
Dec 09 15:38:03 2 DM3.6 Task 0
(<Universal><appStatus><p3#p3_1351550138.1.1159#Sentry::All#__0e1351550138.119.0+>)
NOT ready
Dec 09 15:38:03 2 DM3.6 call to wait for connection seconds =0
Dec 09 15:38:04 2 DM3.6 connect exception: Timeout after -1 secs.
Dec 09 15:38:04 2 DM3.6 returned from connection thread call
```

Figure 173. The dm36.log File - SentryEngine Running (Log Level 3)

If the information available in the dm36.log is not enough to pin-point the problem, the following has to be checked, especially when monitoring through custom scripts or using response scripts, prior to any further investigation:

- Ensure any custom monitor script contains `#!/bin/sh` at the start and `exit 0` at the end.
- Check the file permissions for the program in use for correct access rights.
- Check the user name and group name being used.
- Try setting the response level to **always** and **log to file**. Then you can review the log file.

12.9.3 Monitoring Command Overview

The following commands can be used to determine how Distributed Monitoring is set up and running:

- On the TMR Server:

  `wlsmon` Lists the monitors for a SentryProfile providing a view of the top level monitor on the server's database.

  `wdumpsnt` Exports the monitor definition to be saved to a file and is used to create another identical monitor.

  `wloadsnt` Imports the monitor definition.

  `wsetmon` (`+` or `-d`) Enables or disables a monitor.

- On Sentry engines, commands can be run on any managed node with a parameter to specify the desired node:

  `wclreng` Clears the Sentry engine on the client specified. Useful if a monitor was inadvertently defined to run too often.

  `wdelprb` Deletes a specified Sentry monitor from a client engine (*not* available on TMA endpoints).

  `wrunprb` Runs a specified Sentry monitor from a client engine.

  `wlseng` Lists the contents of the Sentry engine.

Usage Note

When running the `wclreng`, `wrunprb`, or `wlseng` commands on a TMA endpoint, it is necessary to source-in the proper environment.

This can be achieved from the command line with the following command:

UNIX: `$LCF_DATDIR/dm_env.sh`

Intel: `%LCF_DATDIR%\dm_env.cmd`

When running the commands from a managed node, use the `-z` flag to specify that the host referenced in the host argument of the command is a TMA endpoint.

12.9.4 The wlseng Command

The `wlseng` command provides very detailed output of what is happening in a Sentry engine.

There are three main sections of output from the `wlseng` command:

Timer Lists timing information about when Sentry will start and how often it will run.

Monitoring probes Lists the values for each severity level and the current value of the monitor.

Responses Lists what action(s) will take place for each severity level of a monitor.

The `wlseng` command, without any arguments, outputs information from the endpoint where the command was invoked. The command with the name of an endpoint will get information from endpoint and the `-l` option will give a long listing.

12.9.4.1 wlseng Examples

The example in Figure 174 on page 422 is the output from a monitor with all severity levels going off every two hours on week days. In the first response, you can see an entry for severity level marked asterisk (*). This means always, and in this case, an entry will be written to a file. You will also see that this monitor has two responses for critical severity, the second of which is how a task is listed. Note that the emphasis on some of the words is ours and is not normally in the output.

```
 Timer:
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> 2:H:Y8,17:Y18,7:Y1,5:N6,0:N0,23:N0,6:Custom hours:
Custom days:861830400
      Wakeup: 04/23/97 18:20:00 (112 minutes from now)
Monitoring probes:
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> ("@U_Sentry_2hr_weekday","@G_Sentry_2hr_weekday",
"Sentry_2hr_weekday1980366669.1.327")diskusedpct("/") > 95 : "critical", > 90 :
"severe", > 80 : "warning", : "normal";
      Last value: <none>
Responses:
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> <<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sen
try_1980366669.1.637#Sentry::All#__1e1980366669.1.327>> *
   file("@U_Sentry_2hr_weekday","@G_Sentry_2hr_weekday","0","/var/adm/DiskUsage",
"38x1980366669.1.539#SentryEngine::engine#snapon");

<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>>
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> normal
    icon("@K_Sentry_2hr_weekday");

<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>>
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> warning
    popup("0","54x1980366669.1.179#TMF_Administrator::Configuration_GUI#Root_snapo
n31-region");

<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>>
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> severe
    mail("0","mhahn@tivoli.com"),notify("0","Sentry-urgent"),tec("29x1980366669.1.5
97#Tec::Server#EventServer","CRITICAL","0");

<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>>
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> critical
    popup("0","54x1980366669.1.179#TMF_Administrator::Configuration_GUI#Root_snapo
n31-region"),notify("0","Sentry-urgent"),tec("29x1980366669.1.597#Tec::Server#Event
Server","FATAL","0");

<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>>
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_1980366669.1.637#Sentry::
All#__1e1980366669.1.327>> critical
    [ Tasks: snapon_tasks:CleanRoot ]
```

Figure 174. Output from wlseng for All Severity Levels Every Two Days

The next example in Figure 175 on page 423 lists output from a monitor that has custom hours set:

```
Timer:

 <<Unix_Sentry><daemonct><Sentry_custom_hours#Sentry_custom_hours_1980366669.1.638#Se
 ntry::All#__0e1980366669.1.327>> 20:m:N8,17:N17,8:Y1,5:N6,0:Y17,6:N0,6:Custom
hours:Custom days:861822900
      Wakeup: 04/23/97 17:15:00 (156 minutes from now)

Monitoring probes:
 <<Unix_Sentry><daemonct><Sentry_custom_hours#Sentry_custom_hours_1980366669.1.638#Se
ntry::All#__0e1980366669.1.327>>
 ("@U_Sentry_custom_hours","@G_Sentry_custom_hours","Sentry_custom_hours1980366669.1.
 327")daemonct("doom") > 1 : "severe", : "normal";
     Last value: <none>

Responses:
 <<Unix_Sentry><daemonct><Sentry_custom_hours#Sentry_custom_hours_1980366669.1.638#Se
ntry::All#__0e1980366669.1.327>>
 <<Unix_Sentry><daemonct><Sentry_custom_hours#Sentry_custom_hours_1980366669.1.638#Se
ntry::All#__0e1980366669.1.327>> severe
    mail("0","big_cheese_admin@tivoli.com");
```

Figure 175. Output from wlseng with Custom Hours Set

The intention of this configuration was that anyone playing Doom during working hours will cause mail to be sent to the `big_cheese_admin`. Actually, this example shows a common mistake. This monitor will have to find two copies of Doom running (`>1`) in order for the *severe* level to be triggered.

The next section provides more detail on interpreting the output from `wlseng`.

12.10 Interpreting Sentry Engine Information

Using the `wlseng` command (see “wlseng Examples” on page 421) we can find out everything about a monitor that Distributed Monitoring is using. The three sections of the output are described in more detail here.

12.10.1 Determining Monitor Timing

With the output from `wlseng`, we can read information about when the monitor will next run, the intervals, and so on. The Timer section of the `wlseng` output is made up of the identification, how often the monitor will go off (scheduling), and when the monitor will next go off.

The identification area looks like this:

```
<<Unix_Sentry><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_234907.1.
532#Sentry::All#__0e234907.1.332>>
```

- The first field is the name of the Sentry monitor collection.
- Field two is the name of the monitor.

- Field three is the full name of the SentryProfile. The OIDs listed are for the SentryProfile and the node running the monitor, respectively.

The schedule piece looks something like this:

```
10:m:Y8,17:Y18,7:Y1,5:Y6,0:N0,23:N0,6:Custom hours:Custom
days:805477500
```

This is broken up as follows:

`10:m`	Interval - This example runs every 10 minutes.
`Y8,17`	Yes, runs between 8:00 - 17:00 (Standard hours).
`Y18,7`	Yes, runs between 18:00 - 7:00 (Standard hours).
`Y1,5`	Yes, runs Monday through Friday.
`Y6,0`	Yes, runs Saturday and Sunday.
`NO,23`	No, custom hours not set.
`NO,6`	No, custom days not set.
`Custom hours`	Defines custom hours.
`Custom days`	Defines custom days.
`805477500`	Initial startup time. This is the number of seconds since midnight (GMT) on January 1st, 1970. Newer releases show the converted time in `wlseng` output. Figure 176 is a C source example that can be used to convert this time:

```
#include <stdio.h>
#include <time.h>
main(int argc, char **argv)
{
long t = atoi(argv[1]);
struct tm lt = *localtime(&t);
struct tm gt = *gmtime(&t);
printf("Local: %s", asctime(&lt));
printf("Universal: %s", asctime(&gt));
{

run as follows:
convert_time 805477500
```

Figure 176. C Source to Convert Monitor Startup Time Value

Notes

Tivoli uses the digit `0` to represent Sunday - A user may enter `7`, but it will be stored as `0`.

The initial startup time and the interval are always used to determine when the monitor will next run. It will not run immediately after a Sentry engine is restarted. (A monitor can be run immediately using `wrunprb`.)

- The next occurrence section will look like:

  ```
  Wakeup: 01/27/96 13:35:00 (9 minutes from now)
  ```

The `wrunprb` command can be used to execute a monitor immediately that is on an engine. The syntax is:

```
wrunprb <collection_name> <monitor_name> <sentry_profile_name>
```

12.10.1.1 When Monitors Do Not Run when Expected

When a monitor is created, one of the parameters it is given is an initial start time. This is under the Set Monitoring Schedule window where it shows the `Start monitoring activity`. This is shown in Figure 177:

:

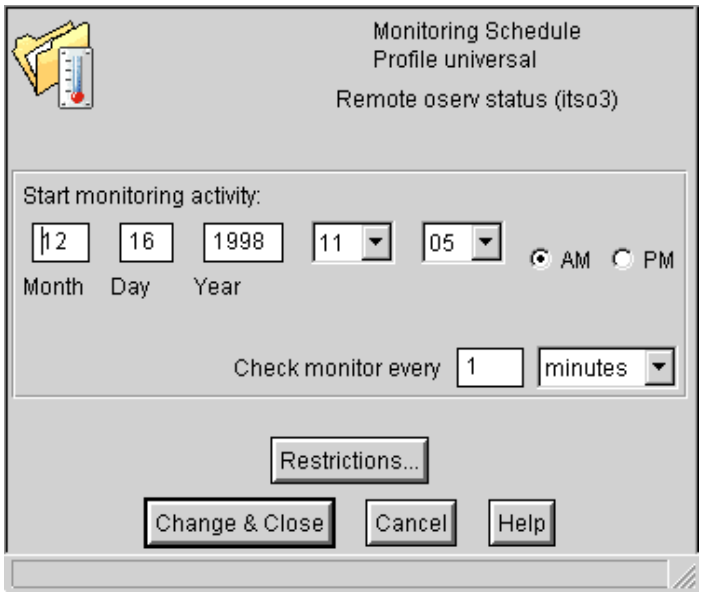


Figure 177. Distributed Monitoring - Start Monitoring Activity

The monitor is given a start date including:

- Month
- Day
- Year
- Hour
- Minute
- Am/Pm

This generates an absolute count of the number of seconds from midnight (GMT) on January 1st, 1970 that this monitor is supposed to start at.

The time is based upon the LOCAL setting on the TMR Server (the TZ setting shown in `odadmin environ set`).

When this monitor is distributed, the `wlseng -l` command shows in the Timer section three pieces:

- The absolute time to first fire
- The translated time (in local time)
- The delta or offset from the current local time

```
Timer:
  <<Universal><scustom>........,6:Custom hours:Custom days:889053600
      Wakeup: 03/04/98 17:20:00 (353 minutes from now)
```

After the `Custom days` section, a number is shown. This is the absolute count in seconds when the monitor should first fire. This number is translated to GMT (also called coordinated universal time) and further converted using the LOCAL setting of the oserv TZ variable (`odadmin environ get`) to produce the Waketime: In this example, `03/04/98 17:20:00`.

The `wlseng` command then does a system call to the localtime() function to return the system timezone adjusted time and uses this time to calculate the `(n minutes from now)` part of the entry.

So, how does this effect when the monitor will really fire?

On the machine running the monitor, if the TZ variable the oserv uses is set to the same timezone as the local timezone setting, or is not set (the default) the probe will fire at the same time in all timezones. This is not the same local time, that is 5pm, but the same absolute time. If the probe was set to fire at 6pm US Central time, it will fire at 7pm US Eastern, 5pm US Mountain, and 4pm US Pacific time. *This may not be the expected behavior!*

If the TZ variable the oserv uses is set to the same timezone for ALL machines, that is CST6CDT, then the monitor will fire at the same LOCAL

time, 5pm in all timezones. Setting the TZ variable to other than the local timezone affects the entire oserv environment. This will cause the Tivoli scheduler to run processes at times different from the expected times. Do this with *EXTREME CAUTION*.

By default, the TZ variable may not be set for a specific platform. This can cause the monitors to fire at unexpected times. Always make sure the oserv environment variable is set so you know when the monitors will fire. To check the settings do:

```
odadmin environ get
```

to modify:

```
odadmin environ get >filename
edit filename and change/add the TZ variable to the set.
```

Example:

```
# odadmin environ get
PATH=/bin:/usr/bin
SHLIB_PATH=
TZ=CST6CDT  <-----------------------------------------------
NLSPATH=/opt/Tivoli/msg_cat/%L/%N.cat
```

then do:

```
odadmin environ set <filename
odadmin reexec <disp#> from the TMR Server or stop/start the oserv on the
machine.
```

Now all of the monitors should be firing at predictable times.

12.10.2 Understanding Monitoring Probe Information

With the output from `wlseng`, we can interpret information about the monitor probes themselves. The Monitoring Probes section of the output contains identification, severity level, and last probed value information.

- The identification section looks like this:

  ```
  <<Sentry2.0><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_234907.1.
  531#Sentry::All#__0e234907.1.332>>
  ```

  This is the same identification as used in the Timer part of the output and is broken down as follows:

  - The first field is the name of the Sentry monitor collection.
  - Field two is the name of the monitor.

- Field three is the full name of the Sentry profile.

- The next line defines the severity levels that will equate with the values returned from the probe:

  ```
  ("@U_Sentry_2hr_weekday","@G_Sentry_2hr_weekday","Sentry_2hr_weekday
  234907.1.332")diskusedpct("/") > 95 : "critical", > 90 : "severe", >
  80 : "warning", : "normal";
  ```

  In this case, if the disk used percentage count rose above 95, this would be indicated as critical, above 90 up to 95 would be severe, above 80 up to 90 would be warning, and other values would be considered normal.

- The last probed value is listed next. In our case, this was:

  ```
  Last value: <none>
  ```

12.10.3 Understanding Monitoring Response Information

With the output from `wlseng`, we can interpret information about the monitor response configuration. The Responses section contains an entry for each severity level defined. This consists of identification and action for each severity level information.

- The identification section heads each Responses entry and looks like this:

  ```
  <<Sentry2.0><diskusedpct><Sentry_2hr_weekday#Cool_Sentry_234907.1.
  531#Sentry::All#__0e234907.1.332>>
  ```

  This is the same identification as used in the Timer and Monitoring Probes parts of the output and is broken down as follows:

  - The first field is the name of the Sentry monitor collection.
  - Field two is the name of the monitor.
  - Field three is the full name of the Sentry profile.

- The severity line will be the identification string repeated and followed by the severity name.

- The action entry can be one or more of the following (separated by commas):

icon	Change indicator collection icon.
popup	Pop-up dialog on listed administrator's desktops.
mail	Send e-mail to specified e-mail addresses (does not have to be Tivoli administrators).
notify	Post Tivoli notice to specified notice group.
file	Log to specified file.
tec	Send an event to Tivoli Enterprise Console. The entry identifies the TEC server and the corresponding TEC severity level.

exec Execute command.
taskname The name of a Tivoli task to execute.

12.11 Distributed Monitoring Recovery Tools

If there are difficulties manipulating the Sentry Engine or distributing monitors to a specific endpoint, two tools are available from the Tivoli US Support Center through your normal support channels:

clear_dm.sh Clears the Sentry Engine totally and cleans-up any left over attributes in the oserv database that may be the cause of the problems.

wreloadeng.sh A tool to rebuild the Sentry Engine based upon the subscriptions and what has previously been distributed to that endpoint.

These tools are not part of the standard Tivoli product and may not be available in all locations. In most localities, you should be able to obtain a copy by contacting your IBM or Tivoli representative.

Chapter 13. Inventory

Tivoli Inventory is a hardware and software inventory-gathering application designed to help system administrators monitor and record changes in software and hardware configurations.

13.1 Tivoli Inventory Overview

Tivoli Inventory collects and stores hardware and software information about the systems within a single TMR. Inventory uses a third-party relational database management system (RDBMS) through the Tivoli Framework RIM component. The RDBMS is used to store all the information gathered.

Inventory can gather information from a variety of platforms. These platforms are UNIX managed nodes (as of Version 3.6, software information can also be gathered), Windows NT managed nodes, PC managed nodes (Windows NT, Windows 3.1.1, Windows 95, Windows 98, NetWare, and from NetWare managed PCs by way of Tivoli's Netware Managed Site). With Tivoli Inventory Version 3.6, information can also be gathered from TMA endpoints - including all the above platforms and OS/2 Warp 4.0 and OS/2 Warp Server 4.0.

Information to be included in the inventory database is basically hardware and software information collected from the endpoints. Furthermore, the UserLink Inventory HTML interface can be used to let end-users supply specific information, and user-scripts can be developed to gather additional data. The basic Inventory database-schema supports the information gathered by the Tivoli supplied scanning programs. When including additional information from UserLink Inventory or custom-built scanning scripts, the database schema has to be updated in order to hold the additional information. Please refer to the *Tivoli Inventory User's Guide* for further details on expanding the database schema.

Like other Tivoli applications, Tivoli Inventory is based on the management-by-subscription paradigm. This means that the information controlling the actions is kept in a profile, which is distributed to an endpoint. The specialized profile for the Inventory application, not surprisingly, has the name, InventoryProfile.

Detailed information on what to do and how to do it is kept in the inventory profile. As shown in Figure 178 on page 432 the inventory profile controls whether to scan, to import, or both. In addition, it can be specified what to scan: Hardware, software, and/or user-data through a script.

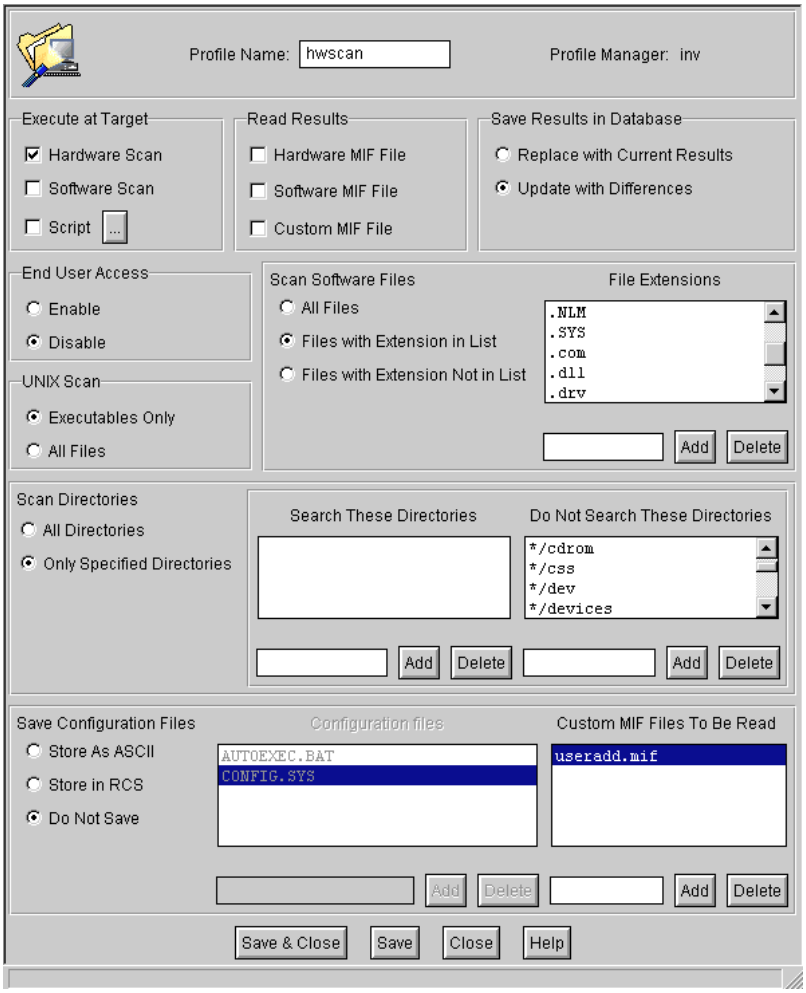


Figure 178. The Inventory Profile

The **Save Results in Database** option was introduced in Inventory 3.2. This option specifies whether to compare the MIF files from a scan with the previous scanning results and transfer only the differences to the TMR Server, which passes them on to the RIM host. If no previous scan exists, or **Replace with Current Results** is specified, the full content in the MIF files is transferred to the RIM host, through the TMR Server, for insertion into the Configuration Repository.

New with Tivoli Inventory 3.6

The Enduser Access option is new with Inventory 3.6.

This parameter controls whether the inventory profile is shown in the UserLink Inventory HTTP page when it is loaded from an endpoint subscribing to the profile. This enables the user to initiate scanning of the workstation.

No matter what kind of information is gathered from the endpoints, the basic function of the Inventory application can be split into:

Scanning A scanning program or script is started at the endpoint. This gathers the desired information and stores it in a Management Information File (MIF) file in accordance with the Desktop Management Task Force (DTMF) Version 2.0 MIF file specifications. This process is fully handled on the endpoint with the exception of TMA endpoints that may have to download programs from the gateway.

When the scanning completes, the resulting MIF file is stored where it is readily available for the following import.The actual location varies dependent upon the type of the endpoint that was scanned:

TMA The MIF files are stored on the endpoints themselves. They are stored along with the scanning programs, which are downloaded from the gateway to the `$LCFROOT\inv\SCANNER` directory.

PC Managed Node and NetWare Managed Sites
The MIF files are uploaded to the resource-host of the PC managed node. Here, they are stored in a directory named after the full object ID of the scanned node in the `$DBDIR\inventory` directory.

Managed Node
When a managed node is scanned, the MIF files are stored in a subdirectory, in the `$DBDIR\inventory` directory, named after the full object ID of the managed node itself.

Import The consolidated data is uploaded to the TMR Server, then on to the RIM host and into the Inventory Configuration Repository implemented in an RDBMS.

The Inventory processes are outlined in Figure 179 on page 434:

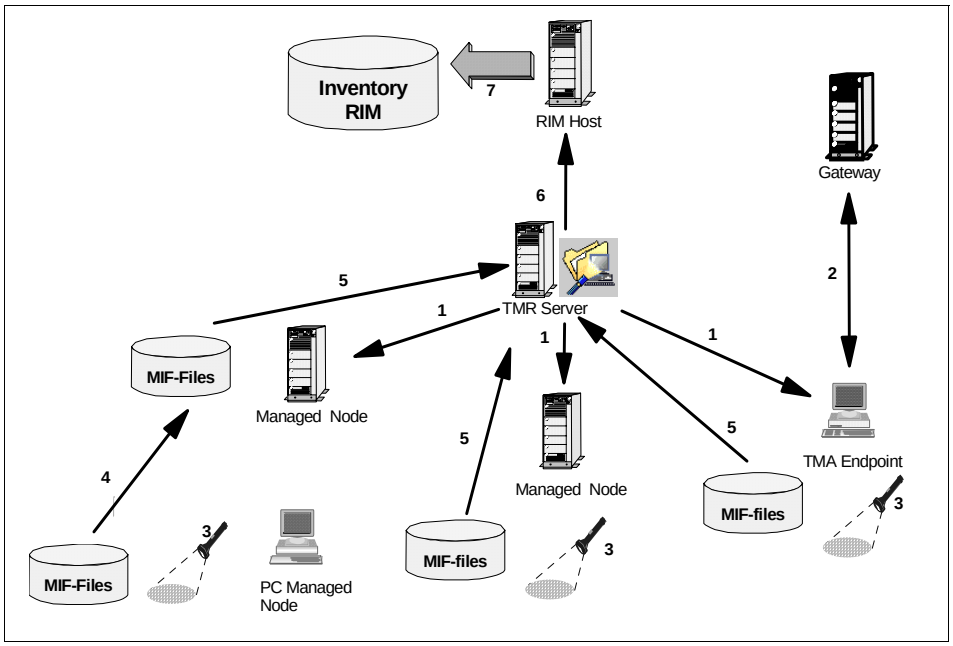


Figure 179. Inventory Process Overview

1 The Inventory is distributed to the endpoints.

2 For TMA endpoints, there might be a need to download the proper scanning programs from the gateway.

3 The scanning program is launched and the results saved in the MIF files.

4 For PC managed nodes, the resulting MIF files are uploaded to the managed node hosting the PC managed node.

5 On import, data from the MIF files are optionally compared to the files from the last run, and data is parsed into database format and is uploaded to the TMR Server.

6 The TMR Server passes the data on to the RIM host.

7 The RIM host loads the data into the RDBMS.

13.1.1 Other Sources of Information

When working with Tivoli Inventory, in addition to the product manuals, the following sources of information are available:

- Redbook SG24-5112 *Using Databases with Tivoli Applications and RIM.*

- Redbook SG24-2135 *TME 10 Inventory 3.2: New Features and Database Support.*
- Support FAQ database at `http://www.support.tivoli.com`

13.2 Inventory Installation Considerations

The Inventory application consists of three main components:

Inventory

This component has to be installed on the TMR Server and can be applied to managed nodes that will be part of the Inventory environment. The Inventory component contains all the binaries and control files needed to create and distribute inventory profiles, run inventory commands from the CLI, as well as execute hard- and software-scans on a managed node.

Inventory Gateway

This component is installed on all gateways that will support TMA endpoints. The component contains the methods needed to run Inventory scans on the TMA endpoints, and when applied to the gateways, they can be accessed by the TMAs through upcalls. In order to be able to scan the gateway itself, either the Inventory (managed node) component or a TMA endpoint must be installed on the gateway.

PC Scanning Program

This is installed on top of the PC Agent on NetWare Managed Sites and PC managed nodes to enable them for scanning.

13.2.1 Inventory Scanning Space Requirements

During deployment planning, a vital parameter is disk space. In order to do this correctly, it is important to remember that more than one copy of the scanning results are stored, and that the scanning results from PC managed nodes and NetWare Managed Sites are stored at the managed node acting as a resource host or the NetWare Server, respectively.

Calculating the space requirements for an Inventory deployment is not an easy task. The space needed depends very much on how much information the software scanners are told to look for and where to look. For planning purposes, the following space requirements (for the scanning results only) should be used as guidelines.

13.2.1.1 PC Managed Nodes/Managed Nodes

The data from the last scan (the MIF files) are stored on each PC managed node in directory `scan\output` in the PC managed node installation directory. A hardware scan typically produces about 35 Kilobytes of data. A software scan from a typical workstation produces about 130-170 Kilobytes. So, the data space needed on the PC managed node will be around 200 Kilobytes.

This figure comes from a scan with the option to scan only for files of type EXE, DLL, NLM, and so on, which is always recommended. Also, bear in mind that a big server with huge disks can produce a lot more software scan data.

The data from the last two scans of each PC managed node are stored on its resource host, in directory `$DBDIR\inventory\<PcManNodeObjId>`, where the scan data for the managed node itself is also stored.

An office with one managed node and ten PC managed nodes would require an average space on the managed node of 4.4 MB (11 hosts * 2 * 0.2 MB).

An office with one managed node and 100 PC managed nodes would require an average space on the managed node of 44.4 MB (101 hosts * 2 * 0.2 MB).

13.2.1.2 TMAs / Gateways

The data from the last two scans are stored on each endpoint PC in the directory `$LCFROOT\Tivoli\lcf\inv\Scanner`. So, each endpoint needs about 0.4 MB.

Only inventory data from the gateway (managed node) itself is stored on the gateway. So, each gateway needs about 0.4 MB.

13.3 Inventory Installation

There are six phases to deploying Tivoli Inventory detailed in this section:

1. The TMR Server installation phase
2. Creating the Configuration Repository
3. The gateway installation phase
4. The managed node installation phase
5. The PC managed node installation phase
6. The TMA installation phase

13.3.1 Installing Inventory on the TMR Server

Installing Tivoli Inventory on the TMR Server serves two purposes:

1. It enables the use of the Inventory application within the TMR.
2. It provides the scripts required to set up the Configuration Repository in a RIM-connected external RDBMS.

Installation is initiated using the familiar Tivoli installation dialog or using the `winstall` command.

13.3.1.1 RIM Installation Options

During installation, the installation options will ask several questions about the RDBMS and how to communicate with it. As with any product using RIM and a RDBMS, the DBA should be involved in the installation process.

For hints and tips on which values to supply to the various fields relating to the RDBMS, refer to the installation chapter in the *Tivoli Inventory User's Guide* and Chapter 9, "RDBMS Interface Module (RIM)" on page 313. Another fine source of information regarding RIM is the redbook SG24-5112 *Using Databases with Tivoli Applications and RIM.*

Having supplied the parameters for the installation process to automatically create the inventory RIM object, the installation process continues the same as any other Tivoli installation. Refer to the *Tivoli Inventory User's Guide* for details.

13.3.2 Creating the Configuration Repository

The Tivoli Inventory application requires an RDBMS that is active and running. This database should be available, locally or through TCP/IP, to what is, or what will become, the RIM host supporting Inventory within the TMR.

Refer to the *Tivoli Inventory Release Notes* for your version of Inventory for information on currently supported databases. Note that even though the Tivoli Framework may support a database, Inventory will not support it until the scripts are available to set up the schema.

RIM Note

The RIM methods that the RIM host needs to access the RDBMS are provided by the Tivoli Framework. Therefore, Inventory only needs to be installed on the RIM host if you want to scan that host as a managed node not as a TMA.

Oracle / HP-UX Note

If you are installing Inventory with Oracle as the RDBMS on an HP-UX system, make sure there is a tmersrvd user ID.

Tivoli Inventory provides configuration scripts that create the actual Inventory database, the RDBMS user for Inventory, and all the views and schemas used by Inventory within the database server. The setup scripts can be found in `$BINDIR/TME/INVENTORY/SCRIPTS/RDBMS`. The following list shows the Inventory 3.6 scripts:

```
# pwd
/usr/local/Tivoli/bin/aix4-r1/TME/INVENTORY/SCRIPTS/RDBMS
# ls -l
-rwxr-xr-x   1 root     system       324 Jul 30 17:46 tivoli_db2_admin.sql
-rwxr-xr-x   1 root     system     52357 Jul 30 17:46 tivoli_db2_schema.sql
-rwxr-xr-x   1 root     system       271 Jul 30 17:46 tivoli_ms_sql_admin.sql
-rwxr-xr-x   1 root     system     63061 Jul 30 17:46 tivoli_ms_sql_schema.sql
-rwxr-xr-x   1 root     system       745 Jul 30 17:46 tivoli_ora_admin.sql
-rwxr-xr-x   1 root     system     74663 Jul 30 17:46 tivoli_ora_schema.sql
-rwxr-xr-x   1 root system      1286 Jul 30 17:46 tivoli_syb_admin.sql
```

The Configuration Repository can be created either before or after installing Inventory on manage nodes, gateways, PC managed nodes, or Netware managed sites as long it is available when the first scanning is initiated distributing an inventory profile.

To create the Configuration Repository, do the following:

1. Log into the RDBMS as a user that has DBADM authority.
2. Execute the `tivoli_DBNAME_admin.sql`.
3. Log out from the RDBMS.
4. Log in as the newly created Tivoli RDBMS user, typically `tivoli` using the password `tivoli`.
5. Execute the `tivoli_DBNAME_schema.sql`.
6. Log out as required.

Note

Prior to running the scripts to create the Configuration Repository, the scripts themselves, especially `tivoli_DBNAME_admin.sql`, should be tailored by the DBA for device usage, user definitions, and space allocation.

Access to the newly created database can be tested using the `wrimtest` command. Details are in Chapter 9, “RDBMS Interface Module (RIM)” on page 313.

13.3.3 Installing Queries

Having created the Configuration Repository, we can add query libraries and queries.

The queries supply a set of standard reports gathering information from the Configuration Repository. These are provided primarily for use by the system administrators but can also be used internally by the Tivoli applications to generate lists of endpoints in the environment that has certain common attributes.

Furthermore, in order to be able to use the hardware and software inventory options for defined nodes/subscribers from the GUI (place the cursor on the desired subscriber and press the right mouse button), the Inventory queries have to be installed.

Tivoli Inventory provides a script to create the query library and related queries. This can be found in `$BINDIR/TME/INVENTORY/SCRIPTS/QUERIES`. The following list shows the Inventory 3.6 query scripts:

```
# pwd
/usr/local/Tivoli/bin/aix4-r1/TME/INVENTORY/SCRIPTS/QUERIES
# ls -l
-rwxr-xr-x   1 root     system     11611 Jul 30 17:47 inventory_queries.sh
-rwxr-xr-x   1 root     system      4351 Jul 30 17:47 subscription_queries.sh
```

Figure 180. Inventory Query Installation Scripts

To install the Tivoli Inventory query library, and the queries within the library, run the `install_queries.sh` script.

As usual, refer to the *Tivoli Inventory User’s Guide* for details.

13.3.3.1 Adding Support for Software Distribution 3.6

In an Inventory 3.6 environment, where Tivoli Software Distribution 3.6 is used, a set of queries, especially designed to report the status of installed software components, can be applied.

To do this, run the **subscription_queries.sh** script from `$BINDIR/TME/INVENTORY/SCRIPTS/QUERIES`.

13.3.4 Adding Software Signatures

When scanning for software, Tivoli Inventory locates name, date, and size information of files in accordance with the options set in the inventory profile. Date, name, and size-information for each individual file, found by the scanning process, is conveyed back to the Configuration Repository through the tivsscan.mif and imported into the table INSTALLED_UNKNOWN_FILE.

Tivoli provides a set of files holding the equivalent date, name, and size information for known files along with an application name. This information is known as a *software signature*. The signatures can be imported into the SOFTWARE_SIGNATURE_FILE table of the Configuration Repository.

The list of files in one table and set of signatures in another are joined in the INSTALLED_SOFTWARE_VIEW to provide support for querying or reporting of what software components are installed where. This kind of information obviously relies heavily on what software signatures are installed.

The standard Tivoli software signatures are delivered in the `$BINDIR/TME/INVENTORY/SCRIPTS/SIGNATURES` directory:

```
# pwd
/usr/local/Tivoli/bin/aix4-r1/TME/INVENTORY/SCRIPTS/SIGNATURES
# ls -l
-rwxr-xr-x   1 root     system     17693 Jul 30 17:47 HPUX_SIGS.INI
-rwxr-xr-x   1 root     system     11985 Jul 30 17:47 SOLARIS_SIGS.INI
-rwxr-xr-x   1 root     system    682960 Jul 30 17:47 SWSIGS.INI
```

Updated signature files can be downloaded by registered customers from the Tivoli support Web site: `http://www.support.tivoli.com/inv`

To install a signature file, use the `wfilesig` command:

```
wfilesig -a -f <filename>
```

To add file signatures for applications not covered by the signature collections provided by Tivoli, either the `wfilesig` command or the *Software Signature Editor*, provided with the UserLink Inventory html-interface, which can add, modify, or delete signatures, can be used.

Usage tip

The `wfilesig` command can be used to generate a list of all the installed software signatures using an undocumented argument `-z`. This option is used to fill the HTML-based Inventory software signature editor.

13.3.5 Installing Inventory on Gateways

Inventory gateway installation only applies when using Tivoli Inventory Version 3.6 or later.

The sole purpose of installing Tivoli Inventory 3.6 on a gateway is to make the Tivoli Inventory files available to TMAs connected to the TMR through the gateway.

The Inventory gateway should, like other gateway components, be installed on all the gateways in the TMR in order to support the scanning of endpoints regardless of which gateway the endpoints are connected through.

OS/2 note

With Tivoli Inventory Version 3.6, the Inventory gateway cannot be installed on an OS/2 gateway.

The installation process is initiated from the TMR Server and is performed through SIS, the installation GUI, or using the `winstall` command. See *Tivoli Inventory User's Guide* for details.

To use Inventory to scan a gateway, it would either need to be installed on the gateway as a managed node, or the scan could take place if the gateway were also a TMA endpoint.

13.3.6 Installing Managed Nodes

In order to be able to scan managed nodes for information regarding hardware and software using Tivoli Inventory, the application has to be installed explicitly on each managed node. The installation process makes the needed files, executables, message catalogs, and so on, available to the managed node.

Besides providing the capability of scanning hardware and software information from the managed node, installing Tivoli Inventory on a managed node enables the creation and distribution of inventory profiles on/from the managed node itself and the use of inventory commands on that managed node.

If providing scanning capabilities is the only purpose for installing Tivoli Inventory on a managed node, new options have to be considered when using Tivoli Inventory Version 3.6. If a TMA has been installed on the managed node, this can be used for scanning purposes. The files supporting the scanning process will be downloaded from the gateway as needed, thus

minimizing the disk usage and administration/maintenance effort at the expense of Network usage. From a scanning perspective, the use of a TMA, instead of the managed node itself, does not result in any restrictions since Tivoli Inventory does not, as many Tivoli applications do, copy profiles to the local oserv database of the managed nodes. From a scanning-only point of view, the two implementations are identical, and the use of TMA is preferred if operational or Network issues does not contradict it.

Installation on a managed node is similar to installation on the TMR Server. The installation has to be initiated from the TMR Server, and the installation GUI, SIS, or the `winstall` command can be used. During installation, the RIM options must be supplied, but since the inventory RIM object has already been created, the values given will not be used to create a new RIM object. Further details can be found in the *Tivoli Inventory User's Guide*.

13.3.7 Installing Inventory on PC Managed Nodes

Adding support for the Tivoli Inventory application on PC managed nodes requires the installation of the *PC Scanning Program*, which is in the same Inventory CD-ROM and can be installed through the Product Install dialog or SIS.

PC Managed Node Notes

The PC Scanning Program MUST be installed in all the PCs that will be scanned. The PC Scanning Program installation requires that the Tivoli Framework PC Agent Version 5.000 or above must be installed first.

With Inventory Version 3.6, it is not possible to install the PC Scanning Program on a PC managed node that has the same label as an existing managed node.

13.3.8 Installing Inventory on TMAs

No specific installation process is required to include TMAs in the Inventory environment except that of installing the Tivoli Inventory gateway component on the gateways. As for other applications supporting the TMAs, the needed modules will be downloaded to the TMAs when needed.

The files needed to scan for hardware and software information on a TMA endpoint is downloaded to the TMA using ordinary up and downcalls to/from the gateway, and thus, are eligible for deletion if the download cache on the TMA grows beyond the predefined size. Even though the files are under control of the TMA cache, all of them but one are not stored in the

$LCF_DATDIR\cache directory. Instead they are stored in an inv directory off the $LCFROOT directory. The exception from that rule is the inv_endpt_meths.exe program. This program is used to control communication with the gateway and the TMR Server and for scanning for software. On TMAs, this program is stored in: $LCF_DATDIR/cache/bin/w32-ix86/TME/INVENTORY/INV_ENDPT

The results from the scanning process, the MIF files, will be stored in the same location as the hardware scanning programs. This is also true for the software MIF file tivsscan.mif.

13.4 Configuring Inventory

On the TMR server, there are new database classes and roles created when Inventory is installed. You need to update these resources in the policy region that will make use of them.

The new object classes are:

- InventoryProfile
- Query (created by Framework)
- QueryLibrary (created by Framework)
- RIM (created by Framework)

These should be made current resources in the policy regions in which you will operate Inventory.

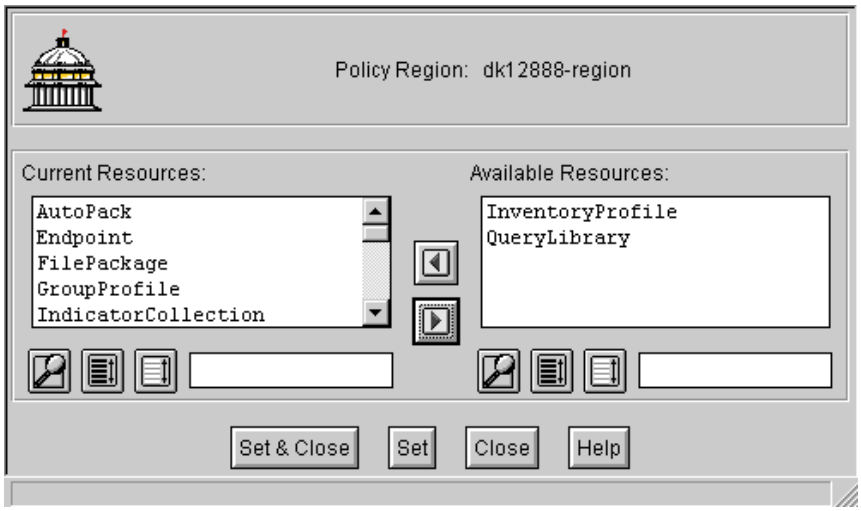


Figure 181. New Inventory Resources

The new roles are:

- Inventory_view
- Inventory_scan
- Inventory_edit
- Inventory_query
- Query_view (created by Framework)
- Query_execute (created by Framework)
- Query_edit (created by Framework)

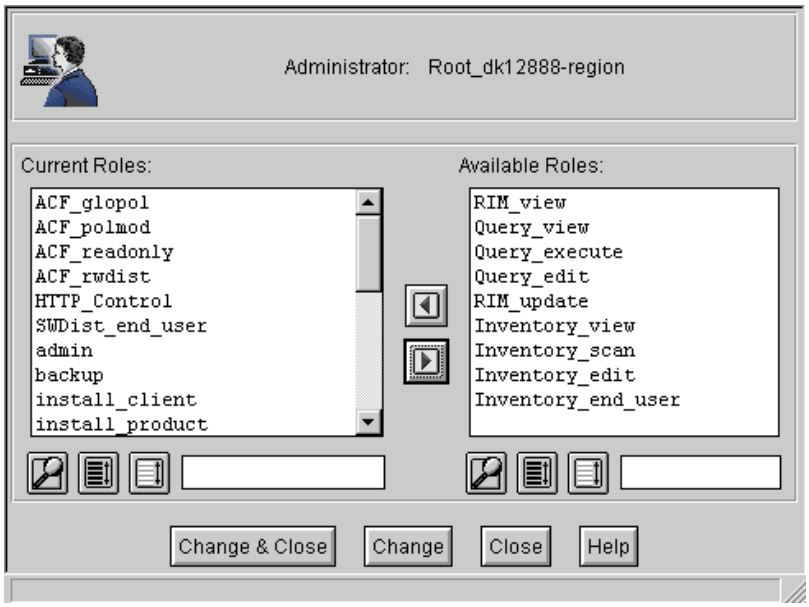


Figure 182. Inventory Roles

The new profile (InventoryProfile) allows Inventory to be set up with the management-by-subscription paradigm. The clients to be scanned are added to the profile manager, and the Inventory equivalent of distribute performs a scan on all or a subset of those clients. After a successful scan, the information is stored in the Inventory database. This information can be seen from the endpoint pull-down menu or through queries and reports from the database.

13.5 Customizing Inventory

The behavior of the scanning process is customized in each individual inventory profile. In the profile, it is determined what to scan for. None, or all three options, hardware, software and script, can be activated simultaneously.

When using a custom script to provide a user-information MIF file, the script itself is kept within the inventory profile and, thus, automatically distributed to the endpoints.

Similar to the selection of what to scan, decided in the profile is what data to import into the Configuration Repository. Again, all or none can be selected. When selecting **Custom MIF File**, the selections made in **Custom MIF Files To Be Read** (at the bottom right in the panel) becomes active.

New with Inventory 3.6 is the **End User Access** that, when enabled, allows end users to initiate the scanning through the UserLink Inventory Interface.

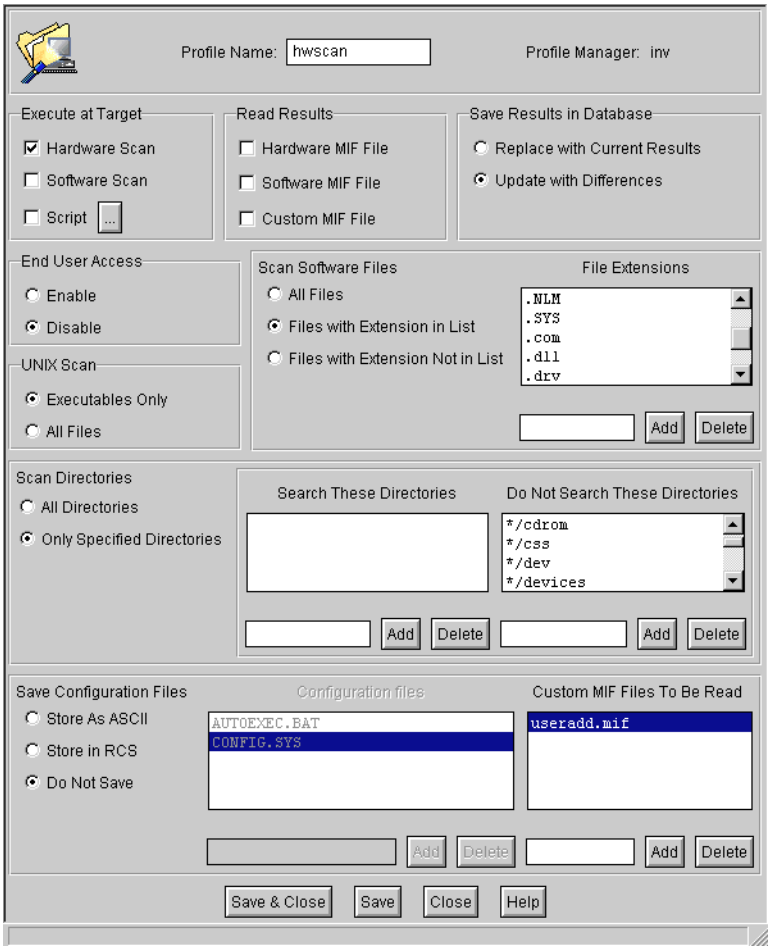


Figure 183. Inventory Profile Dialog

For Intel machines, Inventory 3.6 also offers scanning of the BIOS data as well as scanning of DMI resources on Windows NT and Windows 95.

Scanning software files

Remember to exercise caution when defining the filters for software file scanning. Information for each and every file caught by the filter will be stored in the Configuration Repository. This can be an extensive amount of data.

For PC managed nodes, the tivsscan.mif file is stored in two versions on the managed node acting as resource host for the PC managed node. Since the MIF files can become large, it should be planned to make adequate space available on the managed node.

13.6 Distributing the Inventory Profile

Inventory profiles are distributed in the same way as other profiles in the Tivoli framework either through the GUI or through the CLI using the `wdistrib` command.

Unlike other profiles, however, the inventory profile does not support the notion of local profile copies and can, therefore, not leave copies to be modified somewhere in the distribution hierarchy. Whenever an inventory profile is distributed to a subscribing profile manager, the subscribers of the subscribing profile manager will receive the inventory profile. The distribution works as though the `-a` (all levels) parameter of the `wdistrib` command is always applied.

Likewise, because local profile copies are not used, the `wdistrib` options to control exact copy versus preserve modifications does not apply to inventory profiles.

Note

When a distribution is initiated, the first thing Inventory does is to connect to the Configuration Repository through the RIM host. This occurs regardless of whether data actually will be imported into the Configuration Repository or not.

The implications of this behavior are that, in order to distribute an inventory profile, both the RDBMS server and the RIM host have to be available even if the inventory profile only initiates scanning processes.

13.7 Inventory Scanning Process

Inventory collects hardware and software information from endpoints. When a *scan* is initiated on a managed node, PC managed node, NetWare managed site, or TMA endpoint, the following occurs:

1. Inventory runs scanning software (scanner) on each endpoint.
2. The scanner gathers information and writes it to a MIF file.

When an *import* is initiated on a managed node, PC managed node, on a NetWare managed site of TMA the following occurs:

1. The MIF files are compared and passed. On PC managed nodes, this process is performed by the resource host. For all other types of endpoints, this process is handled locally.
2. If **Update with Differences** is specified in the inventory profile, the MIF files from the current and the previous scans are compared, and differences are converted into database format.
   If **Replace with Current Results** is specified, the current MIF files are converted into database format.
   For PC managed nodes, this process is performed by the resource host.
3. The database-formatted data is sent to the TMR Server that sends it on to the RIM host where it is loaded into the Configuration Repository.

To see the information collected, you can select the icon of the managed node or PC managed node scanned and click the right button of the mouse and select one of the Hardware/Software Inventory options as shown in Figure 184 on page 448:

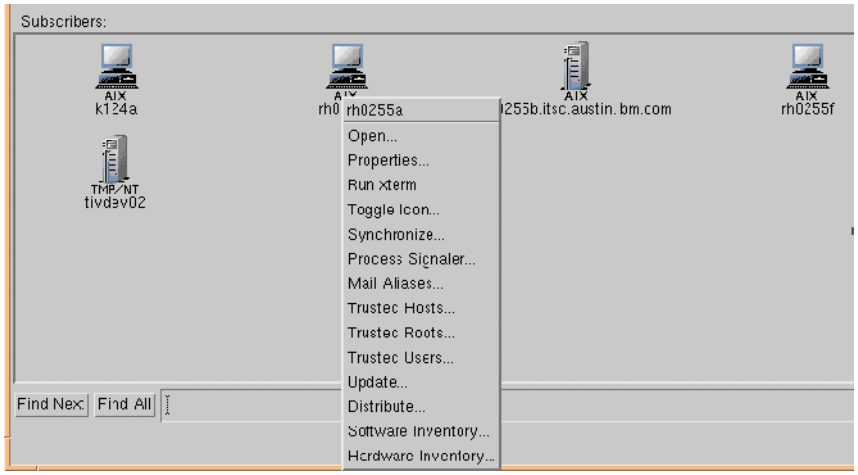


Figure 184. Managed Node Options

When you look at the Inventory hardware or software data for a managed node, you will see one of the following windows as shown in Figure 185 on page 449:

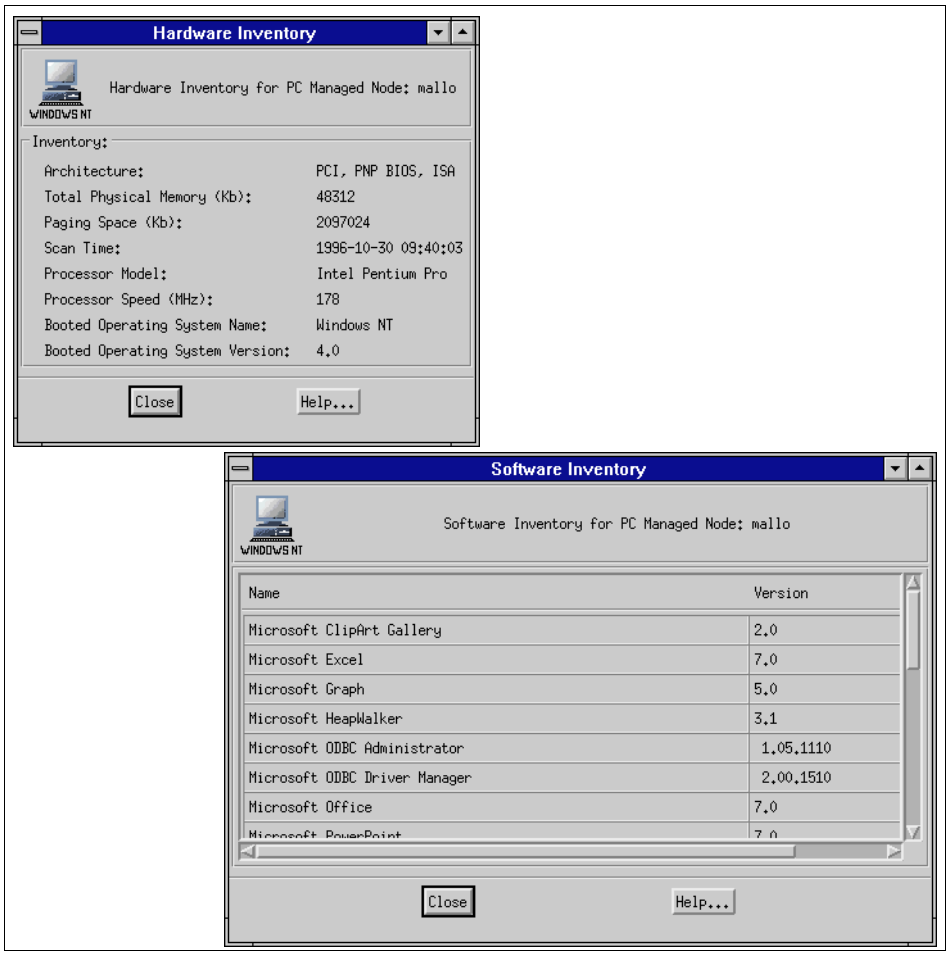


Figure 185. Hardware and Software Inventory Data

13.7.1 Scanning Programs

As of Tivoli Inventory Version 3.2.2, the scanning program was developed entirely by Tivoli. In previous versions, the Intel-based endpoints were scanned by a program from Intel.

With the redesign of the scanning programs, they are split into the several parts shown in Table 18:

Table 18. Inventory Scanning Programs

	Intel	Unix
Hardware scanner	tivhscan.exe	sysinfo
Bios scanner	mrmbios.exe	N/A
Dmi scanner	dmiscan.exe	N/A
Software scanner	inv_endpt_meths.exe	

13.8 Inventory's Use of Methods

This section discusses the process and some of the methods in use when an Inventory scan is performed.

13.8.1 UNIX Managed Node

The `ip_push` method is started on the server and communicates through MDist to begin the Inventory scan. See also section 7.6.1, "Mdist" on page 259.

A UNIX managed node endpoint runs `ip_discover`, which collects hardware information and sends it back to the RIM host. This is illustrated in Figure 186.

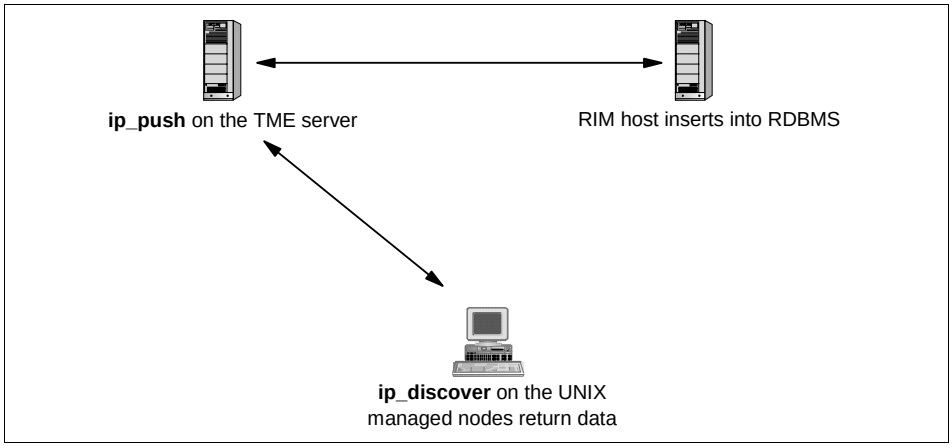


Figure 186. Methods Involved in a UNIX Scan

Here are two examples of an `odstat` of this process. The first is in the TMR server, and the other is from the managed node after you execute a scan.

In the following example, you can see:

- The call of the `ip_push` method on thread 13053.
- The checking of the connection with the RDBMS through the RIM host on threads 13059 and 13060.
- The call of `ip_discover` in the managed node (13068).
- Updating information from the scan in the RDBMS (13074, 075, 096, and 100).

```
13042 O 1-12954 done 18 0 15:08:04 1515280903.2.7#TMF_ManagedNode::Managed_Node# push_copy_in
13043 M hdoq 2-4705 done 97 0 15:08:05 1515280903.1.26 lookup
13044 M hdq 2-4706 done 51 0 15:08:05 1515280903.1.4 lookup_id
13045 M hdq 2-4707 done 162 0 15:08:06 1515280903.1.4##2@InventoryProfile describe
13046 M hdq 2-4708 done 2482 0 15:08:06 1515280903.1.4##2@InventoryProfile _get_type
13047 M 2-4709 done 59 0 15:08:07 1515280903.1.758#InventoryProfile# _get_profile_organizer
13048 M 2-4710 done 21 0 15:08:07 1515280903.1.758#InventoryProfile# _get_label
13049 M hdoqs 2-4711 done 6 0 15:08:07 1515280903.1.478#TMF_UI::Extd_Desktop# execute
13050 M hdoq 2-4712 done 242 0 15:08:07 1515280903.1.538#TMF_CCMS::ProfileManager# default_push_profiles
13051 O+ho 1-13050 done 152 0 15:08:09 1515280903.1.758#InventoryProfile# default_push
13052 O+hdq1-13051 done 152 0 15:08:09 1515280903.1.758#InventoryProfile# push
13053 O+hdq1-13052 done 152 0 15:08:10 1515280903.1.758#InventoryProfile# ip_push
13054 O+ 1-13053 done 15 0 15:08:10 0.0.0 get_name_registry
13055 O+hdoq1-13053 done 675 0 15:08:10 1515280903.1.26 object_get_all
13056 O+hdoq1-13053 done 536 0 15:08:11 1515280903.1.538#TMF_CCMS::ProfileManager# get_subscription_tree
13057 O+ 1-13056 done 886 0 15:08:11 1515280903.1.538#TMF_CCMS::ProfileManager# _get_subscribers
13058 O+hdoq1-13053 done 111 0 15:08:11 1515280903.1.26 lookup
13059 O+hdoq1-13053 done 18 0 15:08:12 1515280903.1.755#RIM::RDBMS_Interface# RIM_connect
13060 O+hdoq1-13053 done 12 0 15:08:13 1515280903.1.755#RIM::RDBMS_Interface# RIM_release
13061 O+hdoq1-13053 done 95 0 15:08:13 1515280903.1.26 lookup
13062 O+hdoq1-13053 done 167 0 15:08:13 1515280903.1.345 obj_route
13063 O+ 1-13062 done 15 0 15:08:14 0.0.0 get_name_registry
13064 O+hdoq1-13062 done 545 0 15:08:14 1515280903.1.26 get_all
13065 O+hdoq1-13062 done 545 0 15:08:14 1515280903.1.26 get_all
13066 O+hdoq1-13053 done 241 0 15:08:15 1515280903.1.26 lookup
13067 O+ 1-13053 done 18 0 15:08:16 1515280903.1.713#TMF_Install::ProductInfo# _get_locations
13068 O a 1-13053 done 2521 0 15:08:16 1515280903.2.7#TMF_ManagedNode::Managed_Node#
ip_discover
13069 O+hdoq1-13053 done 97 0 15:08:22 1515280903.1.26 lookup
13070 O+hdq1-13053 done 59 0 15:08:22 1515280903.1.4 lookup_id
13071 O+hdq1-13053 done 174 0 15:08:22 1515280903.1.4##2@TMF_NetWare::ManagedSite describe
13072 O+hdq1-13053 done 3018 0 15:08:22 1515280903.1.4##2@TMF_NetWare::ManagedSite _get_type
13073 O+hdoq1-13053 done 111 0 15:08:23 1515280903.1.26 lookup
13074 O+hdoq1-13053 done 18 0 15:08:23 1515280903.1.755#RIM::RDBMS_Interface# RIM_connect
13075 O+hdoq1-13053 done 12 0 15:08:23 1515280903.1.755#RIM::RDBMS_Interface#
RIM_delete_rows
13076 O+hdoq1-13053 done 12 0 15:08:23 1515280903.1.755#RIM::RDBMS_Interface# RIM_delete_rows
.....
13095 O+hdoq1-13053 done 12 0 15:08:24 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13096 O+hdoq1-13053 done 12 0 15:08:24 1515280903.1.755#RIM::RDBMS_Interface#
RIM_insert_rows
13100 O+hdoq1-13053 done 12 0 15:08:24 1515280903.1.755#RIM::RDBMS_Interface#
RIM_update_rows
13101 O+hdoq1-13053 done 12 0 15:08:24 1515280903.1.755#RIM::RDBMS_Interface# RIM_update_rows
13102 O+hdoq1-13053 done 12 0 15:08:24 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13120 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13129 O+hdq1-13053 done 60 0 15:08:25 1515280903.1.4 lookup_id
13130 O+hdq1-13053 done 821 0 15:08:25 1515280903.1.4##8@RIM::ExRIMRDBMSCallFailed describe
13131 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13132 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13133 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13134 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13135 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13136 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_insert_rows
13137 O+hdoq1-13053 done 12 0 15:08:25 1515280903.1.755#RIM::RDBMS_Interface# RIM_release
13138 O+hdoq1-13053 done 128 0 15:08:25 1515280903.1.26 lookup
13139 O+hdq1-13053 done 6 0 15:08:25 1515280903.1.88#TMF_SysAdmin::InstanceManager# connect
13140 M hdoqs 2-4714 done 6 0 15:08:26 1515280903.1.478#TMF_UI::Extd_Desktop# execute
```

In the second example, you will only see the `ip_discover` method. If you look at the fourth column and compare it with the first example, you will see the thread ID `1-13068`, which is the TMR server calling this method and the thread ID for the caller of the method.

```
4703 M hdoq1-13042  done    18     0 15:08:05       1515280903.2.7#TMF_ManagedNode::Managed_Node#
push_copy_in
4704 O+     2-4703  done    15     0 15:08:05       0.0.0 get_name_registry
4705 O      2-4703  done    97     0 15:08:05       1515280903.1.26 lookup
4706 O      2-4703  done    51     0 15:08:05       1515280903.1.4 lookup_id
4707 O      2-4703  done   162     0 15:08:06       1515280903.1.4##2@InventoryProfile describe
4708 O      2-4703  done  2482     0 15:08:06       1515280903.1.4##2@InventoryProfile _get_type
4709 O     2-4703  done    59     0 15:08:07       1515280903.1.758#InventoryProfile# _get_profile_organizer
4710 O      2-4703  done    21     0 15:08:07       1515280903.1.758#InventoryProfile# _get_label
4711 O      2-4703  done     6     0 15:08:08       1515280903.1.478#TMF_UI::Extd_Desktop# execute
4712 O      2-4703  done   242     0 15:08:08       1515280903.1.538#TMF_CCMS::ProfileManager#
default_push_profiles
4713 M ho 1-13068  done 2521  0 15:08:17    1515280903.2.7#TMF_ManagedNode::Managed_Node#
ip_discover
4714 O      2-4703  done     6     0 15:08:27       1515280903.1.478#TMF_UI::Extd_Desktop# execute
4715 O+             done    15     0 15:11:30       0.0.0 get_oserv
4716 O+             done 44012     0 15:11:30       1515280903.2.2 query odstat
4717 O+             done    14     0 15:29:30       0.0.0 o_self
4718 O+             done    15     0 15:29:30       0.0.0 get_oserv
4719 O+             done    15     0 15:29:30       0.0.0 get_security_objid
4720 O+             done     0     0 15:29:31       1515280903.2.2 cntl odtrace errors
4721 O+             done    14     0 15:29:38       0.0.0 o_self
4722 O+             done    15     0 15:29:38       0.0.0 get_oserv
4723 O+             done    15     0 15:29:38       0.0.0 get_security_objid
4724 O+             done     0     0 15:29:38       1515280903.2.2 cntl odtrace objcalls
4725 M hdoq1-13143  done    18     0 15:29:51       1515280903.2.7#TMF_ManagedNode::Managed_Node#
push_copy_in
4726 O+     2-4725  done    15     0 15:29:51       0.0.0 get_name_registry
4727 O      2-4725  done    97     0 15:29:51       1515280903.1.26 lookup
4728 O      2-4725  done    51     0 15:29:52       1515280903.1.4 lookup_id
```

13.8.2 Windows NT Managed Node

A Windows NT managed node endpoint runs `ip_discover`, which collects hardware and software information and sends it back to the RIM host through the TMR Server.

- Runs `tivhscan` to gather hardware information and store it in `%DBDIR%\Inventory\OID`.
- Runs `inv_endpt_meth` which collects software data in `%DBDIR%/Inventory/OID`.
- Parses the Inventory data and sends it to the TMR Server, which conveys it to the RIM host. This is illustrated in Figure 187 on page 453.

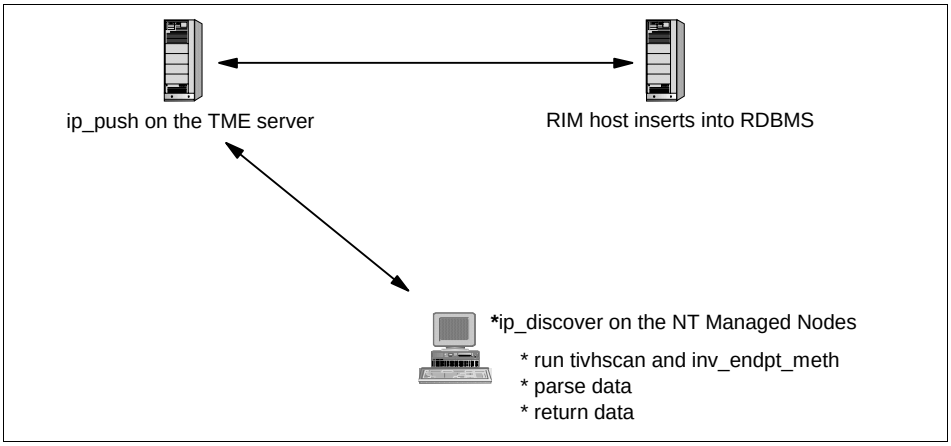


Figure 187. Methods Involved in a Windows NT Scan

13.8.3 PC Managed Node

The resource host (managed node) of the PC managed node runs `ip_discover` to collect hardware and software information, which tells the PC to run `tivhscan.bat` and/or `tivsw.bat`.

After the scan in the PC, the following files can be found in the scan\output directory of the PC managed node installation directory:

- config.mif
- mrmbios.mif
- tivhscan.mif
- tivsscan.mif
- tivsscan.txt

These files are uploaded to the resource host and, if **Update with Differences** is specified in the inventory profile, compared to the result of the previous scan. The resulting information will then be sent to the RIM host through the TMR Server to get inserted into the Configuration Repository. The old MIF files are renamed to **.BK2*, and the new MIF files are given the extension *BK1*; so, they will be used as described previously at the next scan.

Figure 188 on page 454 illustrates the methods involved.

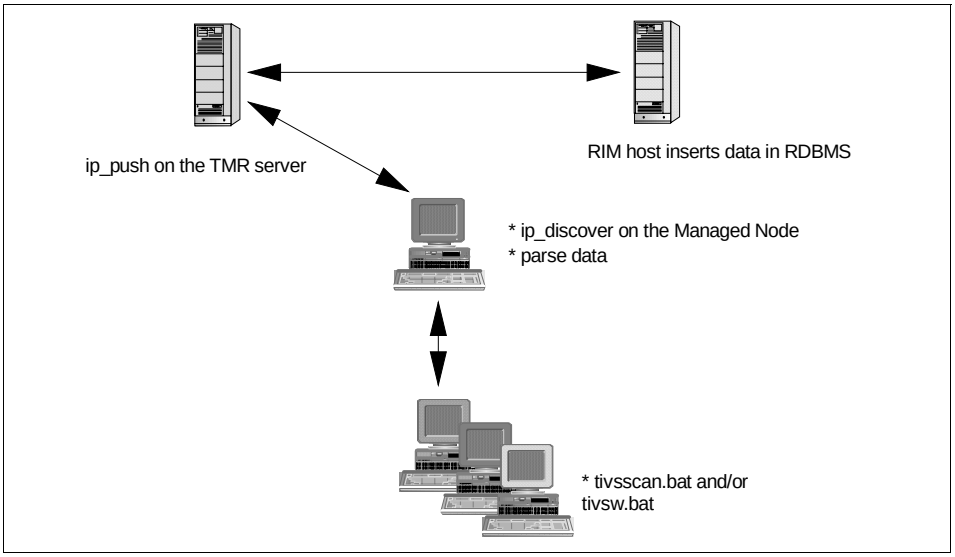


Figure 188. Methods Involved in a PC Scan

13.8.4 TMA Endpoints

The hosting gateway runs the `ip_discover` to collect hardware and software information, which tells the TMAs to run `sysinfo/tivhscan` and/or `inv_endpt_meth`.

The MIF files are compared to those from the previous scan, and the results are uploaded to the TMR Server through the gateway, which acts just like the managed node in the PC managed node scenario. The TMR Server sends the data on to the RIM host.

13.9 Inventory Commands

The following commands are useful when working with Inventory and other RIM applications:

`wlookup` Can be used to identify all the RIM objects in the database:

```
# wlookup -ar RIM
inventory       1515280903.1.755#RIM::RDBMS_Interface#
sybase  1515280903.3.34#RIM::RDBMS_Interface#
tec     1515280903.3.32#RIM::RDBMS_Interface#
```

`wgetrim` Lists information about a RIM object:

```
# wgetrim inventory
```

```
RIM Host:       rh0255b.itsc.austin.ibm.com
RDBMS User:     tivoli
RDBMS Vendor:   Sybase
Database ID:    inventory
Database Home:  /sybinst
Server ID:      rh0255e
```

`wsetrim` Changes the database information for a RIM object.

`wviewmn` Returns hardware and software information about a managed node:

```
# wviewmn rh0255b.itsc.austin.ibm.com
Hardware System Id              : 1515280903.1.327#TMF_ManagedNode
::Managed_Node#
Hardware Architecture           :
Booted Operating System Name    : AIX
Booted Operating System Version : 4.2
Processor Model                 : UNKNOWN
Processor Speed                 : 1
Physical Memory (KB)            : 65536
Paging Space (KB)               : 196608
Time of Last Scan               : 1997-11-13 14:59:23
```

`wviewpcmn` Returns hardware and software information about a PC managed node:

```
# wviewpcmn rh0255d
Hardware System Id              : 7D8BJ9TD9+ZKPVV1LQK800000566 [Wed Nov 12
18:00:25 1997]
Hardware Architecture           : ISA
Booted Operating System Name    : Windows 95
Booted Operating System Version : 3.95
Processor Model                 : 486 DX2
Processor Speed                 : 66
Physical Memory (KB)            : 32268
Paging Space (KB)               : 32268
Time of Last Scan               : 1997-11-21 14:50:45
# wviewpcmn -s rh0255d
Microsoft DOS XMS Driver                            3.95
Microsoft Windows 95                                4.00.950
Microsoft Network Client                            4.2
Microsoft DOS Command Processor                     7.0
Microsoft DOS Command Processor                     7.0
Microsoft Windows Write                             X
```

There are a number of ways to get a list of the tables Inventory creates in the database:

- Look in the Inventory manuals. This is not necessarily going to be accurate.
- Use SQL commands, such as `sp_help` in Sybase (but this will not list all columns in the table).
- Look in `$BINDIR/TAS/RIM/SQL/scripts` and look at the schema files for Oracle, Sybase, and so on, and look for the `CREATE TABLE` paragraphs:

```
CREATE TABLE COMPONENT_MONITOR (
MONITOR_NAME         varchar(32) NOT NULL,
SOFTWARE_COMPONENT_NAME varchar(32) NOT NULL,
SOFTWARE_COMPONENT_VERSION varchar(32) NOT NULL, COMPONENT_LANGUAGE_EDITION
varchar(16) NOT NULL, MONITOR_PROFILE_NAME varchar(32) NOT NULL
)
go

ALTER TABLE COMPONENT_MONITOR
ADD  PRIMARY KEY (MONITOR_NAME, SOFTWARE_COMPONENT_NAME, SOFTWARE_COMPONENT_VERSION,
COMPONENT_LANGUAGE_EDITION, MONITOR_PROFILE_NAME)
go
```

13.10 Querying the Inventory Database

All the information that you have in the database can be consulted by queries defined in the Query Libraries. You can execute these queries for tasks such as a Software Distribution, subscribing managed nodes to a profile manager, and others.

User Defined Data

Having added custom data to the scanning, new tables have been introduced in the database schema of the Configuration Repository. To make these tables available to the queries, the tables or views defined over these tables should be added to the QUERY_VIEWS table.

Examples of this can be found in the script used to create the original database schema. For example:
`$BINDIR/TME/INVENTORY/SCRIPTS/RDBMS/tivoli_syb_schema.sql`

Look for lines like:
`insert into QUERY_VIEWS (VIEW_NAME) values ('SWDISTDATA_VIEW')`

When you run a query, the information is retrieved from a set of database tables. The fields from which the information is retrieved are determined by the RDBMS view specified for the query. A view is a custom table that groups and retrieves information from related fields.

All the standard views, defined when creating the Configuration Repository database schema, are described in the *Tivoli Inventory User's Guide*.

13.11 Troubleshooting Inventory

Inventory does not provide any log files. The standard sources of information, such as `odstat`, `wtrace`, and `tmstat` commands, must be used.

Troubleshooting an inventory profile distribution involves more nodes including the endpoint, the managed node hosting the profile, and the RIM host. For TMAs, the gateway also has to be taken into account.

13.11.1 The Endpoints

To pinpoint a problem during scanning of the endpoints, the following information can be useful:

Time-stamp of the MIF and BK1 files

The time-stamp of the MIF file(s) indicates the last time the scanning was performed successfully. The time-stamp of the bk1-files indicates the previous successful scan.

The process list

During scanning, use the proper utility to list the active processes to see if the expected program is running.
On OS/2, use the `pstat` utility, on Windows 95 and Windows NT use the task list or `ntprocinfo` (NT only).

13.11.2 The Managed Node

You are likely to encounter problems in Inventory if you do not assign the correct Inventory roles to the administrator. You will probably have problems when executing Inventory commands through the CLI.

This is illustrated in the following example:

```
# wviewmn rh0255b.itsc.austin.ibm.com
An authorization error of type "insufficient authorization" occurred

Summary of possible error conditions:
 "insufficient authorization" means that you have insufficient TME authorization to perform the operation
 "permission denied" means that you have insufficient operating system privileges to perform the operations.
 "authorization information expired" means that a task's authorization to run has expired
 "Kerberos ticket expired" means that your Kerberos ticket expired and you must run kinit to get a new ticket.
 "Kerberos unauthorized request" means that Kerberos has rejected the operation.
 "Delegation Credential failure" means that invalid security credentials were used in the operation.
```

You can also see the error in the odstat output:

```
  4496 O+hdoq         done   122     0 14:51:17        1360991896.1.26 lookup
  4497 O+hdoq         done   110     0 14:51:17        1360991896.1.26 lookup
*4498 O              done     0     0 14:51:17 UNAUTHORIZED
   1360991896.2.33#RIM::RDBMS_Interface# RIM_connect
  4499 O     1-1361  done 23523     0 14:51:28
   1360991896.2.35#TMF_PcManagedNode::Pc_Managed_Node# toggle_state
  4500 M      2-325  done 40925     0 14:51:29
   1360991896.1.344#TMF_UI::Presentation# _get_dialogs
```

Similarly, if the Configuration Repository cannot be contacted, either because any of the RDBMS components are down or the RIM host is not running, you will receive a Could not connect to RDBMS message.

The odstat output indicating the same situation is shown in the following figures:

```
* 3773 O+hdoq 1-2207  done   745     0 13:03:34  e=12
1293184718.1.347#TMF_ManagedNode::Managed_Node# push_copy_in
  3774 O+    1-3773  done    15     0 13:03:37        0.0.0 get_name_registry
  3775 O+hdoq 1-3773  done    97     0 13:03:38        1293184718.1.26 lookup
  3776 O+hdq 1-3773  done    51     0 13:03:39        1293184718.1.4 lookup_id
  3777 O+hdq 1-3773  done   162     0 13:03:39       1293184718.1.4##2@InventoryProfile
describe
  3778 O+hdq 1-3773  done  2482     0 13:03:40       1293184718.1.4##2@InventoryProfile
_get_type
  3779 O+    1-3773  done    60     0 13:03:41       1293184718.1.1595#InventoryProfile#
_get_profile_organizer
  3780 O+    1-3773  done    23     0 13:03:42       1293184718.1.1595#InventoryProfile#
_get_label
  3781 O+hdoqs 1-3773  done     6     0 13:03:42
1293184718.1.529#TMF_UI::Extd_Desktop# execute
* 3782 O+hdoq 1-3773  done   745     0 13:03:43  e=12
1293184718.1.1594#TMF_CCMS::ProfileManager# default_push_profiles
* 3783 O+ho  1-3782  done   745     0 13:03:44  e=12  1293184718.1.1595#InventoryProfile#
default_push
* 3784 O+hdq 1-3783  done   745     0 13:03:44  e=12
1293184718.1.1595#InventoryProfile# push
* 3785 O+hdq 1-3784  done   745     0 13:03:45  e=12
1293184718.1.1595#InventoryProfile# ip_push
  3786 O+    1-3785  done    15     0 13:03:45        0.0.0 get_name_registry
  3787 O+hdoq 1-3785  done   998     0 13:03:45        1293184718.1.26 object_get_all
  3788 O+hdoq 1-3785  done   843     0 13:03:45
1293184718.1.1594#TMF_CCMS::ProfileManager# get_subscription_tree
  3789 O+    1-3788  done   873     0 13:03:46
1293184718.1.1594#TMF_CCMS::ProfileManager# _get_subscribers
```

Figure 189. Odstat - RDBMS Failure (Part 1 of 2)

```
3790 O+    1-3788  done   438     0 13:03:46
1293184718.1.1122#TMF_CCMS::ProfileManager# _get_subscribers
  3791 O+    1-3788  done   225     0 13:03:46
1293184718.1.1633#TMF_CCMS::ProfileManager# _get_subscribers
  3792 O+hdoq 1-3785  done   112     0 13:03:46        1293184718.1.26 lookup
* 3793 O+hdoq 1-3785  done   358     0 13:03:47  e=12
1293184718.1.1063#RIM::RDBMS_Interface# RIM_iom_session
  3794 O+hdoq 1-3785  done    97     0 13:03:58        1293184718.1.26 lookup
  3795 O+hdq 1-3785  done    56     0 13:03:58        1293184718.1.4 lookup_id
  3796 O+hdq 1-3785  done   802     0 13:03:58
1293184718.1.4##8@RIM::ExRIMConnectFail describe
  3797 O+    1-3783  done    15     0 13:03:58        0.0.0 get_name_registry
  3798 O+hdoq 1-3783  done    97     0 13:03:58        1293184718.1.26 lookup
  3799 O+hdq 1-3783  done    63     0 13:03:58        1293184718.1.4 lookup_id
  3800 O+hdq 1-3783  done   802     0 13:03:59
1293184718.1.4##8@InventoryProfile::ExIPError1 describe
  3801 O+hdq 1-3783  done    56     0 13:03:59        1293184718.1.4 lookup_id
  3802 O+hdq 1-3783  done   802     0 13:03:59
1293184718.1.4##8@RIM::ExRIMConnectFail describe
  3803 O+    1-3782  done    15     0 13:03:59        0.0.0 get_name_registry
  3804 O+hdoq 1-3782  done    97     0 13:03:59        1293184718.1.26 lookup
  3805 O+hdq 1-3782  done    63     0 13:03:59        1293184718.1.4 lookup_id
  3806 O+hdq 1-3782  done   802     0 13:03:59
1293184718.1.4##8@InventoryProfile::ExIPError1 describe
  3807 O+hdq 1-3782  done    56     0 13:03:59        1293184718.1.4 lookup_id
  3808 O+hdq 1-3782  done   802     0 13:03:59
1293184718.1.4##8@RIM::ExRIMConnectFail describe
  3809 O+hdq 1-3773  done    63     0 13:03:59        1293184718.1.4 lookup_id
  3810 O+hdq 1-3773  done   802     0 13:03:59
1293184718.1.4##8@InventoryProfile::ExIPError1 describe
  3811 O+hdq 1-3773  done    56     0 13:04:00        1293184718.1.4 lookup_id
  3812 O+hdq 1-3773  done   802     0 13:04:00
1293184718.1.4##8@RIM::ExRIMConnectFail describe
```

Figure 190. Odstat - RDBMS Failure (Part 2 of 2)

13.11.3 The Gateway

Use the ordinary `odstat`, `wtrace` type of information to locate problems regarding up- and downcalls.

13.11.4 The RIM Host

To capture any problems in the RDBMS interactions, use the facilities supplied by the `wrimtrace` command documented in Chapter 9, “RDBMS Interface Module (RIM)” on page 313.

Chapter 14. User Administration

Tivoli User Administration is a profile-based application that runs on the Tivoli Framework. The base product manages UNIX, Windows NT, and NetWare user accounts. It also manages UNIX group accounts, Windows NT, and NetWare group account memberships as well as Network Information Service (NIS) management. Additional products extend Tivoli User Administration to manage accounts on many other platforms including Lotus Domino/Notes, the OS/390 Security Server, OS/400, and so on.

User and group management are profile-based applications that can be distributed to profile manager endpoints. NIS support is provided as an endpoint that allows you to distribute user and group profiles to it and manage the other NIS maps.

Most large organizations will make many customizations to User Administration, such as modifying polices or altering the GUI operation using the Tivoli Application Extension Facility (AEF). This is the way User Administration is extended by most products, such as those mentioned above.

It needs to be stressed that the most important part of a Tivoli User Administration implementation is prior planning. If you do not have a sound, detailed design for managing user and group resources, you will have problems implementing and supporting these resources with User Administration. Refer to the *Tivoli User Administration Design Guide,* SG24-5108 redbook for how to design a User Administration implementation before attempting to architect and implement your solution. See also SG24-5339 *The OS/390 Security Server Meets Tivoli* for information on managing users on OS/390.

14.1 Changes to User Administration with Release 3.6

Almost all of the information presented in the chapter applies to all recent releases of Tivoli User Administration. This section introduces the major changes to User Administration in the 3.6 release:

- Endpoint Management
- Immediate Propagation of Passwords
- Interaction with Tivoli Security Management
- Technology Preview Program

14.1.1 Endpoint Management

User Administration 3.6 can now manage Windows NT and Novell NetWare as TMA endpoints. This requires the User Administration gateway to be running on the endpoint gateways. TMA endpoint for UNIX support is also being added with release 3.6.1. The NetWare TMA endpoint for User Administration was completely re-written for 3.6. You should use this instead of NetWare PC managed nodes if you are running Version 3.6 or higher. Note that the use of a TMA endpoint means that we can use the lcfd.log for problem determination.

Important Note

If you are managing TMA endpoints, it is a good rule of thumb to install the User Administration gateway on all of your endpoint gateway servers. This way, if you move endpoints around or reconfigure your gateways, you know User Administration for TMA endpoints will still work.

14.1.2 Immediate Propagation of Passwords

Password changes made through `wpasswd` are effective in all user profiles that contain records about the user as soon as the system is propagated. A new option, `wpasswd -L`, allows for password changes to immediately propagate to all subscribers for all profiles containing that user without distribution.

14.1.3 Interaction with Tivoli Security Management

User Administration 3.6 interacts more closely with Tivoli Security Management, enabling you to use a role-based security model to implement a comprehensive, consistent security policy across your enterprise.

14.1.4 Technology Preview Program

User Administration 3.6 includes two new innovative technologies you can install and evaluate. They are:

- LDAP Connection - a utility that provides a method for managing user accounts on any system accessible through the Lightweight Directory Access Protocol (LDAP).
- OnePassword - a utility that enables users and administrators to change users' passwords in the Tivoli database and distribute the updated passwords to subscribed endpoints.

These Technology Preview utilities are not covered in this redbook.

14.2 Profile Policy

There are default and validation policies set up for all attributes in user and group profiles. These policies can be edited either from the profile GUI or the CLI. Many implementations of User Administration will involve modifications to default and validation policies; so, we will spend some time reviewing policies as they apply to User Administration here.

From the GUI, this is done from the profile dialog under **Edit->Default Policy** or **Edit->Validation Policy**. From the command line interface, use the `wgetpol` command to extract the policy, then use the `wputpol` command to replace the old policy with the new or updated one.

Each attribute's policy can be set to a constant, a script, a regular expression (validation policy only), or none. When using a script, you choose the arguments to be sent.

Note

If the policy type is set to constant while using a script, the script will be erased. These policies are stored as attributes, which is not the same as policies for other resources. Policies can differ greatly from one user profile to the next, but with some other types of resources, the policy is set for all resources of that type in a given policy region.

Default and validation policies are stored in the profile and can be set for any attribute on any profile or profile copy. This allows different policies on the top-level profile and on profile copies.

For a complete list of the default and validation policies, see the *Tivoli User Administration User and Group Management Guide.*

14.2.1 Default Policy

When adding new records to a profile, all fields can be manually generated with their default values by using the **Generate Defaults** button or by creating the user after filling in a minimal set of attributes. Default policy is run by going through each attribute one at a time and trying to generate the default value. If there is not enough information to generate the default, then that attribute will be skipped, and the next attribute's default policy is run. When all attributes have been checked, the process will start again until there are either no more values to generate with default values or no more values can be generated, which would result in an error.

Note

Populate does not run default policy for any attribute by design. If you customize a user profile, and then add attributes with default values, when the profile is populated, the new attributes will not have default values.

14.2.2 Validation Policy

Validation policy is used to validate the values of a profile attribute upon saving a newly-created or modified record. It can also be checked when the `wvalidate` command is run against a profile, or **Validate** is selected in the GUI. By default, there is no validation policy set for customized attributes.

Validation policies for profile copies are different. The following lists the different conditions in which validation policy is checked:

- If the profile copy has no local modifications (changed attributes or policies), no validation is done.
- If the profile copy has a local modification to an attribute, that attribute is validated when that record attribute changes.
- If the profile copy has a local modification to a validation policy for a particular attribute (such as the shell), the values of that attribute are validated for all pushed records (only those records pushed to the profile copy will have their shell attribute validated).

Note

Validation policy will, by default, prevent the adding of UNIX UID 0 accounts.

14.3 Creating and Using User and Group Profiles

When a profile is created, only one copy exists. This copy must live in a profile manager. After the profile is created, records can be added one record at a time or by populating from a system file. Valid endpoints to populate a User Administration profile from are managed nodes, NIS Domains, and TMA endpoints. When records are added, the administrator can supply some of the values, and other values can be filled by default policy. Validation policy will also be used to check each of the attribute values to ensure they conform to the policy set for an individual profile.

14.3.1 Creating Profiles

Tivoli User Administration adds the UserProfile and GroupProfile profile types. A profile is created using the normal profile creation mechanisms of the Tivoli Framework.

Reminder

If you are managing TMA endpoints with User Administration, you must use Dataless profile managers to subscribe those endpoints.

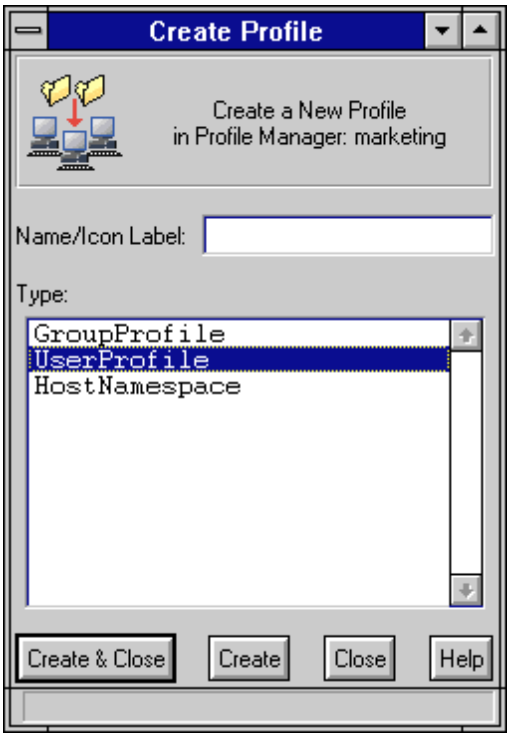


Figure 191. Creating a User Administration Profile

14.3.1.1 User Profile Properties

The properties (attributes) in use in a user record vary depending on the platform type being managed. If all the systems being managed were of one or two platform types, many customers use `wdelusrcat` to remove categories from dialogs and policies for unused platforms. Conversely, adding RACF support, for example, will add over 100 new attributes.

Figure 192 on page 466 shows a user profile containing a number of user records in which you can see some of the attributes, such as UNIX Login Name, Telephone, and so on.

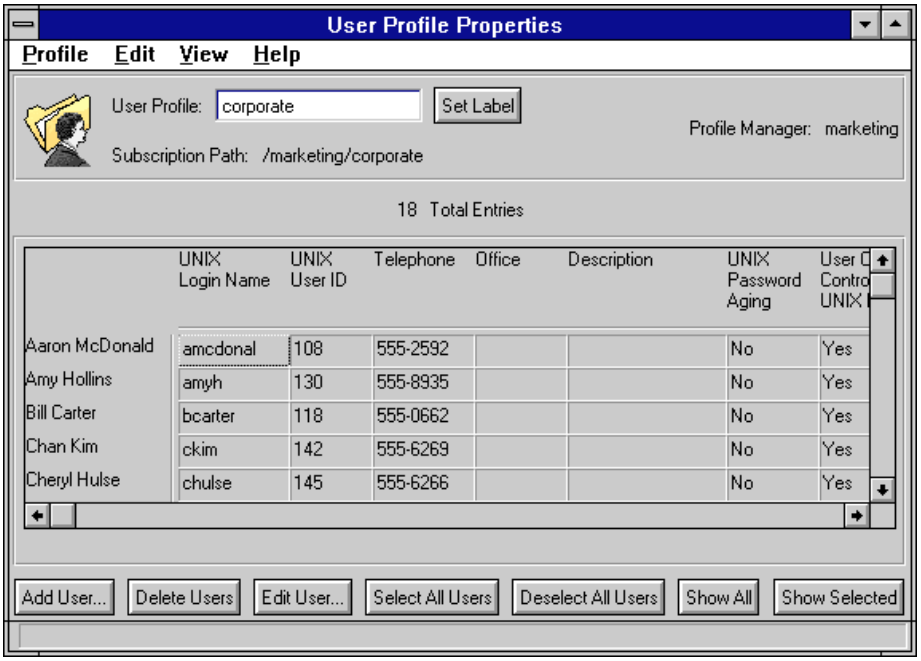


Figure 192. User Profile Properties Dialog

14.3.1.2 UID Allocation

In User Administration Version 3.1.4 and above, the UID allocation algorithm has been changed from earlier versions to improve performance. The default UID policy has `wallocid -l $MIN_UID -u $NOBODY`. You probably want to change each new profile to have valid lower and upper bounds for UIDs. Otherwise, the next created user will always be the last UID + 1.

14.3.2 Populating Profiles

Populating is the function of taking all the users or groups from the system account files and adding them as records to a User Administration user or group profile.

Features of populate are:

- Populate will fail validation for UIDs for root and nobody (unless you change the validation policy).

- Population does not run default policy.
- Populate cannot get NT or NetWare passwords. UNIX passwords are retrieved in their UNIX-encrypted form.

The `wpopusrs` command allows you to provide a list of user names as an input file. User Administration will then connect to the endpoint you designate as a source for population and retrieve the user data for each user in the list you provided.

14.3.3 Distributing Profiles

Once a profile is populated, it can be distributed to the endpoints. The endpoints available to receive profiles are determined by the profile manager and can consist of the following resource types:

- NisDomain
- Managed node (UNIX or Windows NT)
- TMA endpoints (UNIX from 3.6.1 onwards, NetWare, or Windows NT)
- Profile manager

TMA Note

UNIX TMA support for User Administration is a release 3.6.1 enhancement and only covers Tier 1 UNIX platforms.

When distributing a profile from the desktop, there are four options. These four options are:

- Next Level
- All Levels
- Preserve Local Modifications
- Exact Copy

The Next Level and All Levels distribution options dictate how far down the subscription chain the profile is distributed.

Next Level Moves the profile to the next object, which could be any one of the profile manager's subscribers including another profile manager.

When distributing a profile with Next Level to a profile manager, managed node, or NisDomain, the system user files of the ultimate destination are not modified. Instead, the copy of the profile in the Tivoli database on that endpoint is updated with the changes to the profile. Distributing to the next

level allows another administrator to make specialized changes at the managed node level. For example, if you have a set of users in the top level profile that are global users (accounts on every machine), and this user profile is distributed to the next level of a managed node, users can be added that have accounts on just this managed node.

Note

If you are at the managed node or NIS domain level (that is, you double-clicked on the managed node icon), and you do a distribute, it is the same as All Levels, and it will modify system files.

All Levels Distributes a copy of the profile to each object in the subscription chain. Whether the subscriber is a managed node, NisDomain, or a profile manager with subscribed managed nodes or TMA endpoints, then the system files are updated.

The Preserve Modifications and Exact Copy options dictate how modifications made in the Tivoli database for the local system and/or the system files are affected by the distribution of the profile.

Preserve Modifications Any changes made on the local copy of this profile (for example, in a managed node Tivoli database or a subscribing profile manager) are not overwritten.

For example, if a distributed user profile is edited at the managed node level, and the top-level (original) profile is distributed with Preserve Modifications, then the changes at the local copy will not be overwritten. This also applies to the system files.

When profiles are distributed with Preserve Local Modifications, only the records that have changed are actually pushed. For example, if a user profile with 200 users has been distributed at least once, then any subsequent distribution will only push out those user records that have changed, not all 200. If you wish to distribute all records, you would use Exact Copy.

Exact Copy Used to replace the local copy and system files with the contents of the distributed profile.

Distributing to all levels with the Exact Copy option is not as simple as it looks. Exact Copy distributes the profile to the endpoint. If this is a NisDomain or managed node with a local copy of user profiles, User Administration combines all the records of that profile type and applies them to the endpoint system files. Exact copy means that the system files will look exactly like all

profiles that have been distributed to it, removing any changes that have been made locally to the system files.

As no merging can take place on a TMA endpoint (as there is no data held in a Tivoli database on a TMA endpoint), an Exact Copy distribution directly to an endpoint from a dataless profile manager would result in the endpoint system files matching exactly just the data that was distributed. Note, however, that this behavior may vary from one platform type to another. For example, the OS/390 endpoint will not delete users as a result of an exact copy distribution of a user profile. The normal way to signal to the endpoint that a user should be deleted is to delete an existing user record from the Tivoli user profile. This explicit delete action will always result in the deletion of the user from the endpoint system files (with the exception of the previously mentioned reserved IDs, such as Administrator or UID 0 users).

Note

Using Exact Copy with user profile distribution never removes the UNIX root user or the Windows NT Administrator account. It will, however, remove all other accounts including other NT administrators and other UNIX system accounts if they are not defined in the profile.

If you defined a user profile containing a new user account named MyDog, with no other accounts in the profile, then distributed that profile alone using All Levels and Exact Copy options, then MyDog and UNIX root accounts would be the only accounts left on the target systems.

Password Note

The Exact Copy distribution option replaces all passwords in the system files.

14.3.3.1 Additional Command Line Distribution Method

Problems occur when users don't want to distribute with Exact Copy but do want all records to be distributed. For example, a user accidentally removes some entries from the /etc/group file, but the user does not want to distribute with the Exact Copy option because local modifications were made. Distribution with Preserve Modifications does nothing because none of the user profile records have been changed.

There are three options for distributing profiles:

- Merge and distribute changed records (Preserve Modifications).

- Merge and distribute all records.
- Make an exact copy and distribute all records (Exact Copy).

The second option above is a hidden command line option for the `wdistrib` command. This option distributes all the records in a profile without making an exact copy. The option is `over_opts`.

```
wdistrib -l over_opts <prof_name>
```

This option will replace all the passwords in the system file.

Merging Profiles Note

The Merge functionality just described does not work the same for TMA endpoints as it does for other resource types. Merging will NOT occur unless all the profiles for that TMA endpoint are in the same dataless profile manager it is subscribed to. An option to make merge work is to subscribe the TMA endpoint to a dataless profile manager and then subscribe that dataless profile manager to multiple data based profile manager(s) containing the Profile(s) for that TMA endpoint. This way, the dataless profile manager acts as a central location for merging the profiles together. With managed nodes, the merging takes place in the local Tivoli database of the managed node.

14.4 Deleting a User Profile

Deleting a user profile deletes the original profile and its records from the profile manager. It also (optionally) deletes copies of the profile from the profile manager's subscribers. It does not delete user account information from the subscribing endpoints. To delete the user account from the subscribing endpoints, you must delete the records from the profile, then distribute the empty profile to the endpoints. This deletes everything that was previously defined in the profile. There are a few exceptions. For example, Tivoli User Administration will not remove the following:

- UNIX root (ID 0) users.
- Any NIS directives in the configuration files on a UNIX endpoint.
- The Administrator user on a Windows NT.
- The ADMIN account on a NetWare endpoint.

Note

The UNIX root account is determined by the 0 UID. The Windows NT and NetWare accounts are keyed by their text names. The US English version of Tivoli User Administration will not remove an NT account named Administrator, but it will remove administrator accounts using other names including Administrator in other languages where the spelling is different.

So, to stop Tivoli User Administration from managing a particular user account, the process would be as follows:

1. Create a temporary profile.
2. Move the user to another profile - as this does not constitute a deletion from the original profile, a distribution of the original profile will not delete the user from the endpoint.
3. Delete the user from the new profile.
4. Delete the temporary profile. Be careful not to distribute this profile after deleting the user, or the user will be deleted from the target system.

14.5 File Versions

The use of file versioning for User Administration gives the system administrator a backup to go to in case of emergency. This is only available for UNIX managed nodes. Tivoli uses a revision control system (RCS) to update and maintain versions of system files. The directory Tivoli defines for file versioning is $DBDIR/file_versions. This directory contains a file hierarchy matching the system files that are modified. For example, a user profile distributed to a UNIX box will have a $DBDIR/file_versions/etc/RCS/passwd file. Likewise, if you distribute the same user profile to an NIS domain, you would find a $DBDIR/file_versions/var/yp/src/passwd file.

The location of the file versions directory is stored as an attribute in the Tivoli object database. To get or set the location of the file versions directory, you can use the following commands:

Get the OID for `fileioRef` object:

```
#idlattr -tg 0.0.0 fileioRef Object
 1234567890.1.322#TMF_FileIO#
```

Use `fileioRef` OID to get or set the directory name:

```
#idlattr -tg 1234567890.1.322#TMF_FileIO# versioning_area string
```

```
#idlattr -ts 1234567890.1.322#TMF_FileIO# versioning_area string
‘"newpath/file_versions"’
```

14.5.1 Extracting File Versions

Backups of configuration files modified by the distribution of user or group profiles are kept in $DBDIR/file_versions/system_directory/config_file. Each backup has a v extension. Table 19 on page 472 summarizes the RCS commands that can be used to view and manipulate versioned configuration files.

Table 19. .RCS File Version Commands Summary

Command	Purpose
`wci`	Checks in RCS revisions.
`wco`	Checks out RCS revisions.
`widen`t	Identifies files.
`wrcs`	Changes RCS file attributes.
`wrcsdiff`	Compares RCS revisions.
`wrcsmerge`	Merges RCS revisions.
`wrlog`	Prints log messages and other information about RCS files.

Refer to the *Tivoli Framework Reference Manual* for more information on the use of these commands.

14.6 User Profile Passwords

When user profiles are distributed, user passwords are only overwritten in the system files if the password itself is changed in the profile (or if `wpasswd -l`, `wdistrib over_opts`, or Exact Copy is run). Any other field in the user record can be changed, and the system password will not be overwritten.

Note

The password field is the only field in the user record that behaves this way. UID can optionally behave this way from Version 3.6.1.

For all the other fields, the following applies:

If anything in the user record is changed, the entire user record is distributed to the endpoint. For example, if the user’s shell in the system file is changed

(meaning it is different than in the Tivoli database), and the user's GCOS information is changed in the Tivoli database, the user's shell will be changed back to the value in Tivoli when the profile is distributed.

An additional field was added to the UserProfile resource from Version 2.5 onward. This attribute is named *profile_oneshots*. This attribute contains a list of users created with pre-expire password set to TRUE. Directly after distributing the profile for the first time after the user(s) have been created, User Administration traverses this list to reset the flag to FALSE. This ensures that the users' passwords will not be set to pre-expire (in the system files) if something is changed in the user record within User Administration.

14.7 User Profile Home Directories

A user record can have a local or remote home directory. Windows NT users do not have home directories defined within User Administration. However, documentation does exist on how to get User Administration to maintain Windows NT home directories using AEF (for an example see chapter 6 of the redbook *Getting Started with Tivoli User Administration*, SG24 2015).

The contents of the UNIX home directory for any user are by default copied from `/etc/Tivoli/Tivoli_Admin`. The default permissions of the home directory are 700. This can be changed with the *umask* default policy. (For example, a umask of 18 will provide a directory permission of 755; 18 is octal). The files in the directory will be 644. Changes to the umask default policy must be done before user records are added to the profile.

Note

Umask values before Version 3.1 were decimal, after 3.1, they are octal.

14.7.1 Local Home Directory

When the local option is used for a user record's home directory, then the home directory is created on the endpoint that will receive it. The home directory will only be created when it is distributed the first time to all levels.

14.7.2 Remote Home Directory

When the remote option is used for a user record's home directory, the home directory is created on the host that is exporting the file system if it is a Tivoli managed node or if it is mounted on the TMR server as root-writable. The remote home directories are created when the profile is distributed to the next

level for the first time. If the server is not a managed node, then the files will be created on the TMR server.

14.7.3 Problems with Creating Home Directories

If the system that is trying to create the home directory does not have root write access to the directory, the home directory creation will fail, but the user will still be created.

14.8 User Administration Notice Group

Tivoli User Administration adds a notice group. Use this to track operations during installations and upgrades and any changes to user profile records. The notice will include date and time stamps and the record that changed and the name of Tivoli administrator who performed the task.

14.9 NIS Domains

Network Information Service (NIS) is a profile endpoint. It allows you to manage the NIS maps on an NIS server through the user and group profiles as well as editing the other maps through Tivoli on the NIS icon. Tivoli User Administration does not support the creation of the NIS server or NIS replica servers. NIS must already be set up.

NIS may encounter problems, such as the following:

- Cannot distribute profile to NIS domain.
- Cannot make or push an NIS map.

Some of the best ways to look at problems when distributing to NIS domains are:

- Turn off automatic make and push of maps.
- Distribute to the next level, then distribute to all levels.
- Do the make and push by hand.
- Distribute to a fake NIS Domain.

NIS Note 1

Creating a user with a local home directory on an NIS endpoint adds the user to the NIS password map but does not create the home directory. Always use remotely mounted Home Directories for NIS, even if the NIS server is the file server.

NIS Note 2

If you ever distribute user/group profiles directly to the managed node where the NisDomain resides, you most likely will NOT be able to ever distribute the same type of profile to the NisDomain. You will get some error message about a conflict of target files.

14.9.1 NIS Default Policies

Table 20 contains the methods for setting custom default policies for NIS maps:

Table 20. NIS Default Policies

Method	Description
nis_def_map_makeflag	Sets the initial value of a map's makeflag.
nis_def_map_pushflag	Sets the initial value of a map's pushflag.
nis_def_map_makescript	Sets the initial value of a map's make script.
nis_def_map_pushscript	Sets the initial value of a map's push script.
nis_def_map_push_targets	Creates the initial list of push targets for a map.
nis_def_map_sourcefile	Sets the initial value of a map's source file.

The methods call shell scripts that you can modify to set a new default policy are shown here, and you can set default policy on a per-map basis.

14.9.2 NIS Validation Policies

Table 21 contains the methods for setting custom validation policies for NIS maps:

Table 21. NIS Validation Policies

Method	Description
nis_val_make_map	Validates the make operation.
nis_val_push_map	Validates the push operation.
nis_val_add_map	Validates a map before adding it to an NIS domain.
nis_val_remove_map	Validates a map before removing it from an NIS domain.
nis_val_map_source_name	Validates the name of the source file for the specified map.
nis_val_map_source_attrs	Validates the file system security of a map source file.
nis_val_domain_label	Validates the NIS domain label name against a naming policy.
nis_val_master_server	Validates the master server.
nis_val_delete_domain	Validates deletion of an NIS domain.

Tivoli User Administration NIS management provides a level of control over when notices are generated. The desired verbosity of an application (with respect to notices) is very user-specific, and, therefore, giving the user a means to control it is essential. To allow this to happen, User Administration NIS has a notice-level attribute set on each NIS domain object. For each operation the NIS application performs at a user's request, a fixed set of notification events occur. These events are listed in the table below. If the notice level of the domain is set higher than the notification event's level, then a notice is generated for that notice event. Otherwise, no notice is generated.

Table 22. NIS Notice Group Severity Levels

Severity	Level
DEFAULT	5
CRITICAL	9
ERROR	7
MAJOR	6
NORMAL	5

Severity	Level
INFO	4
NIT	2
DEBUG	1

The valid notice levels can range from 0 to 9. Level 0 notices provide the most verbose information.

While a notice event may generate a notice, the type of notice is determined by the notice event itself. Some events are debug events, which are useful only for debugging the application. Other notice events are useful when training a novice administrator. Some notice events reflect errors or critical conditions and, therefore, have a high level.

If the notice level set on a domain is 5, then all notice events with level 5 and above will generate a notice. It is strongly recommended that you leave the setting at 5 for normal use. Since error notices are generated at level 7, setting the notice-level above 7 will suppress error conditions.

Information logged in events with a notice level of 3 or less may not be meaningful to anyone except Tivoli Customer Support. For detailed listing of all the actions that can be logged for NIS Notifications, see the *Tivoli User Administration NIS Management Guide.*

14.9.3 Creating Fake NIS Domains

One useful method when working with NIS problems is creating fake NIS domains. The fake NIS domain will undergo the same methods that an NIS domain will. The only difference is that there is no make or push file at the end and there are no *yp* processes running. The following are the commands needed to create a fake NIS domain (`wcrtdomain` is documented in the *Tivoli User Administration NIS Management Guide*):

```
# ## Create a dummy NIS domain on host rainbow.
# mkdir /etc/yp/src
# cd /etc/yp/src
# head -50 /etc/passwd > passwd
# head -50 /etc/aliases > aliases
# mkdir nbNIStest
# wcrtdomain -c /etc/yp/src /etc/yp/src nbNIStest @rainbow @PolicyRegion:"Nancy's
Region"
# echo $?
0
# ###  list sourcefile for maps
# wlsmaps  -s @nbNIStest
aliases /etc/yp/src/aliases
passwd  /etc/yp/src/passwd
# ###  create new profile manager
# wcrtprfmgr @PolicyRegion:"Nancy's Region" nbPM
200205.1.741#TMF_CCMS::ProfileManager#  nbPM
# echo $?
# ###  subscriber to ProfileManager
# wsub @ProfileManager:nbPM @NisDomain:nbNIStest
# echo $?
0
# ###  create UserProfile
# wcrtprf @ProfileManager:nbPM UserProfile "nbUSER NIStest"
200205.1.743#UserProfile#       nbUSER NIStest
# echo $?
0
# ### create some new users
# wcrtusr -h /is/user2 -t MOUNTED -S overlook:/export/is/user2 -u 107
@UserProfile:"nbUSER NIStest" user2
# wcrtusr -h /engr/user3 -t MOUNTED -S crescent:/export/engr/user3 -u 201
@UserProfile:"nbUSER NIStest" user3
# wcrtusr -h /staff/user4 -t MOUNTED -S chisos:/export/staff/user4 -u 874
@UserProfile:"nbUSER NIStest" user4
# wcrtusr -h /mgr/user5 -t MOUNTED -S rainbow:/export/mgr/user5 -u 305 @UserProfile:
"nbUSER NIStest" user5
# ###  Where
# ###  -h xxxxxx  corresponds to Add User dialog field for Home Directory Path...
# ###  -t xxxxxx                                          Home Directory Type
# ###  -S xxxxxx                                          Server Path...
# ### distribute user profile to <fake> NIS domain
# wdistrib -m -l maintain @UserProfile:"nbUSER NIStest" @NisDomain:nbNIStest
# echo $?
0
```

There are various uses for fake NIS domains besides testing NIS problems and implementations. One example is the migration of Tivoli user data from one TMR to another. In an all UNIX environment, you could distribute your user data to a fake NIS domain and then use populate in the second TMR to bring all the data into the second TMR.

14.9.4 General Approach to User Administration Customization

Most installations will benefit from customization of User Administration. To avoid problems caused by the modifications, use the following tips:

- Change one policy at a time and test it.
- Do not change everything at once.
- Test changes in an isolated or lab environment before trying to implement them in production.
- If you make too many changes without incremental testing, debugging User Administration problems can become extremely complicated
- If you experience a problem, you can run tests on a profile that has not been modified to see if it is the customizations that are the cause of the problem
- Avoid changing the default user profile (TivoliDefaultUserProfile). This is the profile used when you create a new profile from scratch. Instead, you should modify an existing profile and then use that as a source for cloning. If you encounter problems with a modified profile, you can then return to the default profile for testing. Also, the default profile may be modified by Tivoli updates.

14.10 User Administration Data

There are several databases that Tivoli User Administration uses within the Tivoli management object database to keep up with user and group records. They are:

- UserNameDB
- UID
- GroupNameDB
- GID
- LocatorDatabase

These databases have references into user and group profiles and are used with the UserLocator application and with commands, such as `wlsids` and `wlsnams`, that are commonly used in default and validation policies. These are part of the Tivoli Name Registry; so, TMR commands, such as `wlookup` and `wregister`, can be used here.

These databases are examples of what may move to RIM in future implementations.

14.11 User Administration Methods

Table 23 lists the method calls that can be observed with `odstat` and `wtrace` when trying to trace problems with User Administration:

Table 23. User Management Methods

Method	Path to binary
`UserProfile_synchronize`	/TME/USERMANAGEMENT/umbo_skel1
`UserProfile_verify`	/TME/USERMANAGEMENT/umbo_skel1
`change_password`	/TME/USERMANAGEMENT/umbo_skel1
`um_discover`	/TME/USERMANAGEMENT/umbo_skel1
`um_discover_ext`	/TME/USERMANAGEMENT/umbo_skel1
`um_gen_strlist`	/TME/USERMANAGEMENT/umbo_skel1
`um_runcmd`	/TME/USERMANAGEMENT/umbo_skel1
`um_set_login`	/TME/USERMANAGEMENT/umbo_skel1
`um_update`	/TME/USERMANAGEMENT/umbo_skel1

14.12 Troubleshooting User Administration

This section lists some of the most common problems you may come across and how to deal with them.

14.12.1 Code Level Consistency

The TMR must have one consistent level of User Administration installed. For example, if you are going to Version 3.6, then all nodes in that TMR must be brought up to 3.6. If you have Tier 2 systems that are not available at the same level, you must stay with the Tier 2 level of Tivoli User Administration for the entire TMR in which the Tier 2 systems reside.

14.12.2 Populate Considerations

The following considerations apply to User Administration record population:

- If the subscriber node is a managed node, Tivoli User Administration must be installed on that node before you can populate a profile from it. If the subscriber is a TMA endpoint, then the User Administration gateway must be installed on the endpoint gateway before you can populate a profile from an endpoint on that gateway.

- The populate will fail validation for UNIX UIDs for root and nobody (by default).
- Populate cannot get NT or NetWare passwords.
- If you encounter a problem with populate from a UNIX box, run the platform command to check `/etc/password` and `/etc/group`. The command will be something like `pwck` (Solaris) or `pwdchk` (AIX) and `grpck` (Solaris and AIX). This will provide a syntax sanity check.

14.12.3 Distribute Considerations

If you are having trouble getting a User Administration profile distributed, the following tips may help:

- Verify that Tivoli User Administration is installed on the managed nodes you are managing, and User Administration gateway is installed on your endpoint gateways if you are managing TMA endpoints.
- Remember that Distribute with Exact Copy will erase system accounts such as nobody, adm, bin, lp, and any non-root uid 0 account unless they are explicitly in a user profile. Also, Exact Copy will overwrite all passwords.
- Pushing to Windows NT Backup Domain Controllers (BDC) can cause problems and is not recommended. The PDC should always be used as the target. This is more of a problem with versions of User Administration prior to 3.6.
- The administrator doing the distribution will need root write access to home directory file systems in order to create home directories.

14.12.4 Interregion Considerations

If you are having problems distributing to another TMR, verify that the systems in that TMR are running the same version of Tivoli User Administration, and that a TMR resource update has taken place recently. Some resource updates are critical to User Administration working across TMR boundaries. Which resources to exchange really depends on your specific Tivoli environment. If all User Administrator profiles are kept on a central TMR, then you do NOT want to exchange resources, such as the Name database and UID database.

14.12.5 Modifying Records

The following is a list of things to check that may lead you to the source of problems encountered when making changes to records or will help Tivoli support representatives to assist you:

- Was the record populated or added?
- Are you in a top-level profile?
- What type, local or remote, is the home directory?
- Are there AEF customizations?
  - Extensive AEF customizations can be extremely complex. Without detailed AEF and User Administration experience, you are unlikely to be able to do much to resolve AEF customization issues.
  - Try testing without the AEF customizations to isolate if the problem is in base Tivoli or in your AEF customizations.
  - Be aware that upgrading User Administration from 2.x to 3.x will disable any AEF changes. It will be necessary to reapply the changes following the upgrade. This will probably be the same for any upgrade including a future upgrade from 3.x to 4.x.
  - There is little Tivoli can do to prevent syntax errors in stand-alone shell scripts. Users should run `sh -n` on their scripts before they import them as AEF actions. Tivoli just runs them as they are typed in; it is not designed to ensure that actions are syntactically correct.
- Check for any policy customizations. If you have a custom policy, and you have also changed TivoliDefaultUserProfile, then create a new profile and execute the following commands and try same thing again. If it then works, then the problem is the Policy customizations you previously did:

  ```
  wsetdefpol UNIX DISABLE <profile_name>
  wsetdefpol UNIX ENABLE <profile_name>
  wsetdefpol NT DISABLE <profile_name>
  wsetdefpol NT ENABLE <profile_name>
  wsetdefpol NW DISABLE <profile_name>
  wsetdefpol NW ENABLE <profile_name>
  ```

  Disabling and re-enabling default policy in this way deletes any customer policy definitions. Note that this only applies to the platform types provided in the base product (UNIX, NetWare, and Windows NT).
- What are the administrator's TMR and policy region roles? Some commands will need TMR roles.
- You can use `wtailnotif` to track notices.
- Where is the endpoint? Check if it is in an interconnected TMR.
- Look at the oservlog for any indications of a problem.

14.12.6 Other Troubleshooting Hints and Tips

This is a list of miscellaneous tips worth noting:

- If you receive a failure in the GUI, trying the CLI may make it work or provide more information about the failure. This is particularly true if the GUI was customized.
- CLI commands to manipulate User Administration profiles do not work on interconnected TMRs; they must be executed locally within the TMR. Either use the GUI, or login to the remote TMR system and execute the command locally.
- Certain endpoints, such as Windows NT, can put a lot of detail about errors in the notice group, and this should be checked as well as other logs.
- Implementing password aging on AIX 3.2.5 can be difficult because it uses per-system password aging rather than per-user password aging.
- The number of users in a profile can have a major impact on the performance of distributions or GUI operations. Prior to 3.6, the recommendation was to keep profiles down to 500 users or less. You should avoid having more than 1,500-2,000 users in one 3.6 profile. More than 2000 users in a profile can cause massive slow-downs because of the way Tivoli transactions work. Performance on distributions to UNIX depends upon the size of the /etc/passwd and /etc/aliases files as well as the size of the profile being distributed. What actually happens is a `wpopulate` is done to a virtual profile, then the distributed profile is merged into the virtual profile, then the virtual profile is sent down to the system files.
- If you change the common login name, by definition, it will change all other login names.
- A few other tricks to try:
  - Unsubscribe and delete pushed copies. Then re-subscribe and try again.
  - Make Exact Copy to next level, then Preserve Local Modifications to system files.
  - Push a new profile with one dummy user to test distribution and populate.
  - Always distribute a newly populated profile before making changes.

14.13 The wpasswd Command

The wpasswd command is used to modify a user's password in user profiles. It is not meant to replace the UNIX passwd command. However, you may want your users to make use of wpasswd without retraining them to use a new command. Replacing passwd with wpasswd is not recommended because this would require a very clever wrapper to be created. The man page for wpasswd gives a few reasons:

> The wpasswd command is not intended to be a replacement for the UNIX passwd command. wpasswd will not run if the oserv process is not running on the local host or if the Tivoli server is down. You may need to change UNIX passwords when a host is in "single-user mode" or other times when the oserv process is down.

If you wanted to write a wrapper, it would need to respond correctly to the following situations:

- The script being called by a Tivoli administrator changing a regular user's password by a regular user changing his own password or by root.
- The oserv being down.
- The system being in single-user mode.
- The TMR connection being down if connected to a central TMR.
- The handling of local accounts not managed by User Administration (such as root and kroot).
- The handling of expired passwords and login issues.
- Being invoked by a Tivoli administrator who does not have admin role for the user profiles.
- The case where the UNIX password and the Tivoli password were so different that the comparison with the old password did not work in all cases.
- User interrupts (^C).

Do not ever change a password in a copy of a profile. You cannot unchange it (only change it back if you knew the original password), and wpasswd won't work for a user, as the password the user gives will not match. The copy's password value will stay unless you do a distribution with Exact Copy or over_opts and reset everybody's password. The other option, in this case, would be to unsubscribe the subscriber and delete the offending profile copy.

Chapter 15. Security Management

While Tivoli User Administration manages the creation and authentication of users, Tivoli Security Management manages the access control those users will encounter. Security Management is transparent to the end users. They continue to use the normal login and access methods to use IT resources. Security Management works with the access control mechanisms of supported endpoints to determine the access rights users have to defined resources. For example, for Windows NT, Tivoli Security Management manipulates the Security Account Manager (SAM) database to ensure users have the desired access rights and permissions.

The special case for Security Management is the UNIX platforms. For UNIX, a new security product is included called the Tivoli Access Control Facility (TACF). This intercepts resource calls from users and applications and applies access control mechanisms defined in security management profiles.

Troubleshooting Tivoli Security Management is going to be largely a question of troubleshooting either TACF or someone else's security system (such as the Windows NT SAM or MVS RACF).

Security Management provides four types of record that can be defined within a security profile.

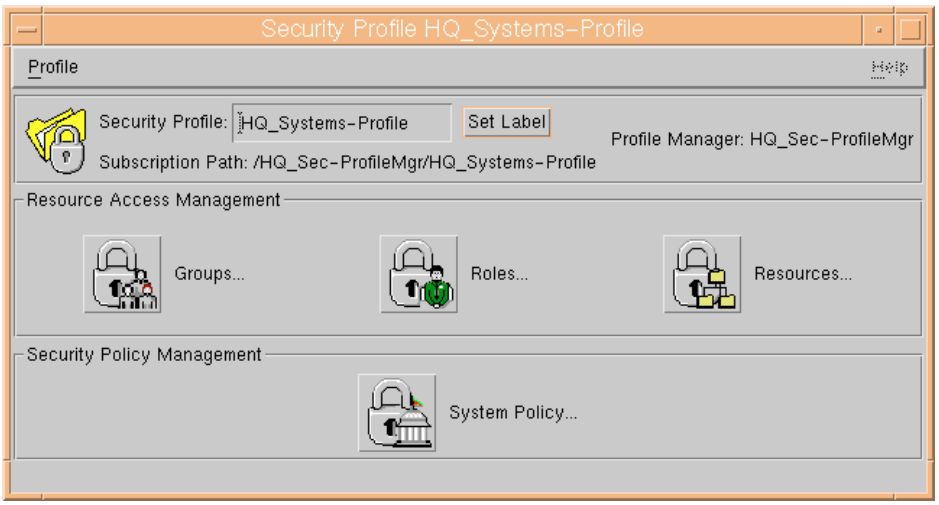


Figure 193. Security Profile

Access is managed in a hierarchical fashion. Users are collected together into groups that fit into the company organizational structure. That group is then assigned roles that correspond to the job functions members of that group will perform. The roles define the access capabilities for sets of resources. In addition, there is a set of attributes that can be set for whole systems to implement system policy. These include items, such as password controls (maximum and minimum lengths, and so on). The four security record types: Groups, roles, resources and system policy can all exist in the same security profile.

The security profile contains definitions of all groups, resources, roles, and system-wide policies, each of which is stored in its own type of record. Records are stored in a system-independent format that allows profile records to be distributed across an enterprise that contains many different system types.

As important as describing what Security Management is, it is equally important to give examples of what it is not. There should be no misconceptions of what the product is designed to provide:

- It does not alter the security of the Tivoli framework.

  This means it is not a framework security product. It is possible to use Security Management to enhance framework security, for example, by managing access controls for file systems where the Tivoli applications reside or restricting connectivity to systems running Tivoli software.

- It does not manage Web server, database, or application security.

  Security Management can manipulate the access control to the files in use in these programs; or in UNIX, we can say which processes can have access. However, each of these examples usually maintain their own security information (access controls, users, and so on). Security Management is not provided with interfaces to these security mechanisms (at the time of writing). With the 3.6 release, Security Management is more easily extended using AEF in a similar fashion to Tivoli User Administration making the addition of new endpoints (such as a database) a possibility.

Security Management, as a tool, helps security administrators to enforce a consistent security policy enterprise-wide by providing a means to centrally define and distribute these policies to their enterprise systems' local security databases.

Policies defined with Security Management specify the allowable access to system-level resources (files, accounts, printers, etc.) on UNIX and Windows

NT managed nodes and TMA endpoints. Only Windows NT machines, which belong to a Windows NT Domain, are supported; stand-alone NT Workstations are not. Additional platforms, as with Tivoli User Administration, are supported through add-on modules (examples include OS/390 Security Server and OS/400).

Security Management, on all platforms besides UNIX, uses the native security system. For example, on Windows NT, distributing a profile modifies the access control lists on the real NT resources defined in the profile. This means they must exist, or an error will be raised. Population varies by platform but will generally not discover all resources on the endpoint.

When distributing/populating from an NT endpoint, NT global groups are mapped to security group records, and NT local groups are mapped to role records. This imposes some restrictions, foremost among them being that *group* records should only be distributed to Primary Domain Controller (PDC) machines.

System policy maps to the policies accessible through the Windows NT User Manager. Distributed to a PDC, this affects the domain's policy. On other machines, it has only local effect. Distribution of system policy to a BDC will fail as the copy of its account database is read-only.

Editing records directly within Windows NT is discouraged as we need to keep the information between Windows NT and Security Management synchronized. Security Management manages an NT domain user account for each global group managed with a group record. This user account named *tme_groupname* is for internal use only. Do not edit this user account.

Security Management implemented on UNIX will be discussed in 15.6, "Tivoli Access Control Facility" on page 492.

15.1 Tivoli Security Management Installation

Tivoli Security Management installs like any other Tivoli Framework application either through the Desktop Installation mechanism, the command line with `winstall`, or using Software Installation Services (SIS). While it will work hand-in-hand with User Administration, it is not a prerequisite. There are a few points regarding installation to note about this product:

- There are five components to Tivoli Security Management:
  - Tivoli Security Management (for the TMR server and any managed node that will use Security Management).

    - Tivoli Security Management for gateways (for each managed node that will serve as a gateway for TMA endpoints).
    - Tivoli Access Control Facility (TACF, for UNIX managed nodes and UNIX TMA endpoints only).
    - TACF Installation Utilities (for each managed node from which you will install TACF onto UNIX TMA endpoints).
    - Security Tasks, Events, and Monitors.
- TACF can only be installed on supported UNIX endpoints. This is a subset of UNIX systems and levels supported by the Framework. Check the release notes for details.
- When installing TACF, the administrator chooses the location where the binaries will be stored. This will not necessarily be the Tivoli `$BINDIR`.
- If you subsequently add products, such as Tivoli Distributed Monitoring or TEC, you will need to reinstall the monitors and events component to make the security ones available, therefore, plan on installing these products before Security Management.
- The ownership rights of TACF and many of its processes belong to the user ID *tmesec*. The User IDs of TACF Administrators field allows you to define additional TACF administrators. Avoid setting *root* as a TACF administrator as this will give *root* the ability to kill TACF processes and, thereby, circumvent the security policies.
- To remove Tivoli Security Management products from a managed node or the TMR server use the `wuninst` command. To remove TACF from UNIX endpoints use `wuninsttacf`.

TACF may be installed anywhere on the system; it does not have to reside under the Tivoli directory structure hierarchy. There is only one restriction regarding location; it must be installed to a local file system. For convenience, a link, `/usr/seos`, is created to the root of the TACF installation. Additionally, links to the TACF binaries are created in `$BINDIR/bin`; so, you should always be able to find these on a system.

When TACF is installed, a local user, `tmesec`, is created. This is the account used by Security Management to administer TACF. This account is as powerful in TACF as `root` is in UNIX; so, the same means used to secure `root` against misuse should also be applied to `tmesec`.

Because TACF must constantly reside and run on a system, it does not fit into the on-demand download model of a standard TMA endpoint. Thus, TACF must be installed like a product, even on an endpoint. Since the standard

installation mechanisms will not install software on a TMA endpoint, the TACF Endpoint Installation Utilities must be used to install TACF. The TACF Endpoint Installation Utilities must be installed on the gateway serving the endpoint. The install utility, `winsttacf`, utilizes a reduced version of Tivoli Software Distribution to install TACF on the endpoint.

TACF is very sensitive to operating system versions. Installing on unsupported versions could result in a variety of problems. This issue must also be considered before upgrading the operating system on a machine running TACF. Refer to the *Security Management Release Notes* for specifics on supported versions.

15.1.1 Security Notice Group

Tivoli Security Management adds a Security notice group. Use this to track operations during installations and upgrades and any changes to security profile records. The notice will include date and time stamps, the details of any record change that took place (including what was changed), and the name of the Tivoli administrator who performed the task.

15.2 TMR and Policy Region Roles

The new Tivoli administrator roles for Tivoli Security Management are as follows:

- `security_admin`

  This assigns authority to administer access to system-level resources. Here are some of the functions that an administrator with this role can perform:

  - Create and modify security group, security role, and security resource records.
  - Create and modify System Policy records.
  - Distribute security profiles.

- `security_auditor`

  This assigns authority to audit the use of system resources and to control which security events are logged. This role is required to execute the Audit Report Generator task.

- `security_operator`

  This assigns authority to view all security resource data.

Existing authorization roles still apply to some functions related to managing security profiles. For example, assigning security profiles as managed

resources in a policy region requires a senior authorization role in the policy region.

Full details of the authorization roles required to perform each function are documented in the *Tivoli Security Management User's Guide*.

New Roles in 3.6.1

The 3.6.1 release introduces a whole new set of so-called Fine-grain roles. These allow a finer distinction of a security administrator's capabilities. For example, an administrator could be allowed to create a role but not a group, and so on. See the release notes for *Tivoli Security Management 3.6.1* (or manuals for subsequent releases) for details.

15.3 Populating and Distributing Profiles

This section details some considerations for populating and distributing Tivoli security profiles.

15.3.1 Populating Records

Note that the population of records will vary depending on the endpoint type. Population from UNIX will be from TACF records and not from any native UNIX access control mechanism. This means UNIX files, groups, and other system specific resources are not searched. UNIX groups can be added to the TACF database by executing the `UxImport` command, or by using the `Synchronize TACF/UNIX Users and Groups` task.

For Windows NT, population will create a security group record for each NT global group, and a role record will be created for each NT local group. Be aware that as a group record corresponds to an NT global group, and as you cannot define a global group on an NT workstation, that you cannot distribute a group record to an NT workstation. With the exception of SYSTEM resource records, NT resource records cannot be populated from the profile. All other resource records must be populated at the record level. If this action was permitted, every file, directory, and registry entry on the endpoint would be its own resource record. The Resource record of type SYSTEM is given the name *User Rights*. NT System Account Policy is used to populate the Security System Policy.

15.3.2 Security Profile

Security Management is a profile based application. Security policies are represented by Security Profiles that must be distributed to the endpoint to be

put into effect. The security model is a three-tiered, role-based model described in some detail in Tivoli Security redbooks (such as SG24-2021, SG24-5101, and SG24-5339) as well as the product manuals.

You should note that it makes little sense to have multiple system policy records in a single profile; the last one pushed to an endpoint wins.

Groups, Roles, and Resources connect together to define the security policy. These connections can span across Security Profiles, but all Security Profiles involved will need to be distributed to the endpoint.

The *Tivoli Security Management Users Guide* has a detailed description of all resource types provided by default

15.3.3 Distribution Options

As with Tivoli User Administration, we have options to distribute to all or next level and using preserve modifications or exact copy.

It is unlikely that you will want to use all levels and exact copy. This option will replace any existing definitions and remove any that are not in the profile being distributed. Check the *Tivoli Security Management Users Guide* sections “Setting Distribution Defaults” and “Distributing Security Profiles”, for limitations with Exact Copy distribution, as not all records will necessarily be overwritten.

15.4 Auditing

Most of the controls Security Management puts in place can be audited with settings for access attempts, such as failures only, success only, and no auditing. The default auditing level is for failures only.

TACF has a warning mode for auditing. You can define the controls you want to put in place, but TACF will allow all accesses and generate a warning if the controls would have resulted in failed access. This is a very useful feature for testing new installations.

15.5 Security Tasks

Tivoli Security Management provides a set of tasks that allow administrators to run security jobs. These tasks and jobs behave in the same way as all other tasks and jobs in allowing the execution characteristics to be changed, for example:

- On which systems the tasks/jobs will run.
- Where the output is displayed.
- Whether to run serially, in parallel, or staged in groups.

The following tasks are provided:

- Tivoli Security Audit Report tasks:
  - Endpoint Audit Report
  - TME Security Database Report
- TACF tasks:
  - Add/Remove TACF Auditor/Administrator
  - Backup TACF Database
  - Disable Concurrent Logins
  - Enable/Disable TACF Trace
  - Extract TACF Trusted Programs
  - Restore TACF Database
  - Show TACF Auditors/Administrators
  - Start TACF Servers
  - Stop TACF Servers
  - Synchronize TACF/UNIX Users & Groups
  - TACF root Compliance Report
  - TACF Database Maintenance
  - TACF Run-time Statistics Report
  - TACF/UNIX Integrity Report
  - Terminate TACF User

These tasks are all described in the *Tivoli Security Management User's Guide*. Note that during installation, Security Management does not assume it should place the task libraries in an existing Policy Region. It places them in policy regions created specifically for the Security Management install. Whether you leave them there or not depends on your own administrative requirements.

15.6 Tivoli Access Control Facility

TACF is Tivoli's name for the SeOS technology licensed from Memco. It is a critical part of Security Management on UNIX systems providing a non-intrusive, yet powerful, extension beyond the 9-bit/uid/gid UNIX native security. By non-intrusive, we mean the kernel does not need to be re-compiled to install TACF, nor are the objects being secured modified in any way. When TACF is shut down or de-installed, resource access remains as it was under UNIX before TACF.

There are three main pieces to TACF itself:

- Kernel extension
- Daemons
- Security database

The kernel extension replaces a subset of the standard system calls with security checking routines of its own. If the security check fails, failure is immediately returned. If the check succeeds, the original system call is invoked and the results returned. Note that TACF must allow access then UNIX. Both must be successful for access to be granted; TACF does not grant a way around UNIX security.

There are at least three and as many as five daemons that comprise an active TACF instance. The three most important, which are always present, provide the core services of TACF, and are:

seosd Handles authorization checks
seagent Performs actions on behalf of seosd
seoswd Watches over protected resources and other daemons

These daemons cannot be killed; only an authorized TACF administrator can shut them down. The other two daemons are used to provide optional extra functionality:

selogrd Forwards TACF audit records to a central collection point. TACF TEC integration relies on selogrd.
serevu Enforces login failure policy.

TACF also maintains its own security database containing all the security information for TACF: The groups, users, and resource ACLs. The seosd daemon will refer to the database whenever a security check is requested. Additionally, items, such as password history, expiration, and last login, are stored here.

15.6.1 TACF Architecture

The TACF process places hooks in system services that need to be protected; therefore, control is passed to TACF before the service is performed. When a user requests a system service that is protected by TACF, the process accesses the object record and checks to see if the user is allowed access to the object. Access rules and policies are defined in the TACF database that is installed on each UNIX managed node. The TACF database contains information about the resources, groups, roles, and system policies that are applicable for that node. The basic structure of the access rule is an Access Control List (ACL) that is attached to the protected

record. An ACL is a list of users or groups that are allowed access to a particular object along with the access authority for each user or group.

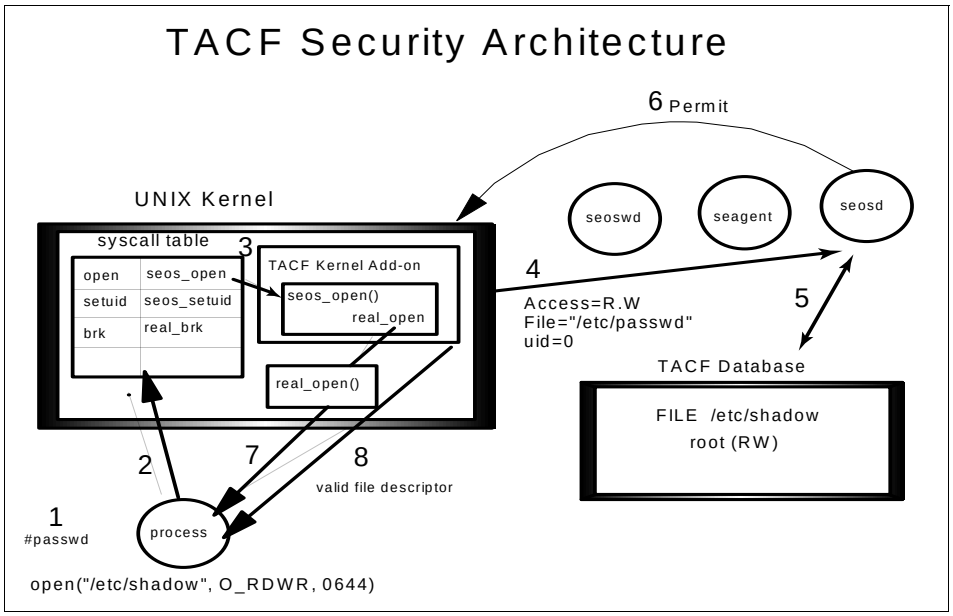


Figure 194. TACF Security Architecture

In this:

1. A process makes a system call, such as root invoking bin/passwd.
2. Kernel routine for system call (open) is looked up in UNIX kernel table.
3. Kernel looks up address of routine which services the (open) request (seos_open) and invokes it.
4. TACF version of system call checks with seosd if requested access to object is allowed for this user.
5. seosd checks the rules in the TACF security database.
6. seosd returns Permit or Deny.
7. If Permit, the system-supplied kernel routine is invoked, and results are passed back to calling process.
8. If Deny, return failure (EPERM) to the calling process.

TACF has administrator and auditor roles just as Security Management does.

Administrators have the authority to create, modify, and delete access rules defined in the database as well as the ability to shut down TACF (only root can start TACF). Auditors may view the rules and the audit logs. These roles are completely distinct and separate from the similar Tivoli authorization

roles: `security_admin` and `security_auditor`. TACF administrators and auditors can be created at install time or later through the *Add / Remove Auditor / Administrator* task.

15.6.2 TACF Utilities

Several utilities are included with TACF, mostly for administration:

secons For controlling the daemons (stopping them, enabling tracing, obtaining information).

selang A CLI to the security database. With `selang`, an administrator can check on the rules created by Security Management or add their own.

Additional utilities exist that replace or supplement existing UNIX password or `su` utilities, such as `sepass` and `sesu`. Note, that as of 3.6.1, `sepass` is not integrated in any way with the Tivoli User Administration `wpasswd` command. This integration is planned for a future release.

15.6.3 TACF User Mapping

Unlike UNIX, TACF permissions are granted based on who you log in as, not based on your effective/real/saved UIDs. So su and setuid/setgid executables have no effect on the access rights granted/denied through TACF although they still affect UNIX rights.

UNIX system calls return numeric IDs instead of names to the caller. TACF allows the definition of rules using names instead of IDs; so, TACF installation automatically generates and installs a *Lookaside Database* on all TACF endpoints. This lookaside database provides the resolution of TACF ID to name. The installation process also creates the user *tmesec*, and the groups *tme_aud* and *tme_sec* on each UNIX managed node and endpoint if they do not already exist. As the */etc/passwd* and */etc/group* files are not likely to be the same on all nodes, the ID assigned to *tmesec*, *tme_aud* and *tme_sec* will most probably be unique. If the same ID is preferred on all nodes, create the user and groups manually or with a product, such as Tivoli User Administration, before installing TACF. If the ID of the user or group is changed at a later stage, to prevent conflicts within NIS maps, as an example, you must synchronize all TACF managed node and endpoint databases using the job *Create/Update TACF Lookaside Database.*

15.6.4 Distributing and Populating with TACF

Distributing a security profile to UNIX causes the rules in the TACF security database to be updated. Security profiles can only be populated from records already in the TACF database:

- Roles cannot be populated on UNIX.
- Group records are mapped into TACF groups. TACF groups are not UNIX groups and need not contain the same members.
- Roles have no physical mapping.

TACF resources protect UNIX resources, yet the UNIX resource need not even exist. The protection will come into effect when the UNIX resource is created. TACF also allows a test or *warning* mode. If set, the rules on an object are not enforced. However, should an access take place that normally the rules would deny, an audit record is generated. This is exposed in Security Management through the Warning attribute on Resources.

15.6.5 TACF Command Line

Under normal circumstances, it will not be necessary to interface directly with TACF from the command line. Instead, Tivoli Security Management adds a number of commands to the TME environment, such as `wcrtsec`. These are what you would use for normal command line activities. Using the command line, however, as opposed to w commands, such as `wcrtsec`, might be more useful in problem determination where you are getting unexpected access or denial of access to resources.

The user ID `tmesec` owns the TACF database and most of the associated files. It is defined as a TACF user with attributes that allow it to add, modify, and delete resources and accessors. Other user IDs, for example `root`, may have the authority to start up the command line environment but could be restricted to only viewing resources and accessors.

TACF commands are available to run in two environments:

- TACF command shell

  These commands interface directly with the objects and resources defined in the TACF Database.

- UNIX environment

  These commands interface with the UNIX operating system allowing you to add, modify, and delete UNIX users and groups and modify file permissions from within the TACF command line interface.

To access the TACF command line:

1. Login as `tmesec`.
2. Set up the Tivoli environment:

   ```
   . /etc/Tivoli/setup_env.sh
   ```

3. Enter: `selang`

   This will give you the TACF prompt:

   ```
   TACF>
   ```

You can enter `help` to get a list of commands and `environment unix` or `environment tacf` to switch between the UNIX and TACF command sets.

Refer to the *Tivoli Security Management Reference Manual for TACF* or the redbook SG24-2021 *Managing Access from Desktop to Datacenter* for more details on the commands available. This command interface can be used to determine what TACF believes is the status of resources, user groups, and other features.

15.6.6 TACF Initialization File

The TACF initialization file is called `seos.ini` and will be located in the top-level directory specified during TACF installation, such as:

```
/usr/local/Tivoli/TACF/seos.ini
```

The `/usr/seos` link can also be used to find this file:

```
/usr/seos/seos.ini
```

The initialization file `seos.ini` defines the environment in which TACF will start. You can find a full description in the *Tivoli Security Management Reference Manual for TACF*, but here is some of the information that it contains. Replace the directory name with the directory of your installation:

- Installation directory:

  ```
  Default : /usr/local/Tivoli/TACF
  ```

- Directory of the seadm rules and other configuration files:

  ```
  Default : /usr/local/Tivoli/TACF/data
  ```

- TACF database:

  ```
  Default : /usr/local/Tivoli/TACF/seosdb
  ```

- TACF log files (including the audit log):

  ```
  Default : /usr/local/Tivoli/TACF/log/seosd.audit
            /usr/local/Tivoli/TACF/log/seosd.error
            /usr/local/Tivoli/TACF/log/seosd.trace
  ```

- Start up file for TACF Daemons:

  `Default : /usr/local/Tivoli/TACF/rc.SeOS.base`

- Symbolic link to TACF directory:

  `Default : /usr/seos -> /usr/local/Tivoli/TACF`

- Options for TACF trace messages.
- Options for resolving IP addresses to host names.

15.7 Tips and Troubleshooting

Unfortunately, although the concepts behind Security Management are simple, the details are many and complicated. Controlling access to a resource can sometimes have much further reaching consequences than expected. Also, because of its power, Security Management will often take the blame for many failures with which it has no connection.

However, there is usually a fairly good source of information for resolving these problems - the audit logs. Since access failures are generally audited, there will often be an audit record detailing the failure: Who attempted the access, how the attempt was made, and so on. This is important because what the users think they are doing is not necessarily the whole story.

With TACF, there is a tracing tool. The `seosd` daemon can provide a trace of its activities if requested though `secons` (`-t+` to turn on, `-t-` to turn off). This trace is written to `/usr/seos/log/seosd.trace` and shows what authorization checks `seosd` is performing as well as what commands are being executed against the database.

15.7.1 TACF Trace

TACF tracing can be controlled through the initialization file seos.ini. Figure 195 on page 499 shows the relevant lines from seos.ini:

```
; The destination of trace messages. There are several valid options:
; file       - Trace messages are written to a file.
; none       - Trace messages are not written at all.
; file,stop  - Trace messages are written to a file and automatically
;              disabled once the daemon has passed it's initialization.
; Default Value: file,stop
trace_to = file


; Location of the file to which trace messages are written
; Default Value: /usr/seos/log/seosd.trace
trace_file = /usr/local/Tivoli/TACF/log/seosd.trace


; Trace file type. This can be one of the following values:
; text   - The trace messages file is a text file.
; binary - The trace messages file is a binary file which
;          reduces the size required by this file.
; Default Value: text
trace_file_type = text

; Location of a trace filtering file.
; Default value: /usr/local/Tivoli/TACF/etc/trcfilter.
init
trace_filter = /usr/local/Tivoli/TACF/etc/trcfilter.in
it

; Free space to leave in filesystem when trace_to file is allowed.
; When less than this space is free, trace will be disabled (in KB)
; Default Value: 1024 (1MB)
trace_space_saver = 5120
```

Figure 195. Tracing TACF Options in seos.ini File

To start a full trace, set:

```
trace_to = file
```

Or from user ID tmesec or a user with the ADMIN attribute, enter:

```
secons -t+
```

To watch the trace as resources are accessed:

```
tail -f /usr/local/Tivoli/TACF/log/seosd.trace
```

or

```
secons -tv
```

To check the status of the trace:

```
secons -ts
```

The trace is automatically disabled if free space in the file system gets too low. Figure 196 shows an example of a TACF trace where a Telnet access attempt was denied:

```
INET    : P=764  , from 146.84.32.173:2749  port 23
INET    > Result: 'D' 146.84.32.173->23, [0,-1], stg=406
               Why?    HOSTNET entry in TCP service ACL
FORK    : P=764   U=0    G=-1   Child=7255            Pgmd
FILE    : P=7255  (/usr/sbin/inetd) U=0    (D=40000003 I=d
FILE    > P=7255  (/usr/sbin/inetd) U=0    /etc/passwd   S
EXEC    : P=7255  U=0    G=0    (D=40000008 I=2978  ) Pgm3
EXECARGS: 'telnetd'
GPEERNAM: P=7255 , ADDR=146.84.32.173, N=-1839980371
```

Figure 196. Example of TACF Trace - TCP Access (Telnet) Denied

Figure 197 shows the trace entries for a write to file access denied:

```
EXEC    : P=7285  U=0    G=103  (D=40000008 I=12671 ) Pgmi
EXECARGS: 'vi testfile'
FILE    : P=7285  (/usr/bin/vi) U=0    (D=40000005 I=1751e
FILE    > (/usr/bin/vi) Result: 'P' [stage=59 gstag=59 AC]
               Why?    Resource universal access check
FILE    : P=7285  (/usr/bin/vi) U=0    (D=40000005 I=1751e
FILE    > (/usr/bin/vi) Result: 'D' [stage=69 gstag=0  AC]
               Why?    No rule granting access to resource
```

Figure 197. Example of TACF Trace - Write Access to File Denied

In the above examples, `Result: 'P'` means Permitted action, and `Result: 'D'` means Denied action.

15.7.2 Distribute and Populate Failures

The endpoint methods for populate/distribute are `security_discover` and `security_update`. These do not call other methods; so, it is not really possible to determine the point of failure from `odstat` or `wtrace`. On TACF, however, it is possible to use both the audit log and tracing to determine what commands were executed and what failed. On NT, some information may be available in the Application section of the Event Log, but it is usually of little value except for identifying known problems. Of course, it is possible that the error occurred before any records were processed.

For distribute/populate to work on UNIX, TACF must be running at the time. Otherwise, Security Management will not be able to access the security

database. It is important to remember that TACF rules apply to Tivoli just as they do to other applications. Since the endpoint methods run as `tmesec` and `tmesec` is protected with a SURROGATE rule, `oserv` or `lcfd` would normally be prevented from starting the method. These processes must be granted special access allowing them to circumvent the rule. However, since this privilege cannot be granted without some sort of authentication, `oserv` or `lcfd` must be reliably identified. There are cases where this identification cannot be made. Should should this happen, the method will fail with `s=9` because TACF sends a `SIGKILL` to the method daemon.

Restarting `oserv` or `lcfd` on the node should correct this problem. Figure 198 shows this case in a seosd trace:

```
Example: oserv recognized properly (seosd.trace)

FORK : P=29769 U=0 G=60001 Child=29804 Pgm:/data/Tivoli/bin/solaris2/bin/oserv
SGID > P=29804 U=0 (RG=0 EG=0 SG=0 ) to (RG=60001 EG=60001 SG=60001) ()
BYPASS
SETGRPS : P=29804 to
LOGIN : P=29804 User=tmesec Terminal=0.0.0.0
LOGIN > Result: 'P' [stage=11 gstag=54 rv=0] ACEEH=45
EXEC : P=29804 U=101 G=60001 (D=80001f I=239233)
Pgm:/data/Tivoli/bin/solaris2/TME/SECURITYE/SecEpt

Example: oserv not recognized properly (seosd.trace and odstat)

FORK : P=17321 U=0 G=60001 Child=29740 Pgm:
SGID > P=29740 U=0 (RG=1 EG=1 SG=1 ) to (RG=60001 EG=60001 SG=60001) ()
BYPASS
SETGRPS : P=29740 to
SUID : P=29740 U=0 (R=0 E=0 S=0 ) to USER.tmesec (R=101 E=101 S=101 ) D=00000000
I=0
ACTION : TACF killed P=29740
SUID > Result: 'D' [stage=69 gstag=0 ACEEH=42 rv=0]
Why? No rule granting access to resource

* 6424 O+hdq done 90 0 15:50:13 e=12 2119115577.1.584#SECURITYP::SecurityProfile#
  populate_ext
* 6425 O+hdq 1-6424 done 0 0 15:50:13 s=9 2119115577.1.348#TMF_ManagedNode::Managed
  _Node# security_discover
```

Figure 198. Example of TACF Trace - oserv Not Authenticated

15.7.3 Access Problems

Possibly the most frequent problems are those that revolve around access being granted or denied unexpectedly on an object. These are much more common on UNIX than on NT.

On UNIX, these almost always boil down to who TACF identifies as the user. Often, this is because the lookaside database is out of synchronization. This

can be checked by running `sebuildla -U` (or the *Show TACF Lookaside Database* task) and seeing if the user is listed in the output (and that the UID listed is correct). If not, the user will never be identified correctly by seosd and will always be given default access. To correct this, the lookaside database must be rebuilt using `sebuildla -u` (or the *Create/Update TACF Lookaside Database* task).

15.7.3.1 Example: Lookaside Database Consistency Problem

On our neptune system, we can check who the system believes the currently logged on user to be:

```
neptune$ id
uid=102(tuser1) gid=1(other)
```

Next, we use the TACF command `sewhoami` to confirm that TACF thinks the currently logged on user is the same:

```
neptune$ /usr/seos/bin/sewhoami
tuser1
```

Now we will attempt a resource access for which `tuser1` is supposed to have the necessary rights:

```
neptune$ cat /tmp/example
cat: cannot open /tmp/example
```

To check if `tuser1` is supposed to have access, we start `selang` and check the file `ACL` in the `TACF` database:

```
neptune$selang
TACF selang v2.02 (2.01) - TACF command line interpreter
Copyright (c) 1996-1997 Tivoli Systems Inc.
Portions of Tivoli-Access-Control-Facility
Copyright (c) by MEMCO Software Ltd.
TACF> sf /tmp/example (sf is short for showfile)
(localhost)
Data for FILE `/tmp/example'
------------------------------------------------------------
Defaccess : None
Acls :
Accessor Access
tuser1 (USER ) R, W, X, Cre, Del, Chown, Chmod, Utime, Sec, Rename
Audit mode : Failure
Owner : tmesec
Create time : 01-Apr-1998 14:26
Update time : 01-Apr-1998 14:27
Updated by : kirwin
TACF> quit
```

So, tuser1 is supposed to have all rights to /tmp/example. Therefore, we'll check the audit log for all records from the date and time we tried the access attempt:

```
neptune$seaudit -a -sd 01-APR-1998 -st 14:29
01 Apr 98 14:29 D FILE tuser1 Read 69 3 /tmp/example /usr/bin/cat
neptune.dev.tivoli.com
Total Records Displayed 1
```

The audit log entry shows a Denial of a FILE access request (D FILE) for user tuser1 performing a read. So we use sebuildla -U to check if our user is correctly specified in the lookaside database:

```
neptune$sebuildla -U | grep tuser1
```

tuser1 is not there, and the database will need to be re-synchronized. A second possibility for a similar problem is that the user had used su. TACF grants access based on logon, not euid/ruid. Running the sewhoami command will display the user as identified by TACF; so, this problem would have been spotted much earlier in the above example as shown here:

```
neptune$id
uid=15704(kirwin) gid=40(Development)
neptune$su
Password:
neptune# id
uid=0(root) gid=1(other)
neptune#sewhoami
kirwin
```

Even though we su to root, TACF still treats us as the original user (kirwin).

Tasks will run as if the user designated in the task had logged in, and access will be given accordingly. No Access always wins when there is an access rights conflict. Otherwise, access rights are cumulative amongst all Roles to which a user belongs.

15.7.4 System Policy Problems

With TACF, the implementation of some of the System Policy features might not work as expected:

- Password Quality

  Password quality checks are only done if the sepass utility is used to change passwords. As of 3.6, sepass is not integrated with Tivoli User Administration's wpasswd.

- Grace Logins and Password Expiration

Grace logins and password expiration is handled with the segrace utility. To be effective, all login profiles must execute segrace.

- Lockout Policy

  Lockout policy requires use of the serevu daemon. While this will be configured properly when distributing a System Policy record, with Security Management 3.6, serevu will not revoke accounts unless started by a TACF administrator su'ed to *root*. Note that serevu, unlike seosd, seagent and seoswd, is not automatically protected against kill.

Note

SunOS does not log login failures, and Solaris only logs failures if five failures occur during an attempt. Since TACF relies on operating system logs to detect login failures, lockout policy is necessarily restricted on these interpreter types.

For most system policies, it is necessary to have the user defined to TACF in order for it to work. This can be accomplished by using the *Synchronize TACF/UNIX Users & Groups* task.

15.7.5 Miscellaneous Considerations

This is somewhat of a catch-all. There are many problems which aren't obviously connected to the access control policies but appear to be traceable to the distribution of a security profile (the converse is also true - it may look like security profile rules are responsible, but they are not). These are in particular the problems that rely on the audit logs. Often, something is indirectly accessing a resource in a way that was overlooked.

One example of such a problem is illustrated by CONNECT resources and ftp.

Attempting to use ftp from a remote machine, a user can connect, but upon attempting to list directory contents, an error occurs. Why should this be? CONNECT resources restrict connections going out - not in. The audit logs reveal that the machine running TACF is opening a connection to the remote machine (because passive mode client is the default, the server must open the data channel). If the user does not have access to the CONNECT resource, the open of the data channel back to the remote host will be denied.

Another example is `/etc/passwd`, `/etc/shadow`, and `login`. Every user must have at least read access to these files, or they cannot log in (since `login` can't validate their password otherwise).

If the audit logs leave doubt as to the cause, try using the tracing in TACF. While generally no more useful than the audit logs (as auditing is always occurring, but tracing must be turned on), it will show all accesses - not just those being audited.

Sometimes it's not clear whether Security Management is causing a problem. At least on UNIX, shutting down TACF can help determine this, though usually the audit log is proof enough.

Note

Restricting access to system files or to resource types without fully considering the implications can be very dangerous, going as far as to making the system unusable. Use of the warning attribute when on UNIX is strongly encouraged in such cases.

Refer to the *Tivoli Security Management Reference Manual for TACF* or to the redbook *Managing Access from Desktop to Datacenter: Introducing TME 10 Security Management*, SG24-2021, for more information on TACF tracing.

15.8 Integrating with Tivoli Enterprise Console

The *Tivoli Security Management Users Guide* includes an overview of auditing, installing, and configuring the TEC log file adapter for TACF. It also discusses treating managed nodes as endpoints (as the log file adapter for TACF can only be installed on endpoints) and the limitations of the TEC log file adapter for TACF. The same Users Guide also contains details of all the event classes.

15.9 TACF Security Monitors

The *Tivoli Security Management Users Guide* provides information required when using the `waddmon` command to add monitors. The TACF Security Monitors monitoring collection provides three monitoring sources: TACF files to monitor trace and audit files created by TACF, TACF daemons, and TACF file systems to monitor the amount of free disk space in the file systems in that the TACF audit and where trace files are created.

Monitors provided with Tivoli Security Management are:

- TACF Server
- TACF Watchdog Server
- TACF Audit Log Routing Server
- Audit File Size
- Audit Log File Size
- Audit Log File Free Space

15.10 Migrating SeOS Access Control to TACF

The *Tivoli Security Management Users Guide* contains an appendix for assistance with migrating SeOS installations to TACF.

Chapter 16. Enterprise Console

Tivoli Enterprise Console (TEC) is a rule-based event management application that uses a central server to process incoming events. These events are generated by adapters running on hosts throughout a TMR. These events can apply to Networks, systems, databases and applications. The TEC acts as a central collection point for alarms and events from a variety of sources, including those of Tivoli applications, Tivoli partner applications, customer applications, Network management platforms, and relational database systems. The central TEC server machine (a managed node) serves as the location of the event server.

Note

There can be only one event sever in the TMR, and it should be separate from the TMR server. A dedicated machine is best.

The TEC consists of the following Components:

- Central Event Server
- Distributed Event Consoles
- Central Event RDBMS (using RIM)
- Distributed TEC Gateway
- Distributed Event Adapters

16.1 TEC Central Event Server

When an event occurs in the enterprise, it travels to the event adapter on a host that translates it into a syntax that the event server can understand. The adapter is either a TME or non-TME adapter. This distinction defines how the event will reach the event server. A TME event adapter sends events through the object request broker (oserv) on that managed node, to the oserv on the TEC server, and then to the *tec_server* process. A non-TME event adapter will send events from the adapter directly to the *tec_reception* process on the event server.

The TEC server consists of five daemons that run on the event server host and process an event. The tec_server process is the controller for the other four processes. When an event comes into the tec_reception, the event is cached in memory and written to the reception log in the database. The event is then sent to the *tec_rule* engine where it is processed. Based on the rules

defined for that event class, the tec_rule process defines the actions that need to be taken and passes the actions to the tec_dispatch process to be executed. The *tec_dispatch* process writes to the event repository in the database and updates the event consoles. If there are programs or tasks to be run for this event, the *tec_task* process is contacted. Figure 199 shows this relationship.

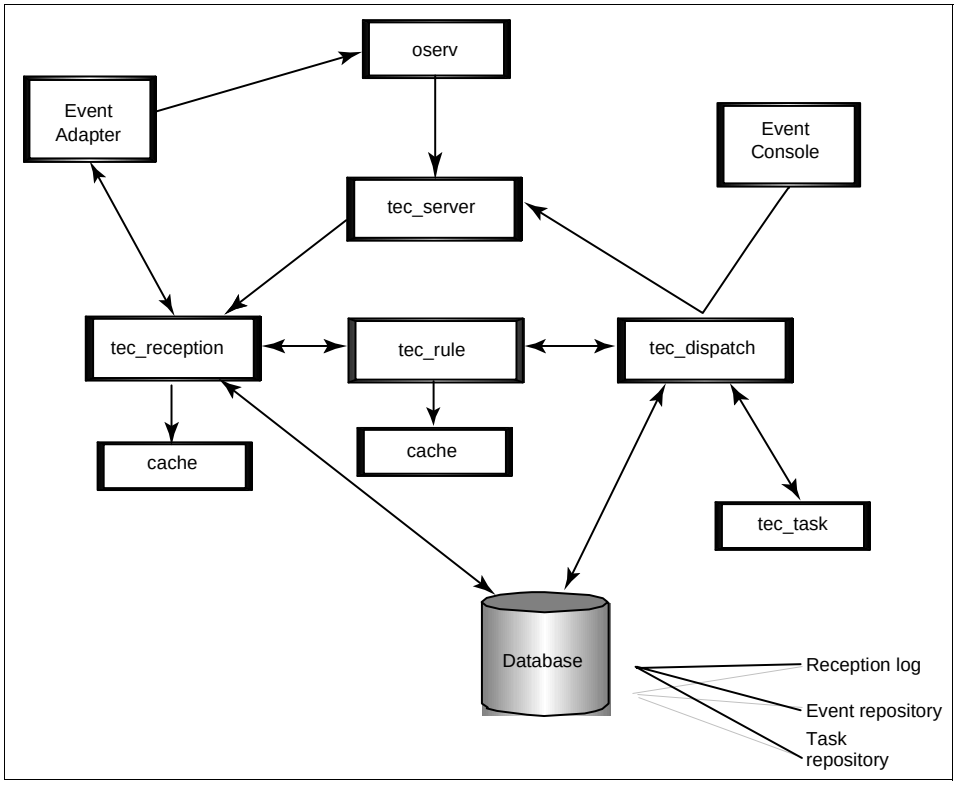


Figure 199. TEC Daemon Relationship Diagram

The event server provides a centralized location for the management of events. The event server performs several functions:

- Logging
- Applying rules
- Correlating events
- Responding automatically to events
- Updating the event console
- Processing input from the event console
- Delaying responses to events

- Escalating events

The event information is placed into a reception log, processed by a Prolog rule engine, and moved into an event repository.

There is a command called `wtdumprl` to show the contents of the reception log. This command is very helpful when an application sends an event that does not appear on the Event Console. The reception log normally tells us what went wrong. For example, Figure 201 on page 510 shows a class (nt_CpuPrcCpuTime) that is not defined and is generating a parse error. This is the error you will see if you do not import the necessary baroc file into the current rule base and load it. Figure 202 on page 511 shows an example of an event containing a slot (probe) not being defined in the event baroc definition (universal_countstr).

The engine, or rule base, can recognize events and perform tasks to determine if it can respond to or modify the event automatically. The server creates an entry in a relational database for each incoming event.

The next three screens show some examples of a `wtdumprl` output:

```
1~370~0~877419610(Oct 21  1997)
### EVENT ###
TEC_Start;source=TEC;msg="TEC Event Server initialized";END

### END EVENT ###
PROCESSED

1~371~0~878766447(Nov 05  1997)
### EVENT ###
TEC_Start;source=TEC;msg="TEC Event Server initialized";END

### END EVENT ###
PROCESSED
```

Figure 200. Output from wtdumprl - Example 1

```
1~79~0~875384794(Sep 27  1997)
### EVENT ###
nt_CpuPrcCpuTime;
source='SENTRY';
sub_source='CPU';
severity='FATAL';
origin='9.164.194.178';
sub_origin='tp760dom';
hostname='tp760dom';
adapter_host='tp760dom';
distrib_admin='Root_tp760dom-region';
response_level='warning';
probe='PrcCpuTime';
probe_arg='0';
tmr='1675599057';
dispatcher='1';
prev_value='10.9793';
value='35.31';
effective_value='35.31';
collection='NT_Processor';
info='';
monitor='Percent Processor Time';
units='(percent)';
relation='Greater than';
relation_delta='';
msg='Sentry CPU/Percent Processor Time on host tp760dom 09/27/97 08:25:07

Status: >>> warning <<<

Percent Processor Time (0)  Greater than 20
(Previous: 10.9793 (percent) Current: 35.31 Effective: 35.31)
';
END

### END EVENT ###
PARSING_FAILED~'Line 1: Class nt_CpuPrcCpuTime undefined'

1~84~0~875385040(Sep 27  1997)
### EVENT ###
TEC_Stop;source=TEC;msg="TEC Event Server shut down";END

### END EVENT ###
PROCESSED

1~85~0~875463018(Sep 28  1997)
### EVENT ###
TEC_Start;source=TEC;msg="TEC Event Server initialized";END

### END EVENT ###
```

Figure 201. Output from wtdumprl - Example 2 - Parse Error

```
1~86~0~875463731(Sep 28  1997)
### EVENT ###
universal_countstr;
source='SENTRY';
sub_source='Sentry_Profile';
severity='CRITICAL';
origin='9.164.194.178';
sub_origin='tp760dom';
hostname='tp760dom';
adapter_host='tp760dom';
distrib_admin='Root_tp760dom-region';
response_level='critical';
probe='countstr';
probe_arg='test, g:\temp\test, ';
tmr='1675599057';
dispatcher='1';
prev_value='';
value='1';
effective_value='1';
collection='Universal';
info='';
monitor='File pattern matches';
units='';
relation='Equal to';
relation_delta='';
msg='Sentry Sentry_Profile/File pattern matches on host tp760dom 09/28/97 06:21:42

Status: >>> critical <<<

File pattern matches (test, g:\temp\test, )  Equal to 1
(Previous:   Current: 1 Effective: 1)
';
END

### END EVENT ###
PARSING_FAILED~'Line 11: Slot probe not defined in class'
```

Figure 202. Output from wtdumprl - Example 3 - Slot Not Defined

16.2 Distributed Event Console

TEC uses distributed event adapters to collect information, a central event server to process the information, and a distributed event console to present the information to the operators. Event consoles display event messages appropriate for specific administrators based on their responsibilities. Consoles can be configured to display groups of events from the available adapters. This allows for easy separation and assignment of system maintenance tasks to the appropriate administrators. Users can have independent or shared views of events.

16.3 Central Event RDBMS Through RIM

TEC 3.6 uses RIM to access and store event data on relational databases. Both the reception log and the event repository are RDBMS tables accessed through the RIM. RIM gives us the ability to choose and connect to a wide choice of RDBMS. Currently, TEC and RIM support Oracle, Sybase, DB2, and MS/SQL.

Note

At the time of writing, DB2 was supported only on UNIX and MS/SQL only on NT. Informix support was planned for the 3.6.1 release.

See Chapter 9, “RDBMS Interface Module (RIM)” on page 313 for more information about RIM.

16.4 Distributed TEC Gateway

If you need to run event adapters on TMA endpoints, then you will need the TEC gateway process running on each endpoint gateway those TMA endpoints will be connected to. This is done by installing the Adapter Configuration Facility (ACF) on each endpoint gateway server.

16.5 Distributed Event Adapters

Event information is collected by event adapters and sent to the event server. Adapters are usually small daemon processes that run on the client host. The host machines can be managed or non-managed nodes. When an event adapter detects an event generated by a source, it formats a message and sends it to the event server.

Figure 203 shows an outline of the TEC event data flow:

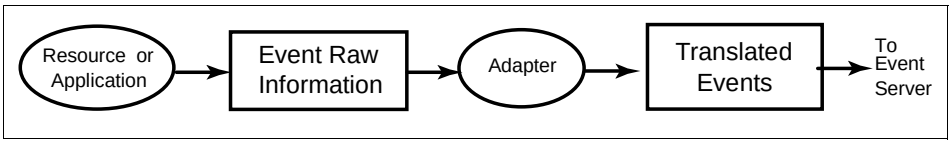


Figure 203. TEC Event Data Flow

16.5.1 How Event Adapters Send Events to the Event Server

This section describes the way event adapters get events to the event server from different types of machines.

16.5.1.1 From a TMA Endpoint

Adapters running on endpoints send their events to a TEC gateway process running on the endpoint gateway that the endpoint is connected to. The TEC gateway, in turn, bundles events and forwards them on to the event server through the oserv.

16.5.1.2 From a Managed Node

Adapters running on managed nodes send their events through the oserv.

16.5.1.3 From a Non-Tivoli Node

Adapters running on non-Tivoli nodes send their events directly to the event server using an IP socket. These non-secure adapters must configure the host name or IP address of the event server in the ServerLocation field in the configuration file.

Note

Tivoli versions of the Logfile, OS/2, Windows NT Event, and SNMP adapters in TEC 3.6 only run on TMA endpoints. Any managed nodes that you want to run these adapters on must also be setup as TMA endpoints. A single node can be configured as both a managed node and a TMA endpoint. If you do this, be careful when distributing ACF profiles, as both the managed node label and the TMA endpoint label will show up in the subscriber list. Make sure you use the TMA endpoint label.

Non-Tivoli versions of these adapters are available and could be run on these managed nodes.

For a detailed description of how these event adapters work, refer to the *Tivoli Enterprise Console Adapters Guide.*

16.6 TEC Installation

This section covers the following topics:

- Pre-installation steps
- Install Enterprise Server
- Install Enterprise Console

- Create new rule bases

16.6.1 Pre-Installation Steps

Before starting an installation, you should:

- Ensure you are using a RIM-supported RDBMS.
- Confirm the RPC Portmapper is running (UNIX only) by issuing the command `rpcinfo -p`.

There are a number of considerations when installing databases for use with TEC. In Appendix C, “RDBMS Install Examples” on page 679, we have detailed an example of how to install Oracle 7.3.2.1 for use with TEC. Much of the installation will apply to different UNIX types and for other applications that use RIM. The rest of this chapter assumes the RDBMS is installed. We use Oracle for examples here and have included information about usage of other RDBMSs. An excellent source for RDBMS installation and setup documentation is the *TME10 Inventory 3.2: New Features and Database Support* Redbook, SG24-2135.

16.6.2 Install Enterprise Server

Install the TEC Enterprise Server according to the installation instructions in the *Tivoli Enterprise Console Users Guide.*

Depending on the Tivoli environment, the Database Server, RIM Host, and TMR server may reside on different systems. Many different combinations are possible. For more information about this see Chapter 9, “RDBMS Interface Module (RIM)” on page 313. Based on this information, the code for the TEC Server and the TEC Console may reside on different systems as well. The console code gets installed on the TMR server or a managed node, and is accessed through an administrator desktop. Rule builder and server code go together on the same machine.

Tivoli User Administration Implementation

If you use, or intend to use, Tivoli User Administration in your TMR for performance reasons, Tivoli recommends you DO NOT install the TEC Server on the TMR Server.

Using the regular install GUI or SIS, the dialog box in Figure 204 pops up. This populates the RIM values for the RIM object named `tec`.

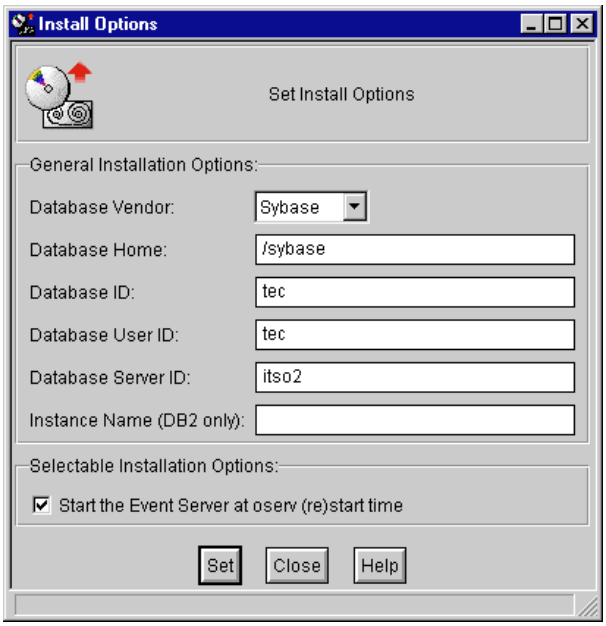


Figure 204. Dialog for TEC Install

The dialog has some field names that do not match with the names used in the Oracle environment or for the RIM host. The fields Database Vendor and Database Home do not need to be discussed because the field name says clearly what should go in here. But the Database ID field and the Database Server ID field need some explanation. To show the relation between these fields and RIM data, it is necessary to get the RIM host settings and to have in mind what you have defined in the tnsnames.ora file. The RIM host settings can be displayed with the `wgetrim rimname` or changed with the `wsetrim` command. Figure 205 shows our RIM settings for the RIM object named `tec`:

```
This screen shows our tec RIM host settings
root@rh0255f:~etc# wgetrim tec
RIM Host:       rh0255f
RDBMS User:     tec
RDBMS Vendor:   Oracle
Database ID:    D1
Database Home:  /oracle/home/app/oracle/product/7.3.2
Server ID:      D1
Instance Home:
```

Figure 205. RIM Host Settings - wgetrim

16.6.3 Install Enterprise Console and TEC Adapters

This section contains considerations for installing the Enterprise Console and TEC Adapters.

16.6.3.1 Enterprise Console

Each Tivoli administrator can have their own individual enterprise console. The installation of the console and adapters should be a trivial task.

16.6.3.2 TEC Adapters

If you are installing TEC adapters on TMA endpoints, you must use the Adapter Configuration Facility (ACF) to configure and install these adapters. Be sure and install ACF on each of the endpoint gateways that will have endpoint adapters communicating through them. Since endpoints can move from one gateway to another, it is a good practice to install ACF on all endpoint gateways if you are going to use endpoint adapters.

When you install ACF on the endpoint gateway, the TEC gateway process is installed there as well.

> **Note**
>
> Tivoli versions of the Logfile, OS/2, NT Event, and SNMP Adapters only run on TMA endpoints. If you were previously running one of these adapters in secure mode and NOT using ACF to configure them, then you will have to manually enter the configuration information into a 3.6 ACF profile.

16.6.4 Troubleshooting Installation

If you are using SIS to install these products, then refer to Chapter 4, “Tivoli Software Installation Service” on page 83 to see how to review your installation log files.

If you are still using the conventional installation process, there are multiple output files in the /tmp directory on UNIX or in the %DBDIR%/tmp directory on Windows NT that the installation process writes. These files are located on the TMR server system not on the machine hosting the RIM host or the TEC server. For each Tivoli product installed, these files should be saved and moved to another location. These files can have one of the following extensions:

.sinst TEC server installation files.

.cinstall Tivoli client and application installation files.

.output Lists normal output during product installation on the TEC server and managed nodes containing event adapters.

.error Lists problems that occurred during product installation on the TEC server and managed nodes containing event adapters.

Some TEC files in /tmp begin with `tec` and are easy to identify. If an installation fails, the last few lines of the .error file will contain troubleshooting information.

16.7 Troubleshooting TEC

There are a number of commands that are very helpful when trying to understand what is going on with TEC:

`wtdbclear`	Clear the reception log and event repository.
`wtdumprl`	Display the events in the reception log.
`wtdumper`	Display the events in the event repository.
`wstartesvr`	Start the enterprise server from command line.
`wstopesvr`	Stop the enterprise server from command line.
`wtdbstat`	Show the RDBMS database server status.
`wstatesvr`	Show the TEC server status.
`wgetrim tec`	Display the RIM parameters.
`wsetrim`	Change RIM parameters.
`wlseg -f`	Lists event groups and their filters.
`wlssrc`	Lists the available event sources.

There will be an example for each command later in this chapter where the command will be used to look at a problem.

16.7.1 TEC Server Troubleshooting

When one of the TEC processes fails, the master process (tec_server) attempts to restart it. If it cannot restart the process, tec_server brings them all down. For each of the five processes, error conditions are written to a file in /tmp on the TEC server system with that process name. If any of these have trouble initializing, error logs are written at the end of the file. These are the file names and process names for each process:

- /tmp/tec_master tec_server
- /tmp/tec_reception tec_reception
- /tmp/tec_rule tec_rule
- /tmp/tec_dispatch tec_dispatch
- /tmp/tec_task tec_task

These paths are specified in a file called .tec_diag_config in the $BINDIR/TME/TEC directory on the TEC server system as shown in Figure 206.

```
## tec_master
##############

tec_master Highest_level              error
tec_master Master                     error        /tmp/tec_master
tec_master Master_Msg                 error        /tmp/tec_master
tec_master Master_Synchro             error        /tmp/tec_master
tec_master Master_Exec                error        /tmp/tec_master

# low level modules
tec_master Exit_Msg                   error        /tmp/tec_master
tec_master Tec_Baroc                  error        /tmp/tec_master
tec_master Timer                      error        /tmp/tec_master

# IPC modules
tec_master Ipc                        error        /tmp/tec_master
tec_master Ipc_Accept                 error        /tmp/tec_master
tec_master Ipc_Dsend                  error        /tmp/tec_master
tec_master Ipc_Alive                  error        /tmp/tec_master
tec_master Ipc_Server                 error        /tmp/tec_master

# Pool modules
tec_master Pool                       error        /tmp/tec_master
tec_master Pool_Master                error        /tmp/tec_master

# Msg modules
tec_master Msg                        error        /tmp/tec_master
tec_master Msg_HI                     error        /tmp/tec_master
tec_master Msg_DP                     error        /tmp/tec_master
tec_master Msg_EP                     error        /tmp/tec_master
tec_master Msg_TP                     error        /tmp/tec_master
tec_master Msg_OK                     error        /tmp/tec_master
tec_master Msg_GO                     error        /tmp/tec_master
tec_master Msg_AC                     error        /tmp/tec_master
tec_master Msg_NA                     error        /tmp/tec_master
tec_master Msg_CA                     error        /tmp/tec_master
tec_master Msg_CC                     error        /tmp/tec_master
tec_master Msg_NE                     error        /tmp/tec_master
tec_master Msg_RR                     error        /tmp/tec_master
```

Figure 206. Example Extracted from .tec_diag_config File

If the event server will not start, there are several common reasons to check for:

- The database engine is not running.
- The database is full.
- The portmapper is not running.

Figure 207 shows an example from the RIM trace log while trying to start the event server while the DB2 database engine was not running.

```
T00030200 [Tue Nov 17 17:12:03 1998] Connection ID: 0, Operation: val_connect:SQL
Connect, DB Call: val_connect:SQLConnect
        DB2 Error Code: -1032  SQLState:08001[IBM][CLI Driver] SQL1032N  No star
t database manager command was issued.  SQLSTATE=57019
```

Figure 207. Example RIM Trace Log - Database Not Running

The dialog box in Figure 208 was displayed on the Tivoli Desktop when trying to start the event server from the GUI when the DB2 database engine was not running.

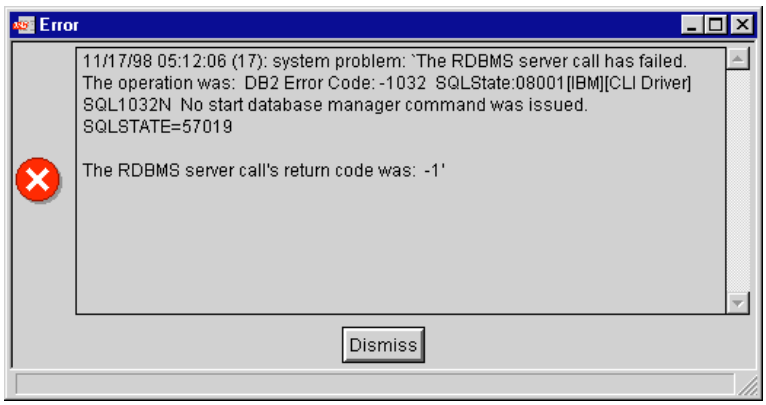


Figure 208. Event Server - Database Engine Not Running Error Dialog

When the database engine is running, but the event server can not connect to the database, the messages in Figure 209 would be generated when trying to start TEC:

```
[root@itso2]/> wstartesvr
The TME 10 Enterprise Console Server is initializing...
Mon Nov  9 12:39:54 CST 1998 (17): system problem: 'Could not connect to RDBMS s
erver to access database tec.'
```

Figure 209. Event Server Cannot Connect to Database Error - wstartesvr

When the database is full, the RDBMS might be running, but the event server will not start. There are several ways to check this:

- running the command `wstatesvr`

- running the command `wtdbstat`

In the case where the database is full, the commands will generate the outputs shown in Figure 210:

```
% wstatesvr
The Tivoli/Enterprise Console Server is NOT running.
% wtdbstat
The RDBMS database server is running.
```

Figure 210. Checking Status of Event Server and Database Server

When the database is full:

- The tec_rule or tec_reception file in /tmp states that the process was terminated with an error code 81 or 17.
- `wtdbspace` reports the database is 100 percent full (see Figure 211 for an example of the `wtdbspace` command).

.

Note

For TEC 3.6, the `wtdbspace` command has not been implemented for DB2 or MS/SQL. This functionality will be added in a future release.

```
root@rh0255f:~mnt# wtdbspace

Tablespace Usage:
Tablespace       Used for      Allocated       Used       Free    %Used    %Free
---------------- ------------ ---------- ---------- ---------- -------- --------
TEC_DATA_TS      Data              50 MB    3.03 MB   46.97 MB   9.35 %  90.65 %
TEC_TEMP_TS      Temporary         10 MB       0 MB      10 MB      0 %    100 %

Table Details:
Name                    Rows     Reserved   Data pages  Index pages Unused pages
-------------------- ------- ------------ ------------ ------------ ------------
tec_t_clt_req_log          0        48 KB         2 KB         4 KB        42 KB
tec_t_evt_rec_log        823      1448 KB        56 KB      1368 KB        24 KB
tec_t_evt_rep            708       238 KB       204 KB         4 KB        30 KB
tec_t_gem_threshld         0        32 KB         2 KB         2 KB        28 KB
tec_t_isa               1708        80 KB        70 KB         0 KB        10 KB
tec_t_op_ass_log         708        48 KB        20 KB         2 KB        26 KB
tec_t_role                 0        32 KB         2 KB         2 KB        28 KB
tec_t_severity             6        32 KB         2 KB         2 KB        28 KB
tec_t_slots_evt        10324       958 KB       880 KB        36 KB        42 KB
tec_t_status_event         4        32 KB         2 KB         2 KB        28 KB
tec_t_status_task          4        32 KB         2 KB         2 KB        28 KB
tec_t_task_rep             0        48 KB         2 KB         4 KB        42 KB
                             ------------ ------------ ------------ ------------
                                 3028 KB      1244 KB      1428 KB       356 KB
```

Figure 211. Example Output for wtdbspace Command

To fix this problem, use the `wtdbclear` command. The `wtdbclear` command has a lot of important parameters that should be used carefully. Lookup the man pages for this command before using it.

Oracle Note

The *percent used* number in `wtdbspace` is calculated by counting the number of database *extents* dedicated to the `tec` tables. Oracle does not reallocate extents in the case of a DELETE statement, and that's what `wtdbclear` uses. Oracle will reallocate those extents on demand so they are available for use. Use the table detail report in `wtdbspace` to confirm that you have deleted rows and, thereby, created space.

In the case where portmapper is not running, the event server won't start and the database server will not be running. You can check to see that the portmapper is running by issuing the command: `rpcinfo -p`. This is only relevant to UNIX systems; there is no portmapper on Windows NT.

When the database is not running, and you try to start the TEC Event server, the `RIM Object tec was not found` message in Figure 212 will appear:

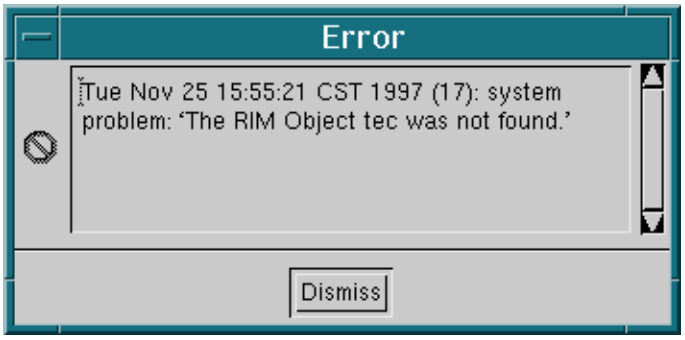


Figure 212. TEC Server -Database Engine Not Running Error Dialog

When getting this message, the following commands should be used to check the actual status:

`wgetrim tec`	To see if the RIM host is configured correctly.
`wrimtest -l tec`	To see if the RIM host is working correctly.
`wstatesvr`	To see if the TEC Server is running or not
`wtdbstat`	To see if the database server is running or not.

16.7.2 Event Console Troubleshooting

The most common error with the event console is the removal of an event console icon from the desktop rather than a deletion. An administrator cannot create a new event console for this user. To fix this, perform the following:

1. Re-link the object with:

   ```
   wln /Library/EnterpriseClient/admin-name /Administrators/admin-name
   ```

2. Delete the event console and create a new one.

Many problems occur because a user does not have the proper roles or has too many roles. These problems usually become apparent when an administrator is unable to perform actions on an event. Check to make sure they have the appropriate role and that they do not accidentally have the none role selected. The dialog in Figure 213 will show up by using right mouse button on the event console icon and choosing the **Assign Event Group** menu item:

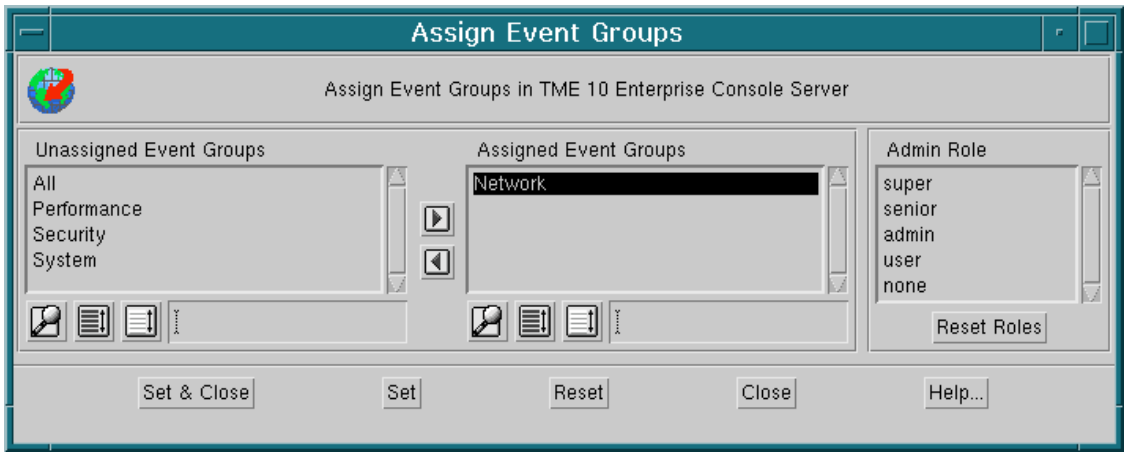


Figure 213. Assign Event Groups to the Event Console

Other problems can occur because of filters that are too restrictive. If too many filters are set, it can take a long time to open a console. To list event groups and their filters, use the following command:

```
# wlseg -f
```

A sample output for this command follows in Figure 214 on page 524:

```
# wlseg -f
Network:
        bitmap: genapp48
        filters:
          id 0, class: EVENT, source: SNMP, sub_source: *, origin * sub_origin *
          id 1, class: EVENT, source: SNM, sub_source: *, origin * sub_origin *
          id 2, class: EVENT, source: HPOV, sub_source: *, origin * sub_origin *
          id 3, class: EVENT, source: NV6K, sub_source: *, origin * sub_origin *
Performance:
        bitmap: genapp48
        filters:
          id 4, class: EVENT, source: LOGFILE, sub_source: *, origin * sub_origin *
          id 5, class: EVENT, source: NT, sub_source: *, origin * sub_origin *
Security:
        bitmap: genapp48
        filters:
          id 6, class: EVENT, source: LOGFILE, sub_source: *, origin * sub_origin *
          id 7, class: EVENT, source: NT, sub_source: *, origin * sub_origin *
System:
        bitmap: genapp48
        filters:
          id 8, class: EVENT, source: AS400_MSGQ, sub_source: *, origin * sub_origin *
          id 9, class: EVENT, source: LOGFILE, sub_source: *, origin * sub_origin *
          id 10, class: EVENT, source: NT, sub_source: *, origin * sub_origin *
All:
        bitmap: genapp48
        filters:
          id 11, class: EVENT, source: *, sub_source: *, origin * sub_origin *
```

Figure 214. Example Output of wlseg Command

16.7.2.1 The Event Does Not Appear in the Event Console

When an event does not appear in the event console, the following steps may help, most of which are described in the rest of this section:

- Event console
  - Is the event source defined?
  - Is the event source being used as a filter in an event group?
  - Is the event group assigned to the right administrator?
  - Is the correct .baroc file being used?
- Event server
  - Did the event get to the event server system?
  - Did the event server receive the event?
- Event adapter
  - Is the adapter running?
  - Is the event server location properly specified?
  - Did the adapter send the event?

Is the event source defined?
Run the command `wlssrc` to see if the source of the event is defined to the event server. Source is the value that tells the event server which adapter is sending the event. Sources are used as a filter value to determine which events will show up in the event console.

```
# wlssrc
Source Name
-------------
LOGFILE
NV6K
HPOV
NT
SENTRY
TEC
SNMP
SNM
AS400_MSGQ
AS400_ALERT
NV390MSG
NV390ALT
```

Figure 215. Output from wlssrc Command

Is the event source being used as a filter in an event group?
Run the command `wlseg -fa` or open the dialog shown in Figure 213 on page 523. This displays the event groups that are defined. Event groups are logical groupings of related events into one spot in the event console. For any given event group, there must be at least one filter. Look for a source filter within the event group. Then check to see if the event group with the appropriate source filter is assigned to the administrator in whose event console the event is expected.

Is there is a parsing failure, is the correct baroc file being used?
Check the defined event classes for the rule bases. The command `wlsrbclass RuleBaseName` displays the event classes defined for the rule base. Be aware that the `RuleBaseName` is case sensitive.

Did the event get to the event server system?
TEC events come from the event adapters, are processed by the event server, stored in the RDBMS, and are displayed in the event console GUI. The problem with missing events could be anywhere along this route.

Did the event server receive the event?
The easiest way to answer this question is to check the output of `wtdumprl` and the `wtdumper` commands. Figure 216 on page 526 shows the output from `wtdumper`:

```
# wtdumper
ES~1~879461181(Nov 13 16:46:21 1997)~1~TEC_Start~
TEC~~~~~~OPEN~
~[ admin]~MINOR~
Nov 13, 1997 22:46~
TEC Event Server initialized~
~0~
0~0~ES~1~879813086(Nov 17 18:31:26 1997)~1~TEC_Start~
TEC~~~~~~OPEN~
~[ admin]~MINOR~
Nov 18, 1997 00:31~
TEC Event Server initialized~
~0~
0~0~ES~1~879973724(Nov 19 15:08:44 1997)~1~TEC_Start~
TEC~~~~~~OPEN~
~[ admin]~MINOR~
Nov 19, 1997 21:08~
TEC Event Server initialized~
~0~
0~0~ES~1~880480032(Nov 25 11:47:12 1997)~1~TEC_Start~
TEC~~~~~~OPEN~
~[ admin]~MINOR~
Nov 25, 1997 17:47~
TEC Event Server initialized~
~0~
0~0~ES~1~880495825(Nov 25 16:10:25 1997)~1~TEC_Start~
TEC~~~~~~OPEN~
~[ admin]~MINOR~
Nov 25, 1997 22:10~
TEC Event Server initialized~
~0~
0~0~#
```

Figure 216. Output from wtdumper Command

See Figure 202 on page 511 to compare the `wtdumper` output in Figure 216 to a `wtdumprl` output. Events that are displayed on the event consoles come from the event repository, which is what `wtdumper` displays. If the event is not in `wtdumper`, it will not show up in the event console.

If an event shows as being processed in `wtdumprl` and does not show up in the `wtdumper` output, the event must have been dropped by the rule base.

If `wtdumprl` generates no output, the database is probably full. Run `wtdbclear` to reduce the size of the logs and restart the server. A parsing failed message indicates what wasn't defined to the server or rule base. If you get output, but the event is not here, the event server didn't receive it.

Is the adapter running?

If the event is not getting to the event server, then the adapter should be checked to see if it is running or not.

On Windows NT, this can be done by checking the services panel, as shown in Figure 217:

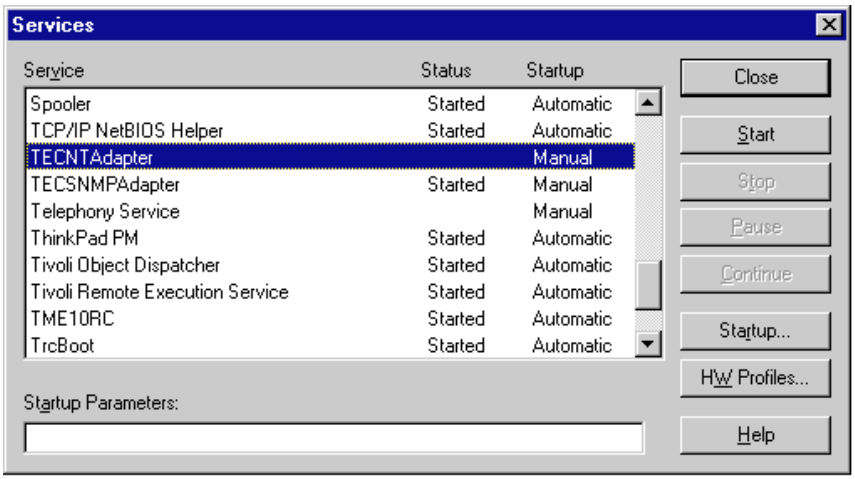


Figure 217. Services Panel for TEC Adapter Control on Windows NT

In Figure 217, you can see an example of two TEC Adapters:

- TECNTAdapter
- TECSNMPAdapter

The services panel allows us to get the actual status, to start and stop the adapter, and to set parameters to automatically or manually start the adapter while the system is re-booting. On UNIX systems, the TEC adapters must be stopped and started manually using script files. Refer to the *Tivoli Enterprise Console Adapters Guide* for specific stop/start commands for each adapter.

Is the event server location properly specified?
Check if the event server location is properly specified in the /etc/Tivoli/tecad/etc/adapter.conf file and if a filter is preventing the event from being generated. The important line is the `ServerLocation=@EventServer` line. For example, if there are several TEC servers available, make sure the right server is addressed within this statement. The default is shown in this example. If it is a Tivoli Version adapter, the server port is not necessary because the oserv IOM channel is used.

For a non-Tivoli adapter, the ServerLocation must be the host name or IP address of the event server. The ServerPort line needs to be uncommented, and the port number should be the port that the event server is listening on. Figure 218 shows an example of the configuration file for the logfile_adapter:

```
# TE/C LogFile adapter configuration file.
#
# $Date: 1995/11/27 16:23:48 $
#
# $Source: /tivoli/development/src/2.0/apps/tec/adapters/logfile/conf/
aix4-r1.conf/tecad_logfile.conf,v $
#
# $Revision: 1.1 $
#
# $Id: tecad_logfile.conf,v 1.1 1995/11/27 16:23:48 wgg Exp $
#
# Description:
#
# (C) COPYRIGHT TIVOLI Systems, Inc. 1994.
# Unpublished Work
# All Rights Reserved
# Licensed Material - Property of TIVOLI Systems, Inc.
#

ServerLocation=@EventServer
BufEvtPath=/etc/Tivoli/tec/logfile.cache
# ServerPort=9999
EventMaxSize=4096

Filter:Class=Logfile_Base
Filter:Class=Logfile_Sendmail
Filter:Class=Amd_Unmounted
Filter:Class=Amd_Mounted
```

Figure 218. Configuration File for a Logfile Adapter

If there are interconnected TMRs or multiple event servers, then the EventServer entry in this file might need to be modified. When using the keyword `TestMode=Yes`, it is possible to specify a logfile location instead of a server location with the `ServerLocation` keyword. This will allow us to send the output to a file without using any connections to another machine. It helps for testing the adapter locally.

Did the adapter send the event?

There should be a daemon process running for each TEC adapter on a host. These daemons can be configured to start at oserv start time, or at system boot time if it is an non-Tivoli adapter. The start up file for the logfile adapter can be found in /etc/Tivoli/tecad/bin/init.tecad_logfile (see Figure 219 on page 529).

```
#!/bin/sh
# script to start/stop the logfile adapter: tecad_logfile must
# point to the tme or non_tme adapter executable in the bin dir.
#
# $Date: 1996/10/17 20:29:35 $
#
# $Source: /tivoli/development/src/2.0/apps/tec/adapters/logfile/bin/
init.tecad_logfile,v $
#
# $Revision: 1.33 $
#
# $Id: init.tecad_logfile,v 1.33 1996/10/17 20:29:35 jmills Exp $
#
# Description:
#
# (C) COPYRIGHT TIVOLI Systems, Inc. 1994.
# Unpublished Work
# All Rights Reserved
# Licensed Material - Property of TIVOLI Systems, Inc.
#
.
.
.
else
                echo "Starting syslogd..."
                case $INTERP in
                        aix3-r2|aix4-r1)
                                startsrc -s syslogd
                                ;;
                        solaris2|dgux5)
                                /usr/sbin/syslogd
                                ;;
                        hpux9|hpux10)
                                /etc/syslogd
                                ;;
                        sunos4)
                                /usr/etc/syslogd
                                ;;
                        *)
                                syslogd
..
```

Figure 219. TEC Adapter Daemon Start-up Excerpt from init.tecad_logfile

If the adapter is running, but no activity is generated in the output file, check that this event is properly defined and formatted in the adapter. Look at the file /etc/Tivoli/tecad/etc/tecad_logfile.err. This file resembles the diagnostic file for the server, a series of stanzas defined as `/dev/null` by default.

Change the `TECIO` stanza to send standard out to a text file as in Figure 220 on page 530:

```
.
.
.
#
# MODULE = TECIO
#
TECIO  MINOR    /tmp/tec_outfile
TECIO  MAJOR    /tmp/tec_outfile
TECIO  FATAL    /dev/null
TECIO  LOW      /dev/null
TECIO  NORMAL   /dev/null
TECIO  VERBOSE  /dev/null
```

Figure 220. Changing TECIO Stanza for Error Output to a File

Restart the adapter. Examine this output file to trace the adapter behavior. The following steps outline this procedure:

1. `tail -f outputfile`
2. Generate event
3. Check for DISCARD
4. Waiting, sent, or fail

If the event is discarded by the adapter, add the event to the following files using the appropriate syntax:

`/etc/Tivoli/tecad/etc/tecad_logfile.baroc` (for logfile adapter)

`/etc/Tivoli/tecad/etc/tecad_logfile.cds` (for logfile adapter)

Plus:

`/etc/Tivoli/tecad/etc/tecad_logfile.fmt` (for logfile adapter)

`/etc/Tivoli/tecad/etc/tecad_snmp.oid` (for snmp adapter)

16.7.2.2 Event Console Initialization is Too Slow

If your TEC consoles are taking too long to start-up, one problem may be the size of your local event cache for your TEC console. This cache is configured from the GUI dialog for Event Console -> Event Group -> Options -> Message Time Limits. The longer your time limits are set for, the larger the cache, and, thus, the longer it takes to bring up the console. This is controlled at the individual TEC client level.

Another factor could be the number and complexity of event group filters. The more event groups and filters you setup, the longer it will take your console to

initialize, as it must retrieve all the data for each event group before control is given back to the user.

16.7.2.3 ACF Distribution Failure

To trace what is occurring and what may be going wrong during an ACF distribution, turn on the following tracing:

To turn on gateway logging:

```
wgateway gateway-name set_debug_level #   where # is 0 through 9
```

The log file is found in $DBDIR/gatelog.

To turn on endpoint logging:

```
./opt/Tivoli/lcf/dat/1/lcf_env.sh (use directory path for your endpoint)
cd $LCF_DATDIR
./lcfd.sh stop
```

Edit last.cfg in this directory and change `log_threshold=1` to `log_threshold=4`

Restart the `lcfd` daemon with:

```
./lcfd.sh start
```

This log file is located in $LCF_DATDIR/lcfd.log. This is the best place to start to see what failed on the endpoint during distribution.

16.7.2.4 NT Event Adapter Running at 100 percent CPU

If your NT event adapter is running at 100 percent CPU utilization, and you have installed NT Service Pack 4, you may have overlaid the eventlog.dll file that Tivoli needs to run the NT log adapter. Service Pack 4 installs a newer version of `eventlog.dll` which is incompatible with the NT event adapter. If you chose to backup files replaced by the service pack, you may be able copy it back over. Otherwise, you will need to obtain the version of `eventlog.dll` prior to SP4 (dated 4/30/97 size: 50960).

16.7.2.5 Columns on Event Console not Wide Enough

The EnterpriseClient class and objects have attributes for setting the TEC Message Viewer column widths and headings. The new attributes are of the form `<column>_cw` (for setting the column width) and `<column>_cn` (for setting

the column name or heading). The complete list of attributes, their IDL types, and their default values is given in Table 24:

Table 24. Event Console Column Attributes, Types, and Values

Column Widths			Column Headers	
status_cw	short 8		status_cn	string "Status"
class_cw	short 20		class_cn	string "Class"
severity_cw	short 12		severity_cn	string "Severity"
origin_cw	short 16		origin_cn	string "Origin"
hostname_cw	short 16		hostname_cn	string "Hostname"
message_cw	short 32		message_cn	string "Message"
date_cw	short 22		date_cn	string "Date"
administrator_cw	short 22		administrator_cn	string "Administrator"
event_key_cw	short 22		event_key_cn	string "EVENT_KEY"
action_f_cw	short 1		action_f_cn	string "F"
id_cw	short 22		id_cn	string "ID"
action_r_cw	short 1		action_r_cn	string "R"
repeat_count_cw	short 22		repeat_count_cn	string "Repeat Count"
action_s_cw	short 1		action_s_cn	string "S"
source_cw	short 16		source_cn	string "Source"
sub_origin_cw	short 16		sub_origin_cn	string "Sub-Origin"
sub_source_cw	short 16		sub_source_cn	string "Sub_Source"
action_u_cw	short 1		action_u_cn	string "U"
dpage_key_cw	short 16		dpage_key_cn	string "dpage-key"

Setting column widths and headers

You can use the Tivoli Framework command line interface `idlattr` to retrieve and modify these attributes. Use `wlookup -ar EnterpriseClient` to get a list of TEC console objects (EnterpriseClient_OID). After modifying a console object, the changes will be effective the next time that the console is initialized.This has to be done for each individual enterprise client console you want to modify.

Note

Refer to the warnings in Chapter 6, “Commands and Logs for Troubleshooting” on page 131 before using `idlattr` calls in your TMR. Always backup the Tivoli database before performing any direct manipulation.

The general form to retrieve an attribute value is:

```
idlattr -tg <EnterpriseClient_OID> <attribute> <type>
```

The general form to modify an attribute value is:

```
idlattr -ts <EnterpriseClient_OID> <attribute> <type> <value>
```

Examples:

To retrieve an attribute value:

```
idlattr -tg <EnterpriseClient_OID> status_cn string
```

To modify an attribute value:

```
idlattr -ts <EnterpriseClient_OID> status_cn string \"Health\"
```

Note

Be sure and include the backward slashes (\) before the the double quotes (") when modifying the column headers. This prevents the shell from stripping the double quotes from the command.

16.7.3 Rule Base Errors

Rule base errors usually fall into two categories:

- Problems creating new rule bases.
- The server crashes and restarts itself.

Important

Never change the default rule base directory contents.

The reason you should never change the default rule base directory contents is that the default rule base contains default rules and default .baroc files, which are necessary for each new rule base. If this is changed, the base can be corrupted, and each new rule base will be corrupted as well. A corrupted

rule base will not compile, and this makes it useless because it is not loadable to the TEC. In the case that there is no valid rule base for the TEC, it will not start. That means whenever creating a new rule base, create a new directory for its contents.

In addition, having an unmodified default rule base allows you to have a stable rule base to be used at any time for testing. If the default rule base is corrupted, a backup is provided in $BINDIR/TME/TEC/default_rb.tar.

The dialog for creating a new rule base path is shown in Figure 221:

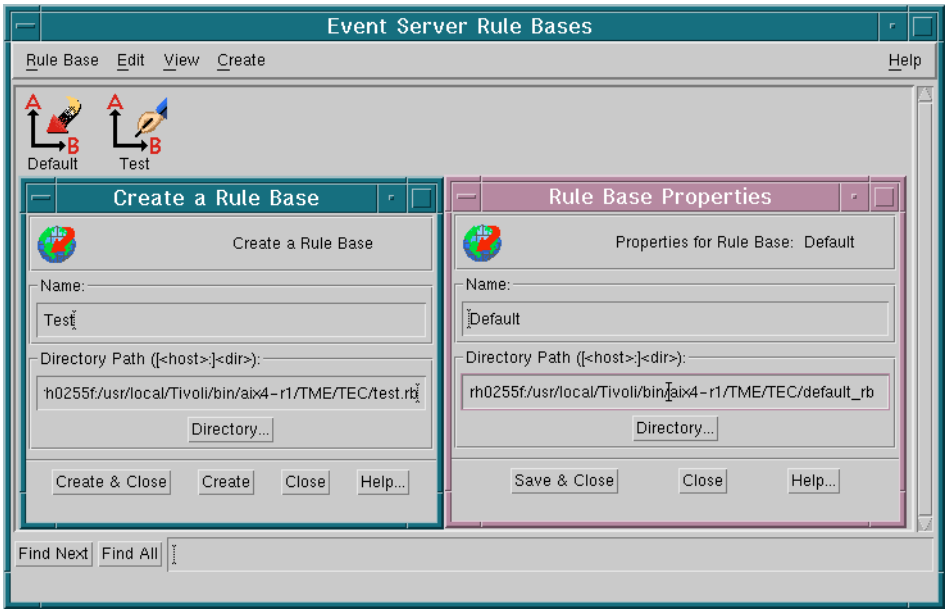


Figure 221. New Rule Base Path

There is another consideration when creating a new rule base directory on Windows NT. You might be inclined to use the form `Hostname:Drive:\DirectoryPath\`, but this will fail. You need to use forward slashes as in `Hostname:Drive/DirectoryPath/`. Doing it the wrong way doesn't cause a problem until an attempt to edit the rules. At this point, the dialog in Figure 222 pops up, and its meaning is not clear:

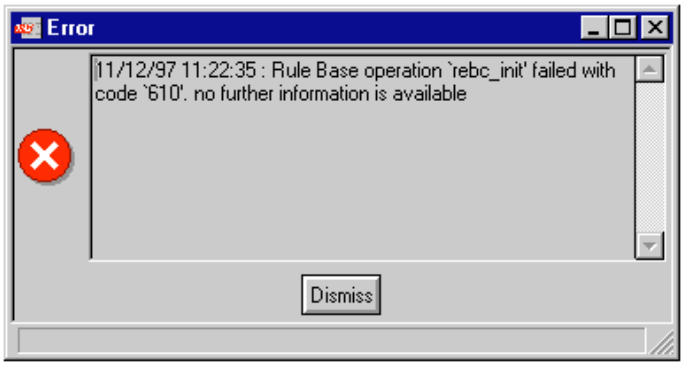


Figure 222. Error Panel for Wrong Rulebase Path

If the event server crashes and (possibly) restarts itself, it may be caused by a problem in the rule base. When the oserv restarts, it tries to restart TEC (if TEC is set to auto-start). If this happens, get the rule base and the list of events that were sent. Output from `wtdumprl` helps provide the events.

Perform the following:

1. Recompile the rule base with tracing on.

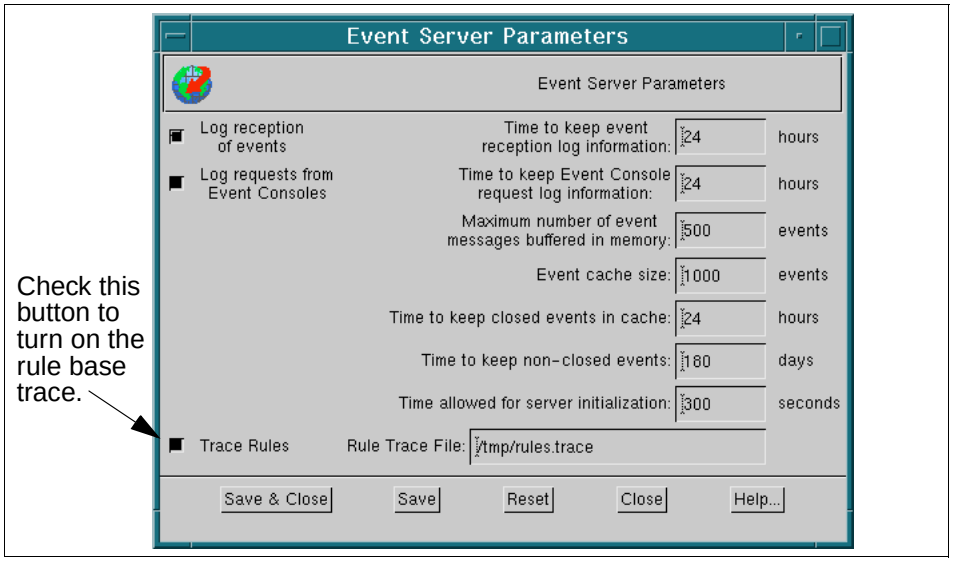


Figure 223. TEC Server Parameters (How to Turn on Rule Base Trace)

2. Recompile the rule base with tracing:

```
wcomprules -t RuleBaseName
```

3. Reload the rule base (see Figure 224 on page 536).

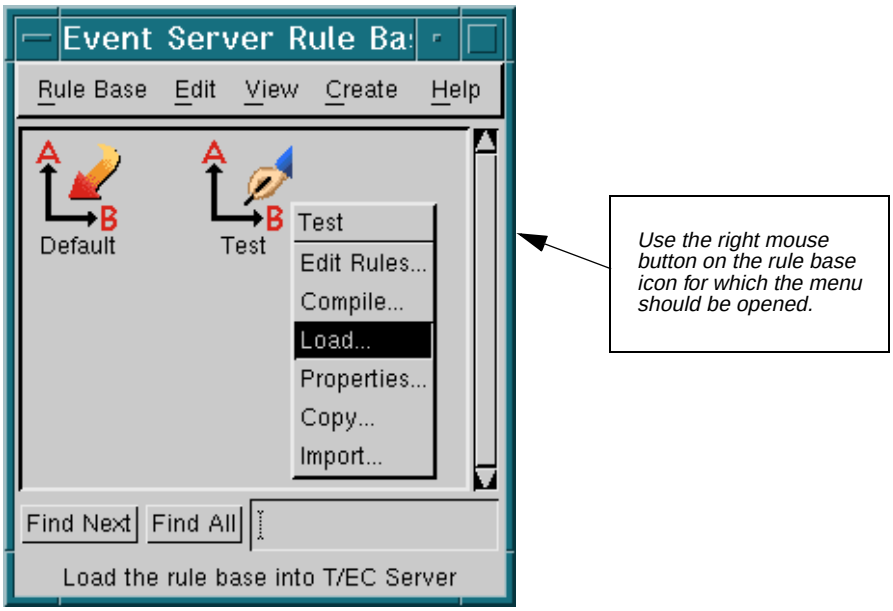


Figure 224. Reload Rule Bases (GUI)

4. Stop and start the TEC server.
5. Generate events until the server crashes.
6. Look at the trace file to see where the failure occurred.

The procedures shown in the previous figures use the GUI interface. What it looks like when using the command line interface for the same scenario is illustrated in Figure 225:

```
% wtdumprl
% wsetesvrcfg -t /tmp/rules.trace
% wcomprules -t RuleBaseName
% wloadrb RuleBaseName
% wstopesvr
% wstartesvr
% wpostesmsg -r severity -m “Value for mesg slot” slot=val slot=val E_CLASS SOURCE
% vi /tmp/rules.trace
```

Figure 225. Command Line Tracing of Rule Bases

Note

Be sure and turn tracing off when you are done testing. Otherwise, you might end up with a huge /tmp/rules.trace file and overflow your file system. To turn off tracing from the command line, use the `wsetesvrcfg -t` command.

To see what went wrong with the events, the first thing to try is the `wtdumprl` command. Depending on how many entries it has, it is necessary to use certain parameters with this command to get only those entries you are looking for. Figure 200 on page 509, Figure 201 on page 510, and Figure 202 on page 511 show examples of `wtdumprl` outputs.

Next, check the rule base itself. First determine that it is actually being used. Use the `wlscurrb` command to determine which rule base is currently loaded into the event server. The command, `wlsrb -d RuleBaseName`, will tell you the name of the rule base and its physical directory location.There can be multiple rule bases, but only one can be loaded into the event server at a time. In the rule base directory, there will be subdirectories called TEC_CLASSES and TEC_RULES. All event class definitions will be in the TEC_CLASSES directory in files called <name>.baroc. All rules will be in TEC_RULES directory in files called <name>.rls.

16.7.3.1 Rules Don't Work Properly

Compile the rules with tracing on and look at the trace output. Get the rule base and a list of events for which the behavior is not as anticipated.

16.7.3.2 Rule Base Loading Is Lengthy and Fills Up File System

When rules are loaded into the event server, three directories in the rule base directory (`wlsrb -d`) are copied into `$DBDIR/tec/rb_dir`; TEC_CLASSES, TEC_RULES, and TEC_TEMPLATES. In actuality, what really happens is everything under the rule base directory is copied to the `$DBDIR/tec/rb_dir` directory. If you did not create your rule base in a new or empty directory, then all of the other files and directories in your rule base directory will also get copied.

16.7.3.3 Server Crashes in Response to an Event

Again, suspect an imperfectly crafted rule: One that perhaps does what you told it to do rather than what you wanted it to do! Load the default rule base. Stop and start the event server. Send the errant event. If the server does not crash, refer to Figure 223 on page 535 on tracing rule output.

16.7.3.4 Forced Clearing of the Rules Cache Event

This event signifies that the rules cache is not large enough for the correlations being performed in the rulebase. The Event Cache, also known as the Rules Cache, is configurable in the event server parameters. The rules cache is maintained for rule correlation purposes. Adjust this value to hold all the events you will need to correlate on but not more than you need. Setting this value too high will adversely impact TECs performance.

Each rule in the rulebase will use this cache to correlate with incoming events. If the rulebase uses primitives like ALL_INSTANCES or ALL_DUPLICATES, and the rules cache is large, CPU usage may increase proportionately, and the time to process a rule through the rule base will increase as well. It is important to use templates, such as `commit_rule` (commits the rule set), `commit_set` (commits the whole rulebase), and `commit_action` (commits just the rule), to minimize the amount of time an event spends in rules processing.

In addition, CLOSED events can take up space in the rules cache and should be cleaned out proportionate to the number of events coming in and being processed and the number of events you want to correlate. The longer it takes to process events through the rulebase, the longer it will be for `tec_reception` to pull new events from the memory buffer, and thus, from the reception log.

Chapter 17. Tivoli Output Manager

This chapter documents troubleshooting information for Tivoli Output Manager. This product was previously known as Tivoli Destiny, and this name is used throughout this chapter. This information has been generously donated by the original authors in Tivoli support and education and has been mostly reproduced as-is.

17.1 Expected Audience and Knowledge

This material was originally aimed at Level 2 Customer Support representatives. You are expected to:

- Have successfully completed Tivoli's Destiny Overview course or have equivalent experience.
- Have reviewed the Destiny Server Administration Manual.

This chapter has the following topics:

Output Manager/Destiny Overview:
Provides an overview of Output Manager processes and tools.

Troubleshooting Destiny Problems:
Describes basic troubleshooting information for Output Manager.

Frequently Asked Questions:
Provides information on how to access FAQs.

Error Solutions:
Provides tables of solutions to various types of problems.

17.2 Output Manager/Destiny Overview

This section provides a description of the Output Manager background processes and tools that can be used to help understand the current state of Output Manager.

17.2.1 Destiny Background Processes

In order for Destiny to perform its job, it makes use of several *background processes*. Each process works together to do the following:

- Get a document to its designated destination.
- Clean up entries in the database.
- Communicate messages to and from each process.

- Get ready for the next document to process.

These processes are described below:

17.2.1.1 Netman

The Netman process must be running before any other Destiny process can run. It is typically set up to automatically start when the Destiny server boots up. This process listens to the port address configured when Destiny was installed. Without the port listening process, no other Destiny process can function. To see if this process is running, it can be viewed by going to Control Panel>Services>Destiny Netman.

17.2.1.2 Spoolman

The Spoolman process is the parent process for all Destiny background processes running on a Destiny server. Spoolman starts up all the required background processes. When Spoolman detects a catastrophic error from one of the processes, or if Spoolman is stopped, Spoolman will attempt to shut down all background processes. To see if Spoolman is running, simply bring up Task Manager.

17.2.1.3 Mapper

The Mapper process is the first to determine that a new file has somewhere to go. It either creates an entry in the NEWS database for LQM (Local Queue Manager) or for NQM (Network Queue Manager) on the Node where the file first appears. If the file is to be sent to another Node, then NQM will handle the file and send it to the designated Node. If the file is destined to a local Queue, then the file is handed off to LQM.

17.2.1.4 LQM

This is the Local Queue Manager (LQM), which handles any transactions that are destined for a local Queue. There will be an LQM process for every Queue defined within Destiny. LQM adds entries into the NEWS database in case it has to forward a file to a remote Queue.

17.2.1.5 NQM/NQM Child

The NQM/NQM Child process handles files destined for a Queue not found on the node on which it first appeared. Network Queue Manager (NQM) follows three rules to get a file to the proper place:

1. Send to the target node directly. If this fails, perform step two.
2. Send to own Default Routing Manager (DRM). If this fails, perform step three.
3. Send to own Domain Manager (DM).

If all three steps fail, then the file will be marked as being in an error state.

17.2.1.6 NetWat

NetWat is the process NQM relies on to actually get a file from one node to another on the Network. If necessary, NetWat places entries into the NEWS database on the node the file is going to.

17.2.1.7 DirWatch

DirWatch processes files placed in a configured directory and deletes them once processed.

17.2.1.8 Trashman

Trashman is the last process involved when a document is processed by Destiny. Its job is to make sure all pieces of a document have been processed and flagged as done.

17.2.1.9 Logman

The Logman process usually runs (based on Spoolman_Log_Period found in Destiny.ini) once a day. Logman purges documents from the NEWS database queues flagged as `printed` and `deleted`.

17.2.1.10 DestDirnt

The DestDirnt process is not a Spoolman child process. However, it may run on a Destiny server if the customer has also installed Destiny Direct on the server machine. DestDirnt handles getting Destiny Direct documents into the NEWS database for processing by Destiny.

17.2.2 Destiny Tools

Destiny has two tools that are installed on the server but not documented in the user guide. The tools reside in the `\DESTINYHOME\util` directory. These tools are `SQLView` (an application) and `DISP` (a batch file).

17.2.2.1 SQLView

`SQLView` is a simple GUI that allows you to view entries in the database. It works with SQL and JET installations:

1. Using Windows Explorer, simply drill down to the `\DESTINYHOME\util` directory.
2. Double-click on the **SQLView** application. The application is very easy to use.
3. Select the database you want to view.
4. Select the desired table.

5. Select the number of records you want selected at a time.

You are allowed to select only two SQL statements (you can't override the SQL statement line). One SQL statement selects ALL records in the selected table. The other SQL statement will display a count of ALL records in the selected table.

17.2.2.2 DISP

`DISP` is a batch file that does a `tail notif` command on the desired `stdlist` file:

1. Get to a command prompt and `cd` to the `\DESTINYHOME\util` directory.
2. Now type `DISP filename` (for example, `DISP UNKNOWN`). This will display the contents of the desired stdlist file followed by a pause.

As long as you have not closed the command prompt window, as processes write entries to the stdlist file, the activity will be displayed in the open command window. This is a good tool for watching Destiny process documents in real time.

17.3 Troubleshooting Destiny Problems

In most cases, Destiny troubleshooting involves reviewing an error log or viewing a pop-up error. Once the error is found, then an action to correct the problem can be determined.

Problems in Destiny can be classified into four broad areas:

- GUI
- Destiny Direct Client (NT/95)
- SLP Client (NT/Unix)
- Destiny Server (NT)

First, determine in which of these areas the difficulties are occurring. To further isolate the problem, you might ask the following questions about the setup. For example:

- Does the problem occur with all or just some of the input programs (Dirwatch, Destiny Direct, Destiny Transport)?
- Does the problem occur with all or just some of the output destinations (printer, email, pager)?
- Are you using NT 4.0 SP3 on their server?

- Are you using SQL or JET?

Once you have determined in what area the problem is occurring, you can then proceed to diagnose the problem.

17.3.1 GUI

Two applications make up the GUI portion of Destiny:

- Composer (for configuring Destiny)
- Conductor (for daily operations)

Once you know it is a GUI problem, then you need to know if the problem is in Composer or Conductor.

17.3.1.1 Composer

To start up Composer, go to the **Start>Destiny>Composer** menu. Errors within Composer will be either pop-up style or can be found in the client directory within the Destiny home directory in a file named composer.log.

Questions to ask would include:

- Is it a configuration issue?
- What is being configured?
- Is the GUI allowing the configuration to be saved?
- Is the GUI allowing the configuration to be modified?
- Are there any errors of any kind (on screen or in `\DESTINYHOME\client\composer.log`)?
- Is it a `Push` issue?
- Is the `Push` working (verify with Conductor to view configuration)?
- Are there any errors of any kind (on screen or in `\DESTINYHOME\client\composer.log`)?

17.3.1.2 Conductor

To start up Conductor, go to the **Start>Destiny>Conductor** menu. Errors within Conductor will be either pop-up style or can be found in the client directory within the Destiny home directory in a file named conductor.log.

Questions to ask would include:

- Does Conductor return incorrect information?
- Is starting/stopping of Spoolman an issue?

- Is starting/stopping a Destination an issue?
- Is starting/stopping a Queue an issue?
- Is starting/stopping a Watcher an issue?
- Is there trouble moving a document from one Queue to another?
- Are there any errors of any kind (on screen or in `\DESTINYHOME\client\conductor.log`)?

17.3.2 Destiny Direct Client (Windows NT/95)

Destiny Direct runs on Windows NT or Windows 95. There are no error log files for Destiny Direct. All errors on the Direct side will be Windows pop-up style error boxes.

Questions to ask could include:

- What OS is the customer running Destiny Direct under (Windows NT or 95)?
- Are you able to log in to the server from Direct?
- Are you able to get to the Direct: Print Destinations window?
- Are you able to see the desired Destination for the document from Direct?
- Does the document print at all (from Windows NT/Windows 95 versus Destiny Direct)?

17.3.3 SLP Client (Windows NT/Unix)

SLP runs on both NT and Unix. The syntax of the commands is identical on both platforms although a few parameters (such as `-crlf`) won't necessarily make sense on the NT platform. Questions to ask could include:

- Has the slp.ini file been configured? Review the slp.ini settings.
- Can you ping the desired Destination (if a printer)?
- Can you print to the desired Destination using LPR (if a printer)?
- Does Destiny even see the document coming from SLP (verify by viewing the `\DESTINYHOME\stdlist\yyyy.mm.dd\unknown` stdlist file)?
- Is the document reaching a Destination of any kind (verify by using Conductor and viewing each Queue)?
- Any errors of any kind (in the stdlist file located in `\DESTINYHOME\stdlist\yyyy.mm.dd\unknown`)?

17.3.4 Destiny Output Server (Windows NT)

The Output Server portion is the heart of Destiny. Most activity can be viewed by using Conductor or by viewing one of the error logs (resides in `\DESTINYHOME\stdlist\yyyy.mm.dd` where `yyyy.mm.dd` = year.month.day Spoolman or Conductor was last started).

17.3.4.1 Unknown

The Unknown error log is the main log showing daily activity on the Output Server. Most of Destiny's background processes post messages to this log file (resides in `\DESTINYHOME\stdlist\yyyy.mm.dd\Unknown` where `yyyy.mm.dd` = year.month.day Spoolman or Conductor was last started).

17.3.4.2 Condserv

Condserv is the error log for the Conductor application. If there is difficulty with Conductor, look at the contents of this log file. This file will list transactions from the Conductor point of view (resides in `\DESTINYHOME\stdlist\yyyy.mm.dd\Condserv` where `yyyy.mm.dd` = year.month.day Spoolman or Conductor was last started).

17.3.4.3 Netman

The Netman file lists network activity on each Output Server (resides in `\DESTINYHOME\stdlist\yyyy.mm.dd\Netman` where `yyyy.mm.dd` = year.month.day Spoolman or Conductor was last started).

17.3.4.4 Netwat

The Netwat file lists messages passed between Output Servers (resides in `\DESTINYHOME\stdlist\yyyy.mm.dd\Netwat` where `yyyy.mm.dd` = year.month.day Spoolman or Conductor was last started).

17.3.4.5 Conductor

The Conductor file resides in the `\DESTINYHOME\client` directory. If there are problems with the use of Conductor, then this log file can be useful (by development) in deciphering what the problem might be.

If you exit Conductor and get back into Conductor, then the conductor.log file will get overwritten. It is important to rename the conductor.log file before restarting Conductor.

17.3.4.6 Composer

The Composer file resides in the `\DESTINYHOME\client` directory. If there are problems with the use of Composer, then this log file can be useful (for Tivoli development) in deciphering what the problem might be.

If you exit Composer and get back into Composer, then the composer.log file will get overwritten. It is important to rename the composer.log file before restarting Composer.

17.4 Troubleshooting a Push Operation

In Destiny, the Enterprise and Domain configuration information is maintained in two databases:

- Enterprise information is kept in the UED (Unison Enterprise Database).
- Domain information is kept in the SCD (Spoolmate Configuration Database).

Changes are made to these databases by using the Destiny Composer application, which modifies and reads these two databases.

Day-to-day control of Destiny (stopping and starting nodes and queues, resubmitting jobs, etc.) is performed through the use of the Destiny Conductor application. Conductor reads and modifies a third database known as the NEWS database, which maintains information about the Destiny Node on which it exists. In addition to node-specific information, the NEWS database also must include information about the node's respective Enterprise (UED) and Domain (SCD).

Information from the UED and SCD is replicated through the Destiny `Push` operation. A `Push` operation is performed by:

1. Selecting a Node or Domain in Composer
2. Using the `Composer->Push` command
3. Selecting whether to replicate the following:
   - Enterprise information (UED) – select **Enterprise Configuration**
   - Domain information (SCD) – select **Destiny Domain Configuration**
   - Both – select **All**

If a node is highlighted when a `Push` operation is performed, then the selected information will be replicated to that Node only. If a Domain is highlighted when a `Push` operation is performed, then the selected information will be replicated to all nodes in that Domain.

17.4.1 Successful Push Operation

If a `Push` operation is unsuccessful, then the NEWS database will not contain the latest information about its Enterprise and Domain. Since Conductor

reads the NEWS database, the Destiny Administrator will not be able to perform day-to-day operations on Destiny objects that have not been replicated to the NEWS database through a Push operation. Therefore, a successful Push operation is imperative to Destiny operability.

Information related to the Push operation is logged in two log files—Netwat and UEDSERV. These files are located in the Destiny\stdlist directory structure.

The log files for a successful Push operation are as shown on the screens that follow:

```
Destiny\stdlist...NETWAT
NETWATCHER7708:16:20/INFO: WATCHER_STARTED sent to caller [3010.19]
NETWATCHER7708:16:20/INFO: Waiting for message from caller [3010.23]
NETWATCHER7708:16:20/INFO: Receiving DB_Push message [3010.27]
NETWATCHER7708:16:20/INFO: Waiting for import to terminate [3010.18]
NETWATCHER7708:16:20/INFO: Waiting for import to terminate [3010.18]
NETWATCHER7708:16:20/INFO: Waiting for message from caller [3010.23]
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* NetWat.c:2893: ERROR: SOCKET_ERROR [3004.12]
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* NetWat.c:243: ERROR: ReceiveData() failed [3004.3]
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* NetWat.c:245: ERROR: Terminating
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* NetWat.c:2927: ERROR: SOCKET_ERROR [3004.12]
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7708:16:20/* NetWat.c:2959: ERROR: SendData() failed [3004.8]
NETWATCHER7708:16:20/* ********************************************************
NETWATCHER7734:16:20/INFO: WATCHER_STARTED sent to caller [3010.19]
NETWATCHER7734:16:20/INFO: Waiting for message from caller [3010.23]
NETWATCHER7734:16:20/INFO: Receiving DB_Push message [3010.27]
NETWATCHER7734:16:20/INFO: Waiting for import to terminate [3010.18]
NETWATCHER7734:16:20/INFO: Waiting for import to terminate [3010.18]
NETWATCHER7734:16:20/INFO: Waiting for message from caller [3010.23]
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* NetWat.c:2893: ERROR: SOCKET_ERROR [3004.12]
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* NetWat.c:243: ERROR: ReceiveData() failed [3004.3]
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* NetWat.c:245: ERROR: Terminating
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* NetWat.c:2927: ERROR: SOCKET_ERROR [3004.12]
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* ********************************************************
NETWATCHER7734:16:20/* NetWat.c:2959: ERROR: SendData() failed [3004.8]
NETWATCHER7734:16:20/* ********************************************************
```

```
Destiny\stdlist...UEDSERV
Tivoli Destiny 1.1.0b10
Export version: 8
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.

 Exporting  tbl_users  .....
 Exporting  tbl_security_roles  .....
 Exporting  tbl_user_security_roles  .....
 Exporting  tbl_groups  .....
 Exporting  tbl_group_security_roles  .....
 Exporting  tbl_dist_list  .....
 Exporting  tbl_dist_list_details  .....
 Exporting  tbl_nodes  .....
 Exporting  tbl_node_communications  .....
 Exporting  tbl_destinations  .....
 Exporting  tbl_domains  .....
 Exporting  tbl_calendars  .....Tivoli Destiny 1.1.0b10
```

17.4.2 Failed Push Operation

If a Destiny Node is configured with the wrong Network Name, or a Destiny Node is configured using the Network Name of a non-Destiny computer, you will receive a `Push failed on following nodes` error for each Node configured incorrectly. The error header will read `Push UED Error` or `Push SCD Error`. The UEDSERV log will be updated with a successful `Push` operation; however, the Netwat log will not get updated with any information.

17.4.2.1 Solution

Verify that each Node for which you are receiving the above error message(s) is configured correctly. You should do the following:

1. Check the Node Properties in Composer. See if you can ping the Network Name of the problematic node.
2. Check to see if Netman is running on the problematic node. This can be done through either NT's Task Manager, or Control Panel -> Services.
3. If Microsoft SQL Server is being used, check to see if `MSSQL` is running on the problematic node.
4. From the Enterprise Server, see if you can ping the Network Name of the problematic node.

17.5 Unknown Log Problem Determination

The Unknown stdlist file will contain the daily activity of Spoolman and all of its child processes since the last time Spoolman was started. These processes are:

- Spoolman
- LQM
- NQM
- Dirwatch
- Logman
- Mapper
- Trashman

The format for most messages in this file is `PROCESS:HH:MM/INFO [mesgno]`. `PROCESS` is the background process generating the message. `HH:MM` is the time. `INFO` is a message generated by the process. `[mesgno]` is the message number. A typical message would look as follows:

```
SPOOLMAN:11:27/INFO:Starting Fileaid process to check consistency of
SPOOLMAN:11:27:database [3001.80]
```

When Destiny starts up, the unknown file will look like the example listed on the pages that follow. Although lengthy, these are the typical messages that will be seen when starting up the Spoolman process and all its child processes.

The fileaid process checks the database for inconsistencies. The fileaid errors generated up to, and including, Version 1.1.0 are normal for SQL installations. These errors will be eliminated in a future release.

The Trashman error near the end of this list is an actual error. If you were diagnosing the problem, then this would be information you would need to fix the problem.

The number of messages shown is based on the number of queues and Destinations configured on the server where Spoolman is running. If more queues and Destinations are configured, then you will see more messages.

```
Tivoli Destiny 1.1.0
Conductor version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
Tivoli Destiny 1.1.0
Spoolman version: 8
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
SPOOLMAN:07:14/INFO: Starting Fileaid process to check consistency of
SPOOLMAN:07:14/database [3001.80]
SPOOLMAN:07:14/INFO: Performing check FixJoes -- Check for JOEs in an
SPOOLMAN:07:14/invalid state.
SPOOLMAN:07:14/Selected JOEs will be set to Ready; Printing JOEs will be
SPOOLMAN:07:14/set to Error. [3022.4]
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/* fileaid.c:583: ERROR: OpenDatabase failed for , user  -
SPOOLMAN:07:14/* [3001.14]
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/* fileaid.c:583: ERROR: OpenDatabase failed for , user  -
SPOOLMAN:07:14/* [3001.14]
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/* ****************************************************************
*
* [23 DUPLICATE ENTRIES DELETED] *
*
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/* fileaid.c:583: ERROR: OpenDatabase failed for , user  -
SPOOLMAN:07:14/* [3001.14]
SPOOLMAN:07:14/* ****************************************************************
SPOOLMAN:07:14/INFO: Fileaid performed no actions on NEWS database [3001.83]
SPOOLMAN:07:14/INFO: Got watcher DemoInput [3001.45]
SPOOLMAN:07:14/INFO: Got watcher FAX [3001.45]
SPOOLMAN:07:14/INFO: Got watcher GHColor5M [3001.45]
SPOOLMAN:07:14/INFO: Got watcher GHlj4si [3001.45]
SPOOLMAN:07:14/INFO: Got watcher PagerIN [3001.45]
SPOOLMAN:07:14/INFO: Got watcher Reports [3001.45]
SPOOLMAN:07:14/INFO: Got queue DemoWeb [3001.57]
SPOOLMAN:07:14/INFO: Got queue DirectoryPush [3001.57]
SPOOLMAN:07:14/INFO: Got queue DMEMAIL [3001.57]
SPOOLMAN:07:14/INFO: Got queue DMFAX [3001.57]
SPOOLMAN:07:14/INFO: Got queue LJ5si [3001.57]
SPOOLMAN:07:14/INFO: Got queue Pager [3001.57]
SPOOLMAN:07:14/INFO: Got queue MailNotifyQ [3001.57]
SPOOLMAN:07:14/INFO: Got queue Notifications [3001.57]
SPOOLMAN:07:14/INFO: Got queue LJ4si [3001.57]
SPOOLMAN:07:14/INFO: Got queue GHColor [3001.57]
SPOOLMAN:07:14/INFO: Starting watcher DemoInput [3001.25]
SPOOLMAN:07:14/INFO: Starting watcher FAX [3001.25]
SPOOLMAN:07:14/INFO: Starting watcher PagerIN [3001.25]
SPOOLMAN:07:14/INFO: Starting watcher Reports [3001.25]
SPOOLMAN:07:14/INFO: Starting mapper process [3001.59]
SPOOLMAN:07:14/INFO: Starting NQM process [3001.60]
SPOOLMAN:07:14/INFO: Starting LQM for DemoWeb [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for DirectoryPush [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for DMEMAIL [3001.32]
```

```
SPOOLMAN:07:14/INFO: Starting LQM for DMFAX [3001.32]
SPOOLMAN:07:14/INFO: LQM not started for 9  [3001.61]
SPOOLMAN:07:14/INFO: Starting LQM for Pager [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for MailNotifyQ [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for Notifications [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for LJ4si [3001.32]
SPOOLMAN:07:14/INFO: Starting LQM for GHColor [3001.32]
SPOOLMAN:07:14/INFO: Starting TrashMan process [3001.62]
Tivoli Destiny 1.1.0
DirWatcher version: 5
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
DemoInput:07:14/INFO: OpenSession succeeded [3007.6]
Tivoli Destiny 1.1.0
DirWatcher version: 5
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
PagerIN:07:14/INFO: OpenSession succeeded [3007.6]
Tivoli Destiny 1.1.0
DirWatcher version: 5
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
Reports:07:14/INFO: OpenSession succeeded [3007.6]
Tivoli Destiny 1.1.0
DirWatcher version: 5
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
FAX:07:14/INFO: OpenSession succeeded [3007.6]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is DirectoryPush [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is MailNotifyQ [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is DemoWeb [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is DMEMAIL [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is DMFAX [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
```

```
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:14/INFO: My queue name is Pager [3002.13]
Tivoli Destiny 1.1.0
Mapper version: 13
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
MAPPER:07:15/mapper.c:5063: INFO: Mapper Process Started.. [3006.54]
Tivoli Destiny 1.1.0
NQM version: 6
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.

My Domain name is Austin
My Node name is Corpus_Christi
My Domain Manager is Macarena

NQM:07:15/INFO: Updating old assigned records to not assigned [3003.10]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:15/INFO: My queue name is Notifications [3002.13]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:15/INFO: My queue name is LJ4si [3002.13]
LQMDIRECTORYPUSH:07:15/INFO: This is not a forwarding queue [3002.27]
LQMDMEMAIL:07:15/INFO: This is not a forwarding queue [3002.27]
LQMPAGER:07:15/INFO: This is not a forwarding queue [3002.27]
LQMMAILNOTIFYQ:07:15/INFO: This is not a forwarding queue [3002.27]
LQMDIRECTORYPUSH:07:15/INFO: Got destination DIRECTORYPUSH [3002.30]
LQMDMEMAIL:07:15/INFO: Got destination DMEMAIL [3002.30]
Tivoli Destiny 1.1.0
LQM version: 10
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
LQM:07:15/INFO: My queue name is GHColor [3002.13]
LQMDEMOWEB:07:15/INFO: This is not a forwarding queue [3002.27]
LQMPAGER:07:15/INFO: Got destination Pager [3002.30]
LQMMAILNOTIFYQ:07:15/INFO: Got destination MailNotify [3002.30]
LQMDEMOWEB:07:15/INFO: Got destination DemoWeb [3002.30]
LQMDMFAX:07:15/INFO: This is not a forwarding queue [3002.27]
LQMNOTIFICATIONS:07:15/INFO: This is not a forwarding queue [3002.27]
LQMDMFAX:07:15/INFO: Got destination DMFAX [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination DemoWeb [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination DIRECTORYPUSH [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination DMEMAIL [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination DMFAX [3002.30]
LQMGHCOLOR:07:15/INFO: This is not a forwarding queue [3002.27]
LQMNOTIFICATIONS:07:15/INFO: Got destination lj4 [3002.30]
LQMLJ4SI:07:15/INFO: This is not a forwarding queue [3002.27]
LQMGHCOLOR:07:15/INFO: Got destination GHColor5M [3002.30]
```

```
Tivoli Destiny 1.1.0
Trashman version: 7
(C) Copyright Tivoli Systems, an IBM Company 1982, 1998
All rights reserved.
TRASHMAN:07:15/trashman.c:1058: INFO: Trashman Process Started.. [3009.54]
LQMNOTIFICATIONS:07:15/INFO: Got destination LJ5si [3002.30]
LQMLJ4SI:07:15/INFO: Got destination GHlj4si [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination Pager [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination MailNotify [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination GHlj4si [3002.30]
LQMNOTIFICATIONS:07:15/INFO: Got destination GHColor5M [3002.30]
TRASHMAN:07:16/* ************************************************************
TRASHMAN:07:16/* trashman.c:1318: ERROR: Cannot obtain Watcher File name
TRASHMAN:07:16/* [3009.14]
TRASHMAN:07:16/* ************************************************************
TRASHMAN:07:16/trashman.c:676: INFO: InActive SEL record updated with DONE
TRASHMAN:07:16/state for SEL ID 61 [3009.60]
TRASHMAN:07:16/trashman.c:676: INFO: InActive SEL record updated with DONE
TRASHMAN:07:16/state for SEL ID 62 [3009.60]
TRASHMAN:07:16/trashman.c:676: INFO: InActive SEL record updated with DONE
TRASHMAN:07:16/state for SEL ID 64 [3009.60]
TRASHMAN:07:16/trashman.c:676: INFO: InActive SEL record updated with DONE
TRASHMAN:07:16/state for SEL ID 65 [3009.60]
```

The following list of messages is an example of what you might see when shutting down Spoolman and all its processes. Again, if more queues and Destinations are configured on the server, then you will see more messages.

```
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue LJ5si terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
TERMINATING due to system shutdown event
TERMINATING due to system shutdown event
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue Pager terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue MailNotifyQ terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue Notifications terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue LJ4si terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: LQM for queue GHColor terminated [3001.47]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* INFO: Trashman terminated [3001.49]
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/* spoolman.c:281: ERROR: Terminating
SPOOLMAN:11:09/* ****************************************************************
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to DemoInput [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to FAX [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to PagerIN [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to Reports [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to DemoWeb [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to DirectoryPush [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to DMEMAIL [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to DMFAX [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to Mapper [3001.74]
SPOOLMAN:11:09/INFO: Sending STOP_PROCESS to NQM [3001.74]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher DemoInput [3001.67]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher FAX [3001.67]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher PagerIN [3001.67]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher Reports [3001.67]
SPOOLMAN:11:09/INFO: Waiting for termination of LQM for DemoWeb [3001.68]
SPOOLMAN:11:09/INFO: Waiting for termination of LQM for DirectoryPush
SPOOLMAN:11:09/[3001.68]
SPOOLMAN:11:09/INFO: Waiting for termination of LQM for DMEMAIL [3001.68]
SPOOLMAN:11:09/INFO: Waiting for termination of Mapper [3001.69]
SPOOLMAN:11:09/INFO: Waiting for termination of Nqm [3001.70]
MAPPER:11:09/* ****************************************************************
MAPPER:11:09/* mapper.c:4660: ERROR: Stop Message issued by parent process
MAPPER:11:09/* [3006.10]
MAPPER:11:09/* ****************************************************************
```

```
NQM:11:09/INFO: STOP_PROCESS received from SpoolMan [3030.4]
NQM:11:09/INFO: Stopping all NQMChildren [3003.9]
TERMINATING due to system shutdown event
NQM:11:09/* ********************************************************************
NQM:11:09/* nqm.c:420: ERROR: Terminating
NQM:11:09/* ********************************************************************
FAX:11:09/INFO: CloseSession succeeded [3007.7]
TERMINATING due to system shutdown event
PagerIN:11:09/INFO: CloseSession succeeded [3007.7]
Reports:11:09/INFO: CloseSession succeeded [3007.7]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher DemoInput [3001.67]
DemoInput:11:09/INFO: CloseSession succeeded [3007.7]
SPOOLMAN:11:09/INFO: Waiting for termination of watcher Reports [3001.67]
TERMINATING due to system shutdown event
```

The following example is what you would see when sending a document to a printer. Since Notifications is turned on, you will also see `LQMNOTIFICATIONS` displaying several messages as well. If Notifications was turned off, then you would not see these messages.

```
LQMLJ5SI:10:55/INFO: Sending command to device manager: %NF -jobid 23 -name
LQMLJ5SI:10:55/"Other.xls" -control
LQMLJ5SI:10:55/"d:\Destiny\spool\HP_LaserJet_5Si\07021998105539.ctl" -handle
LQMLJ5SI:10:55/"LJ5si" -data
LQMLJ5SI:10:55/"d:\Destiny\spool\HP_LaserJet_5Si\07021998105539.DAT"
LQMLJ5SI:10:55/[3002.19]

iJobID = 23
     Name = Other.xls
     Handle = LJ5si
     Data = d:\Destiny\spool\HP_LaserJet_5Si\07021998105539.DAT
     Control = d:\Destiny\spool\HP_LaserJet_5Si\07021998105539.ctl

LQMLJ5SI:10:55/INFO: Received message from device manager: %AK -jobid 23
LQMLJ5SI:10:55/[3002.20]
LQMLJ5SI:10:55/INFO: DM_ACKNOWLEDGE received [3002.37]
LQMNOTIFICATIONS:10:55/INFO: Setting status of destination ID 5 to b
LQMNOTIFICATIONS:10:55/[3002.21]
MAPPER:10:55/INFO: Document Other.xls owned by ldm from watcher
MAPPER:10:55/HP_LaserJet_5Si matched filter LJ5si-0 [3006.69]
MAPPER:10:55/INFO: Document Other.xls owned by ldm from watcher
MAPPER:10:55/HP_LaserJet_5Si - using handle LJ5si [3006.72]
MAPPER:10:55/mapper.c:2744: INFO: JOE created for SEL ID 101 [3006.50]
MAPPER:10:55/mapper.c:425: INFO: SEL record updated for SEL ID 101 [3006.51]
MAPPER:10:55/mapper.c:525: INFO: Active SEL record updated with MAPPED state
MAPPER:10:55/for SEL ID 101 [3006.52]
LQMLJ5SI:10:56/INFO: Received message from device manager: %OF -jobid 23
LQMLJ5SI:10:56/-pageno 0 [3002.20]
LQMLJ5SI:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
LQMLJ5SI:10:56/INFO: Received message from device manager: %OF -jobid 23
LQMLJ5SI:10:56/-pageno 100 [3002.20]
LQMLJ5SI:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
LQMLJ5SI:10:56/INFO: Received message from device manager: %DF -jobid 23
LQMLJ5SI:10:56/[3002.20]
LQMLJ5SI:10:56/INFO: DM_DONE_FILE received [3002.32]
LQMLJ5SI:10:56/INFO: Sending command to device manager: %AK -jobid 23
LQMLJ5SI:10:56/[3002.19]
LQMLJ5SI:10:56/INFO: Received message from device manager: %EX [3002.20]
LQMLJ5SI:10:56/INFO: Device Manager for LJ5si terminated successfully
LQMLJ5SI:10:56/[3001.52]
LQMLJ5SI:10:56/INFO: Setting status of destination ID 8 to r [3002.21]
LQMLJ5SI:10:56/INFO: Setting status of destination ID 8 to b [3002.21]


     GetCommandLocale() HP LaserJet 5Si
     GetCommandDevName() LJ5si
     GetCommandDevType() a
     GetCommandPersistFlag() 0
     GetCommandHelpFlag() 0
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %RY
LQMNOTIFICATIONS:10:56/-major 1 -minor 0 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_READY received [3002.38]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %NF -jobid
LQMNOTIFICATIONS:10:56/12 -name "Other.xls" -handle "LJ5si" -data
LQMNOTIFICATIONS:10:56/"d:\Destiny\spool
```

```
LQMNOTIFICATIONS:10:56/otify\07021998105549.NTF" -init
LQMNOTIFICATIONS:10:56/"d:\Destiny\init\dm-smtp.ini" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %AK
LQMNOTIFICATIONS:10:56/-jobid 12 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_ACKNOWLEDGE received [3002.37]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %OF
LQMNOTIFICATIONS:10:56/-jobid 12 -pageno 0 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %RQ
LQMNOTIFICATIONS:10:56/-jobid 12-sHandleDesc -sInputDevName -sDocumentName
LQMNOTIFICATIONS:10:56/-sFileName -sOutputFile -sCreationTime -sOwnerName
LQMNOTIFICATIONS:10:56/-sUserName -sFullName -sEmplNum -sAccountName -sTitle
LQMNOTIFICATIONS:10:56/-sGroupName -sBusAddress -sBusPhoneNum -sBusEmail
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sInputDevName "HP_LaserJet_5Si" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDocumentName "Other.xls" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sFileName "Other.xls" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOutputFile "Other.xls" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sCreationTime "1998-07-02 10:55:49" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOwnerName "ldm" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sUserName "ldm" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sFullName "Larry McWilliams" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sEmplNum "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sAccountName "ldm" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sTitle "Destiny Admin" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sGroupName "AdminGroup" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusAddress "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusPhoneNum "5124361842" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusEmail "larry.mcwilliams@tivoli.com" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusFax "4361899" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusPager "8887652019" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sBusBin "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherPhoneNum "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherEmail "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherFax "" [3002.19]
```

```
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherPager "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherBin "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -cConfFlag "n" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sOtherAddress "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultAddress "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultPhoneNum "5124361842" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultEmail "larry.mcwilliams@tivoli.com"
LQMNOTIFICATIONS:10:56/[3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultFax "4361899" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultPager "8887652019" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sDefaultBin "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteIPAddress "146.84.110.18" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteDNSName "lmcwilli.ausdev.tivoli.com"
LQMNOTIFICATIONS:10:56/[3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteWindowsName "LMCWILLI" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteConnectionMode "IPADDRESS,DNS,NET" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNotePortNumber "32223" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteProcessName "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %IN -jobid
LQMNOTIFICATIONS:10:56/12 -sNoteCookie "" [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Sending command to device manager: %EI -jobid
LQMNOTIFICATIONS:10:56/12 [3002.19]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %OF
LQMNOTIFICATIONS:10:56/-jobid 12 -pageno 30 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %OF
LQMNOTIFICATIONS:10:56/-jobid 12 -pageno 40 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
LQMNOTIFICATIONS:10:56/INFO: Received message from device manager: %OF
LQMNOTIFICATIONS:10:56/-jobid 12 -pageno 50 [3002.20]
LQMNOTIFICATIONS:10:56/INFO: DM_OUTPUT_FILE received [3002.36]
TRASHMAN:10:56/trashman.c:676: INFO: InActive SEL record updated with DONE
TRASHMAN:10:56/state for SEL ID 100 [3009.60]
  56
  56
LQMLJ5SI:10:56/INFO: Received message from device manager: %RY -major 1
LQMLJ5SI:10:56/-minor 0 [3002.20]
LQMLJ5SI:10:56/INFO: DM_READY received [3002.38]
```

17.6 Frequently Asked Questions

All FAQ data that is available is on the Tivoli web site. FAQs are updated on an hourly basis. A Tivoli customer can access the web site by going to http://www.support.tivoli.com/.

17.7 Error Solution Tables

The following section provides a quick reference for troubleshooting Output Manager/Destiny. Each process is listed in a separate table and includes the following information:

- Problem
- Symptom
- Error Messages
- Solution

The tables are listed in alphabetical order and include the following processes:

- Composer Configuration
- Conductor Configuration
- Dbase
- Destination Status
- Email
- LQM
- Mapper
- Netwatcher
- NQM
- NT Server
- Pager
- Print
- Spoolman
- Trashman

Table 25. Destiny Composer Configuration Issues

Problem	Symptom	Error Messages	Solution
The **Composer Configuration** window keeps popping up OR displays "*Connection to UED failed*."	The **Composer Configuration** window keeps popping up OR displays "*Connection to UED failed*."	`Connection to UED failed`	Check spelling of the **Network Name**. Be sure that it is a valid **UED** server name.

Problem	Symptom	Error Messages	Solution
The **Composer Configuration** window keeps popping up without errors.	The **Composer Configuration** window keeps popping up without errors.	None	Make sure **Netman** is running. If not, go to **Control Panel: Services** and start up **Destiny Netman** on 32222 or reboot PC.
An error message is displayed. The server could not be started on the *default domain* because information about the domain could not be obtained from **UED** after starting up **Composer**.	This indicates the default **Domain Name** entered in the **Composer Configuration** window was incorrect or invalid.	`Server could not be started on the default domain because information about the domain could not be obtained from UED.`	Check the default **Domain Name** entered in the **Composer Configuration** window.

Table 26. Destiny Conductor Configuration Issues

Problem	Symptom	Error Messages	Solution
Cannot log in to **Conductor** or **Composer**.	You receive an *INVALID USER ACCOUNT, PLEASE RE-ENTER* message when trying to log in to **Composer** or **Conductor**.	`INVALID USER ACCOUNT, PLEASE RE-ENTER`	Check that the *user name* and *password* are correct. Other things to check: 1) If this is a new install, make sure a **Composer Push All** has been done. 2) Use **Composer** to make sure the *user* is created and has the proper *password* assigned.
The **Conductor Configuration** window keeps popping up.	The **Conductor Configuration** window keeps popping up.	None	Check the default **Node Name**. It may be wrong, misspelled, non-existent, or the connection to the node is not working. Also, check the *node address*—it may be wrong.
Conductor won't start up.	**Conductor** won't start up.	None	Make sure that C:\UNISONHOME\unison.INI file exists.

Table 27. Destiny Dbase Issues

Problem	***Symptom***	***Error Messages***	***Solution***
You need to recover the **Destiny Databases**.	Your **Destiny Databases** have become corrupted, and you have cleaned one or more of them.	None	Destiny has three databases: **UED**, **SCD**, and **NEWS**. If the **NEWS** database gets corrupted, then this database is the easiest to recover. However, when you recover this database, you usually lose history records. Configuration information for **NEWS** can be easily recovered from the other two databases by doing a **Push**. If you have backups of these databases, then you can easily recover them. Again, you could lose some history, but you will get your databases back. Backups can be done with SQL, a third party backup software, or by using our **DBDATA** utility. If you lose your **UED** and/or **SCD** databases, and you don't have a backup, then the only way to recover them is to manually re-configure your Destiny environment. If your **NEWS** database was unaffected, then you can resume normal operations. However, you won't be able to make configuration changes to your Destiny domain until you recover the **UED** and/or the **SCD** databases.

Table 28. Destiny Destination Status Issues

Problem	**Symptom**	**Error Messages**	**Solution**
The **Destination** status remains in an ERROR state after starting **Spoolman**.	The **Destination** status remains in an ERROR state after starting **Spoolman**.	None	This condition is generally a result of a prior configuration problem. Stop the destination and then restart it. This usually corrects the problem unless the incorrect configuration issue still exists.

Table 29. Destiny E-mail Issues

Problem	Symptom	Error Messages	Solution
An error message is received when attempting to send a job to an **Email Destination**.	The error "*-errno 23002 -errmsg Unable to open output locale"* on email could be due to: a. You do not have an *email address* defined to the user. b. You specified, under your **Email Destination Properties**, an **SMTP Server Name** which does not conform.	`-errno 23002 -errmsg Unable to open output locale`	a. Define an *email address* to the user. b. You must specify, under your **Email Destination Properties**, an **SMTP Server Name** which conforms to the following: *dave.utuk.unison.com* Produces error message (Not defined in Hosts file). *dave.utuk.unison.com* Produces error message (Mapped to IP addr in Hosts file). *dav.utuk.unison.com* Works (Mapped to IP addr in Hosts file). *da.utuk.unison.com* Works (Mapped to IP addr in Hosts file). *hal.unison.com* Works (Not defined in Hosts file). *194.200.159.2* Works (IP addr of our SMTP server). Email Server Works (Mapped to IP addr in Hosts file). It would appear that we only allow for three characters to be in that fourth field.

Table 30. Destiny LQM Issues

Problem	Symptom	Error Messages	Solution

Destiny does not print a document.
—OR—
The document is printed to a different destination than expected.
	The Unknown log shows that a *Filter* was matched that is different than the one you expected.	`LQMWEBQ:10:03/INFO: This is not a forwarding queue [3002.27]`
		`LQMWEBQ:10:03/INFO: Got destination KerrysWeb [3002.30]`
		`KWDIRWAT:10:03/INFO: Record inserted for file boot.ini [3007.3]`
		`MAPPER:10:03/INFO: Document boot.ini owned by Everyone from watcher KWDIRWAT`
		`MAPPER:10:03/matched filter KerrysFilter [3006.69]`
		`MAPPER:10:03/INFO: Document boot.ini owned by Everyone from watcher KWDIRWAT`
		`MAPPER:10:03/- using handle KerrysHandle [3006.72]`
			From **Composer**, check your *filters* and make sure that they are set up correctly. **Watcher** names are case-sensitive. They must be entered into the **Filter Properties** just as they appear in the **Watcher Properties**.

Problem	Symptom	Error Messages	Solution
Destiny does not print anything.	Destiny does not print anything and appears to be in a hard loop. Stdlist displays "*The system cannot find the Locale specified*." This indicates the **System Name** under **Destination Properties** is incorrect or invalid.	LQMLJ5SI:09:27/INFO: Received message from device manager: %ER -jobid 0 LQMLJ5SI:09:27/-errno 23001 -errmsg The system cannot find the Locale LQMLJ5SI:09:27/specified. [3002.20] LQMLJ5SI:09:27/INFO: DM_ERROR received: %ER -jobid 0 -errno 23001 -errmsg LQMLJ5SI:09:27/The system cannot find the Locale specified. [3002.31] LQMLJ5SI:09:27/INFO: Sending command to device manager: %AK -jobid 0 LQMLJ5SI:09:27/[3002.19] LQMLJ5SI:09:27/* *********** LQMLJ5SI:09:27/* C:\spoolmate\develop\Lqm\lqm.c:3907: DB_ERROR: No Data LQMLJ5SI:09:27/* Found [3001.1] LQMLJ5SI:09:27/* *********** LQMLJ5SI:09:27/INFO: Received message from device manager: %EX [3002.20] LQMLJ5SI:09:27/INFO: Received message from device manager: %EX [3002.20] LQMLJ5SI:09:27/INFO: Received message from device manager: %RY -major 1 LQMLJ5SI:09:27/-minor 0 [3002.20] LQMLJ5SI:09:27/INFO: DM_READY received [3002.38] LQMLJ5SI:09:27/INFO: Sending command to device manager: %NF -jobid 319 -name LQMLJ5SI:09:27/"Test Page" -control "C:\Destiny\spool\HP_LaserJet_5Si\11101997092637.ctl" -handle LQMLJ5SI:09:27/"LJ5si" -data LQMLJ5SI:09:27/"C:\Destiny\spool\HP_LaserJet_5Si\11101997092637.DAT" -banner LQMLJ5SI:09:27/"C:\Destiny\banners\fyifax.emf LQMLJ5SI:09:27/INFO: Device Manager for LJ5si terminated successfully LQMLJ5SI:09:27/[3001.52] LQMLJ5SI:09:27/INFO: Setting status of destination ID 10 to r [3002.21]	If this is for a *printer*, then the **System Name** should exactly match the **NT Printer Name**. If this is for a *fax device*, then the **System Name** should be a valid COM port. If this is for an *email device*, then the **System Name** should be the proper mail server name. If this is for a *Web device*, then the **System Name** should be the proper path to the index.htm file: *C*:\UNISONHOME\Demo\Web\ index.htm. **NOTE:** This is a bug in Version B4.
An error in Unknown log.	An error in Unknown log.	LQMDEMOWEB:08:51/* *********** LQMDEMOWEB:08:51/* C:\spoolmate\develop\Lqm\lqm.c:3436: DB_ERROR: No Data LQMDEMOWEB:08:51/* Found [3001.1] LQMDEMOWEB:08:51/* ***********	This error is not catastrophic. However, it indicates a queue has been defined but has not been assigned a destination.

Table 31. Destiny Mapper Issues

Problem	Symptom	Error Messages	Solution
Destiny does not print a document.	An error in Unknown log.	MAPPER:09:30/* [Microsoft][ODBC SQL Server Driver][SQL Server]Attempt to MAPPER:09:30/* insert the value NULL into column 'owner_name', table MAPPER:09:30/* 'NEWS.dbo.tbl_messages'; column does not allow nulls. INSERT MAPPER:09:30/* fails. for in table	This error can be caused by incorrect configuration of a **Watcher** or by having an apostrophe in any of the fields of a Destiny object.
Destiny does not print anything.	Destiny does not print anything. There are errors in the Unknown log.	MAPPER:13:51/INFO: Document Microsoft Word - Document3 owned by ldm from MAPPER:13:51/watcher HP_LaserJet_5Si matched filter 5si-0 [3006.69] MAPPER:13:51/INFO: Document Microsoft Word - Document3 owned by ldm from [3006.72]51/watcher HP_LaserJet_5Si - using handle 5si MAPPER:13:51/* ************* MAPPER:13:51/* C:\spoolmate\develop\Mapper\mapper.c:1218: INFO: No matching MAPPER:13:51/* Calendar found in MAPPER CALENDAR table for Mapper ID 2 MAPPER:13:51/* [3006.42] MAPPER:13:51/* ************* MAPPER:13:51/* ************* MAPPER:13:51/* C:\spoolmate\develop\Mapper\mapper.c:992: ERROR: No Data MAPPER:13:51/* Found for Handle 5si in table MAPPERS [3006.31] MAPPER:13:51/* ************* MAPPER:13:51/C:\spoolmate\develop\Mapper\mapper.c:522: INFO: Active SEL MAPPER:13:51/record updated with ERROR state for SEL ID 19 [3006.53]	A *calendar error* indicates when a date is not found in the calendar. This is not catastrophic. However, the document will **not** be output by Destiny when it can't find a matching calendar date.
An error in Unknown log.	An error in Unknown log.	MAPPER:14:15/WARNING: No filter matched for SEL with filename Kathleen MAPPER:14:15/Gesford, 05:32 PM 9/14 , username ldm and devicename MAPPER:14:15/HP_LaserJet_5Si [3006.30] MAPPER:14:15/* ************* MAPPER:14:15/* C:\spoolmate\develop\Mapper\mapper.c:1891: ERROR: No Data MAPPER:14:15/* Found for QueueName unknown in table QUEUES [3006.33] MAPPER:14:15/* *************	Generally, this error message indicates insufficient configuration for a **Queue**. Appropriate *mappers* and *filters* may not have been set up for the **Queue**. —OR— There is no destination assigned to the **Queue**.

Problem	Symptom	Error Messages	Solution
The *spool file* doesn't print.	Filtering of the *spool file* maps to a user.	MAPPER:14:01/INFO: Document Document owned by nflowers from watcher HPLJ2 MAPPER:14:01/matched filter email [3006.69] MAPPER:14:01/INFO: Document Document owned by nflowers from watcher HPLJ2 - MAPPER:14:01/using handle email [3006.72] MAPPER:14:01/* ********** MAPPER:14:01/* mapper.c:2165: ERROR: Invalid Queue specified [3006.20] MAPPER:14:01/* ********** MAPPER:14:01/* ********** MAPPER:14:01/* mapper.c:3608: ERROR: Cannot create JOE [3006.25] MAPPER:14:01/* ********** MAPPER:14:01/mapper.c:627: INFO: Active SEL record updated with ERROR state MAPPER:14:01/for SEL ID 57 [3006.53]	The user doesn't have a **Queue** defined in their profile. The default **Queue** can be defined in the *user's profile* or in the **Group** assigned to the user.
The *spool file* doesn't print.	The *spool file* is filtered through a mapper that has a calendar that isn't every day.	MAPPER:21:35/INFO: Document test - Notepad owned by nflowers from watcher MAPPER:21:35/HPLJ2 matched filter test [3006.69] MAPPER:21:35/INFO: Document test - Notepad owned by nflowers from watcher MAPPER:21:35/HPLJ2 - using handle hplj2handle [3006.72] MAPPER:21:35/* ********** MAPPER:21:35/* mapper.c:3947:ERROR: Invalid expiration date for calendar MAPPER:21:35/* nicole [3006.80] MAPPER:21:35/* ********** MAPPER:21:35/* mapper.c:1355: INFO: No matching Calendar found in MAPPER MAPPER:21:35/* CALENDAR table for Mapper ID 2 [3006.42] MAPPER:21:35/* ********** MAPPER:21:35/* mapper.c:1129: ERROR: No Data Found for Handle hplj2handle in MAPPER:21:35/* table MAPPERS [3006.31] MAPPER:21:35/* ********** MAPPER:21:35/mapper.c:627: INFO: Active SEL record updated with ERROR state MAPPER:21:35/for SEL ID 85 [3006.53]	The date on which the *spool file* is printing isn't allowed. Change the expiration date on the calendar. —OR— Choose another mapper to route through.

Table 32. Destiny Netwatcher Issues

Problem	Symptom	Error Messages	Solution
Push fails to update the **NEWS** database.	Push fails to update the **NEWS** database. There are IMPORT errors in the **Unknown** log.	NETWATCHER:13:57/INFO: WATCHER_STARTED sent to caller [3010.19] NETWATCHER:13:57/INFO: Waiting for message from caller [3010.23] NETWATCHER:13:57/INFO: Receiving DB_PUSH message [3010.27] NETWATCHER:13:57/INFO: Waiting for import to terminate [3010.18] NETWATCHER:13:57/INFO: Waiting for import to terminate [3010.18] IMPORT:13:57/* ********** IMPORT:13:57/* C:\spoolmate\develop\Import\filetodb.c:468: ERROR: IMPORT:13:57/* [Microsoft][ODBC SQL Server Driver][SQL Server]Line 1: IMPORT:13:57/* Incorrect syntax near 's'. [3020.9] IMPORT:13:57/* ********** IMPORT:13:57/* ********** IMPORT:13:57/* C:\spoolmate\develop\Import\import.c:227: ERROR: Error while IMPORT:13:57/* importing tbl_nodes table [3020.3] IMPORT:13:57/* ********** IMPORT:13:57/* ********** IMPORT:13:57/* C:\spoolmate\develop\Import\import.c:126: ERROR: ImportData() IMPORT:13:57/* failed in main() function [3020.7] IMPORT:13:57/* ********** NETWATCHER:13:57/* ********** NETWATCHER:13:57/* D:\spoolmate\develop NETWATCHER:13:57/* etwat NETWATCHER:13:57/* etWat.c:849: ERROR: DbImport terminated with FAILURE. NETWATCHER:13:57/* [3010.9] NETWATCHER:13:57/* ********** NETWATCHER:13:57/INFO: Waiting for message from caller [3010.23] NETWATCHER:13:57/* **********	This error causes Push to not update configuration information in the NEWS database. The error is the result of creating a node with a description containing an apostrophe. Remove the apostrophe, and the error will be eliminated.

Table 33. Destiny NQM Issues

Problem	Symptom	Error Messages	Solution
Spoolman will not start up.	**Spoolman** will not start up.	NQM:08:47/* ********** NQM:08:47/* C:\spoolmate\develop NQM:08:47/* qm NQM:08:47/* qm.c:926: DB_ERROR: No Data Found [3001.1] NQM:08:47/* ********** NQM:08:47/* ********** NQM:08:47/* C:\spoolmate\develop NQM:08:47/* qm NQM:08:47/* qm.c:111: ERROR: Can not read own Nodes record [3003.5] NQM:08:47/* ********** NQM:08:47/* ********** NQM:08:47/* C:\spoolmate\develop NQM:08:47/* qm NQM:08:47/* qm.c:112: ERROR: Terminating NQM:08:47/* **********	Generally, these error messages are a result of an incorrect or invalid **Domain Name** in the *C*:\UNISONHOME\unison.INI file. The result is **Spoolman** will not start up.

Table 34. Destiny Pager Issues

Problem	Symptom	Error Messages	Solution

The pager does not answer, even though you can hear it ringing on your modem.	You send a page out, and while listening to the modem, all you hear is a ring and no answer.	Below is what you will see in the *Pager.DBG* file if a comma is inserted in the *LocalOutSideLine* definition:
		Pager.DBG (stdlist file found in stdlist directory):
		`COM1: - Dialing 9,,-873-8719`
		`COM1: IN:ATDT9,,-873-8719`
		`NO CARRIER`
		`Timeout waiting for CONNECTion.`
		`COM1: IN T/O:`
		`COM1: IN T/O:`
		`COM port successfully closed`
		`Returning with error - Timeout waiting for CONNECTion.`
		`Initializing Port`
		`Size of PORT structure: 6188`
		`Set modem using AT~`
		`COM1: IN:AT`
		`OK`
			Check the *dm-pager.INI* file. The definition for *LocalOutSideLine* should have just a number **9**. No comma should follow the number **9**. If there is a comma, remove it.

Table 35. Destiny Print Issues

Problem	Symptom	Error Messages	Solution
Destiny does not print a document.	You tried to print a document, but it never printed. Stdlist shows nothing.	None	It is most likely that **Spoolman** is not running. Using **Conductor**, start up **Spoolman**.
Destiny does not print a document.	You tried to print a document, but it never printed. Stdlist shows the document got mapped, but that's all.	None	Check for the following: a) Status of **Destination**. If stopped, start it. b) Status of **Queue**. If down, start it. c) Status of **Watcher**. If down, start it. d) Check order of *Filters*. View Stdlist to see which *Filter* the document was mapped to. Correct as needed. This may involve correcting the *mapper* configuration and/or the filter configuration. e) Check the file size. If destinations are configured based on file size, then the windows file size may not agree with what Destiny thinks the file size is. If this is the case, then change the file size configuration to the appropriate values using **Composer**. Do a **Push ALL** to get your document to print.

Table 36. Destiny Spoolman Issues

Problem	Symptom	Error Messages	Solution
Destiny does not print a document.	An error in Unknown log.	Unison SpoolMate 1.0 (C) Unison Software. Inc. Conductor version: 10 Unison SpoolMate 1.0 (C) Unison Software. Inc. Spoolman version: 7 SPOOLMAN:09:30/INFO: Got watcher KerrysWatcher [3001.45] SPOOLMAN:09:30/INFO: Got queue KerrysQueue [3001.57] SPOOLMAN:09:30/INFO: Starting watcher KerrysWatcher [3001.25] SPOOLMAN:09:30/INFO: Starting mapper process [3001.59] GetCommandPersistFlag() 0 GetCommandHelpFlag() 0 %ER -jobid 0 -errno 23001 -errmsg The system cannot find the Locale specified. %EX SPOOLMAN:09:30/INFO: Starting NQM process [3001.60] SPOOLMAN:09:30/INFO: Starting LQM for KerrysQueue [3001.32] SPOOLMAN:09:30/INFO: Starting TrashMan process [3001.62] SPOOLMAN:09:30/* ********** SPOOLMAN:09:30/* INFO: Watcher KerrysWatcher terminated [3001.46] SPOOLMAN:09:30/* **********	A destination was configured that is a shared printer on another server. The **Destiny Output Server** must be the *last spooler* before the printer. You cannot set up an output server with a destination that is a shared printer on another server.

Table 36. Destiny Spoolman Issues

Problem	Symptom	Error Messages	Solution
Destiny does not print a document.	An error in Unknown log.	Unison SpoolMate 1.0 (C) Unison Software. Inc. Conductor version: 10 Unison SpoolMate 1.0 (C) Unison Software. Inc. Spoolman version: 7 SPOOLMAN:09:30/INFO: Got watcher KerrysWatcher [3001.45] SPOOLMAN:09:30/INFO: Got queue KerrysQueue [3001.57] SPOOLMAN:09:30/INFO: Starting watcher KerrysWatcher [3001.25] SPOOLMAN:09:30/INFO: Starting mapper process [3001.59] GetCommandPersistFlag() 0 GetCommandHelpFlag() 0 %ER -jobid 0 -errno 23001 -errmsg The system cannot find the Locale specified. %EX SPOOLMAN:09:30/INFO: Starting NQM process [3001.60] SPOOLMAN:09:30/INFO: Starting LQM for KerrysQueue [3001.32] SPOOLMAN:09:30/INFO: Starting TrashMan process [3001.62] SPOOLMAN:09:30/* ***************** SPOOLMAN:09:30/* INFO: Watcher KerrysWatcher terminated [3001.46] SPOOLMAN:09:30/* ***************** Unison SpoolMate 1.0 (C) Unison Software. Inc. LQM version: 9 LQM:09:30/INFO: My queue name is KerrysQueue [3002.13] Unison SpoolMate 1.0 (C) Unison Software. Inc. Trashman version: 7 TRASHMAN:09:30/D:\spoolmate\develop\trashman\trashman.c:1057: INFO: Trashman TRASHMAN:09:30/Process Started.. [3009.54] Unison SpoolMate 1.0 (C) Unison Software. Inc. Mapper version: 13 MAPPER:09:30/D:\spoolmate\develop\mapper\mapper.c:4309: INFO: Mapper Process MAPPER:09:30/Started.. [3006.54] Unison SpoolMate 1.0 (C) Unison Software. Inc. NQM version: 6 NQM:09:30/INFO: Updating old assigned records to not assigned [3003.10] LQMKERRYSQUEUE:09:30/INFO: This is not a forwarding queue [3002.27] LQMKERRYSQUEUE:09:30/INFO: Got destination Kerrys5si [3002.30] LQMKERRYSQUEUE:09:30/INFO: STOP_PROCESS received from SpoolMan [3030.4] LQMKERRYSQUEUE:09:30/* *************** LQMKERRYSQUEUE:09:30/* INFO: Terminating normally [3002.15] LQMKERRYSQUEUE:09:30/* *************** MAPPER:09:30/D:\spoolmate\develop\mapper\mapper.c:2324: INFO: JOE created MAPPER:09:30/for SEL ID 11 [3006.50] MAPPER:09:30/* ****************** MAPPER:09:30/*	A **Watcher** was configured incorrectly. In this case, **KerrysWatcher** was configured with the *dm-splnt.exe* as its executable file. Upon starting **Spoolman**, the **Watcher** is found and a start is attempted. Since the *dm-splnt.exe* is an invalid field for the **Watcher**; *errno 23001* is logged. As a secondary symptom, the following error is logged: MAPPER:09:30/* [Microsoft][ODBC SQL Server Driver][SQL Server]Attempt toMAPPER:09:30/* insert the value NULL into column 'owner_name', table MAPPER:09:30/* 'NEWS.dbo.tbl_messages'; column does not allow nulls. INSERT MAPPER:09:30/* fails. for in table This error is described as a separate problem.

Table 36. Destiny Spoolman Issues

Problem	Symptom	Error Messages	Solution
		D:\spoolmate\develop\mapper\mapper.c:1648: ERROR: MAPPER:09:30/* [Microsoft][ODBC SQL Server Driver][SQL Server]Attempt to MAPPER:09:30/* insert the value NULL into column 'owner_name', table MAPPER:09:30/* 'NEWS.dbo.tbl_messages'; column does not allow nulls. INSERT MAPPER:09:30/* fails. for in table MAPPER:09:30/* ****************** SPOOLMAN:09:30/* ****************** SPOOLMAN:09:30/* INFO: Mapper terminated [3001.48] SPOOLMAN:09:30/* ****************** SPOOLMAN:09:30/INFO: Sending STOP_PROCESS to LQM for KerrysQueue [3001.37] SPOOLMAN:09:30/INFO: Sending STOP_PROCESS to Nqm [3001.64] SPOOLMAN:09:30/INFO: Sending STOP_PROCESS to TrashMan [3001.65] SPOOLMAN:09:30/INFO: Waiting for termination of Nqm [3001.70]	
		NQM:09:30/INFO: STOP_PROCESS received from SpoolMan [3030.4] NQM:09:30/INFO: Stopping all NQMChildren [3003.9] NQM:09:30/* ****************** NQM:09:30/* C:\spoolmate\develop NQM:09:30/* qm NQM:09:30/* qm.c:288: ERROR: Terminating NQM:09:30/* ****************** TRASHMAN:09:30/* ****************** TRASHMAN:09:30/* D:\spoolmate\develop\trashman\trashman.c:937: ERROR: Stop TRASHMAN:09:30/* Message issued by parent process [3009.10] TRASHMAN:09:30/* ****************** TRASHMAN:09:30/* ****************** TRASHMAN:09:30/* D:\spoolmate\develop\trashman\trashman.c:972: ERROR: TRASHMAN:09:30/* Exiting Trashman Process ... [3009.1] TRASHMAN:09:30/* ****************** SPOOLMAN:09:30/* ****************** SPOOLMAN:09:30/* C:\spoolmate\develop\Spoolman\spoolman.c:217: ERROR: SPOOLMAN:09:30/* Terminating SPOOLMAN:09:30/* ******************	
An error in Unknown log.	An error in Unknown log.	SPOOLMAN:16:07/+ ++++++++++++++++++ SPOOLMAN:16:07/+ C:\spoolmate\develop\Spoolman\spoolman.c:1833: ERROR: SPOOLMAN:16:07/+ CreateProcess() failed for C:\Destiny\LJ4m6w -n LJ4west SPOOLMAN:16:07/+ [3001.6] SPOOLMAN:16:07/+ ++++++++++++++++++	The error "*CreateProcess Failed*" indicates an invalid or incorrect **Watcher** definition.

Table 36. Destiny Spoolman Issues

Problem	Symptom	Error Messages	Solution
An error in Unknown log.	An error in Unknown log.	SPOOLMAN:08:02/* ********** SPOOLMAN:08:02/* C:\spoolmate\develop\Spoolman\spoolman.c:133: ERROR: SPOOLMAN:08:02/* OpenDatabase failed for , user - [Microsoft][ODBC Driver SPOOLMAN:08:02/* Manager] Data source name not found and no default driver SPOOLMAN:08:02/* specified [3001.14] SPOOLMAN:08:02/* ********** SPOOLMAN:08:02/* ********** SPOOLMAN:08:02/* C:\spoolmate\develop\Spoolman\spoolman.c:134: ERROR: SPOOLMAN:08:02/* Terminating SPOOLMAN:08:02/* **********	This error indicates there is a problem with the *C*:\UNISONHOME\ Spoolmate.INI file. This usually occurs after attempting to start a node from **Conductor**.
Destiny does not print a document.	This only happens at certain times of the day.	SPOOLMAN:17:30/WARNING: Sending STOP_PROCESS to LQM for LJ5si as queue SPOOLMAN:17:30/period is over [3001.44] SPOOLMAN:17:30/INFO: Waiting for termination of LQM for LJ5si [3001.68] LQMLJ5SI:17:30/INFO: STOP_PROCESS received from SpoolMan [3030.4] LQMLJ5SI:17:30/* ********** LQMLJ5SI:17:30/* INFO: Terminating normally [3002.15] LQMLJ5SI:17:30/* **********	The solution here is to: 1) Reconfigure the **Queue** to operate at a different set of hours. —OR— 2) Wait for the **Queue** time to be valid. —OR— 3) Cut and Paste the document that is not printing into another **Queue** that does not have a configured time restriction.

Table 36. Destiny Spoolman Issues

Problem	Symptom	Error Messages	Solution
Destiny does not print a document.	A document does not print. You get a Windows dialog box indicating there is a problem with the printer.	LQMLJ5SI:10:55/INFO: Received message from device manager: %AK -jobid 14 LQMLJ5SI:10:55/[3002.20] LQMLJ5SI:10:55/INFO: DM_ACKNOWLEDGE received [3002.37] LQMLJ5SI:10:55/INFO: Received message from device manager: %OF -jobid 14 LQMLJ5SI:10:55/-pageno 0 [3002.20] LQMLJ5SI:10:55/INFO: DM_OUTPUT_FILE received [3002.36] LQMLJ5SI:11:08/INFO: Received message from device manager: %ER -jobid 14 LQMLJ5SI:11:08/-errno 13014 -errmsg The printer reported an error for this LQMLJ5SI:11:08/job [3002.20] LQMLJ5SI:11:09/INFO: DM_ERROR received: %ER -jobid 14 -errno 13014 -errmsg LQMLJ5SI:11:09/The printer reported an error for this job [3002.31] LQMLJ5SI:11:09/INFO: Sending command to device manager: %AK -jobid 14 LQMLJ5SI:11:09/[3002.19] LQMLJ5SI:11:09/INFO: Received message from device manager: %EX [3002.20] LQMLJ5SI:11:09/INFO: Device Manager for LJ5si terminated successfully LQMLJ5SI:11:09/[3001.52] LQMLJ5SI:11:09/INFO: Setting status of destination ID 8 to r [3002.21]	The solution here is to: 1) Correct the hardware problem. —OR— 2) Cancel printing of the document by pressing the **Cancel** button in the NT dialog box.

Table 37. Destiny Trashman Issues

Problem	Symptom	Error Messages	Solution
An error in Unknown log.	An error in Unknown log.	TRASHMAN:10:28/* ********** TRASHMAN:10:28/* D:\spoolmate\develop\trashman\trashman.c:112: ERROR: Cannot TRASHMAN:10:28/* compress the file specified [3009.33] TRASHMAN:10:28/* ********** TRASHMAN:10:28/NO MSG: D:\spoolmate\develop\trashman\trashman.c,375,78,, TRASHMAN:10:28/catgets failed 0 [3009.62]	This error results from not being able to find the directory to archive to.

Chapter 18. Remote Control

Tivoli Remote Control is used to take control of any Intel-based system in the Tivoli environment from another Intel-based machine in the Tivoli environment. The Remote Control components are:

- Tivoli Remote Control Server
- Tivoli Remote Control Target
- Tivoli Remote Control Controller

Together, these components add Remote Control capabilities to support most configurations. Installing a component on a machine in the Tivoli environment assigns one or more specific *roles* to that machine. Tivoli Remote Control operates with the following roles:

Target A machine that can be controlled from a Controller.

Controller A machine that has the capabilities to take control of targets.

Gateway A relay-station used to control the TCP/IP flow and optionally do protocol conversion between TCP/IP and SPX/IPX protocols. Note that this is distinct from the gateway that manages TMA endpoints.

Server The system that controls the Remote Control environment.

The minimum usable configuration involves at least one server, one controller, and one target.

The mechanism used to ensure that the user at the controller is authorized to take control over the target is implemented using the Tivoli Framework. The same applies to the initialization of sessions between the controller and target. This implies, that some sort of Framework stub - TMA, Tivoli PC Agent, or Tivoli Framework - has to be present on all nodes in the Remote Control environment.

From a controller, the user can issue remote commands on a target, much like the TCP/IP `rexec` service, or activate a remote control session. To do the latter, the user at the controller uses the Tivoli Desktop to select the target for the session, parameters to control the session initialization, and to ask the Remote Control server to initialize the session.

Refer to the *Tivoli Remote Control User's Guide* for a further description.

18.1 Tivoli Remote Control Installation

To implement Tivoli Remote Control in the Tivoli environment, at least one Remote Control server, one Remote Control controller, and one Remote Control target has to be installed.

The Remote Control server ***must*** be installed on the TMR server, and if TMA endpoints are to be supported, on any gateway in the Tivoli Framework.

All components except the Remote Control Server can be installed on any type of Tivoli endpoint as long as they are based on the Intel platform. Table 38 shows the supported platforms for each component:

Table 38. Remote Control - 3.6 Supported Platforms

		Server	Controller	Target
Managed Node	UNIX	✓	N/A	N/A
	Windows	✓	✓	✓
	OS/2	N/A	✓	✓
PC Managed Node	UNIX	N/A	N/A	N/A
	Windows	N/A	✓	✓
	OS/2	N/A	✓	✓
TMA Endpoint	UNIX	N/A	N/A	N/A
	Windows	N/A	✓	✓
	OS/2	N/A	✓	✓

In Table 38, *Windows* refers to all supported versions of Microsoft Windows 3.1x, Windows 95 and 98, as well as Windows NT. Consult the *Tivoli Remote Control User's Guide* and *Release Notes* for further details on hardware and software requirements for each platform.

The installation process of Remote Control components varies slightly between the different types of endpoints:

Managed Nodes Installation on TMR servers, managed nodes, and PC managed nodes is similar to that of most other Tivoli applications. One or more Remote Control components can be installed using one of the three standard ways of installation - SIS, Tivoli GUI, or the `winstall` command.

TMA Endpoints The Remote Control components have to be installed locally. From the product CD, the `SETUP.EXE` program is used on windows endpoints. For OS/2, the installation is initiated using `INSTALL.EXE` on the CD.

Before installing Remote Control components, the *Tivoli Remote Control User's Guide*, as well as the *Remote Control Release Notes*, should be consulted.

18.1.1 Patches

At the time of writing, a patch for Tivoli Remote Control was available to correct various errors for Remote Control on Windows 95 and Windows 98. This can be obtained from the Tivoli Support ftp-site at:

`ftp://ftp.tivoli.com/support/patches/patches_3.6/3.6-RCL-0001/`

Contact your Tivoli representative if you have problems obtaining this patch or if you need more information about it.

18.2 Preparing to Use Remote Control

Upon installation, a few actions have to be completed to set up Remote Control to be used in the Tivoli environment. These are:

1. Authorize administrators.
2. Create one or more RemoteControl objects.
3. Set default policies.

18.2.1 Authorizing Administrators

Installing the Remote Control server adds four new Tivoli administrator roles, some of which have to be assigned to the Tivoli administrators that will be using Remote Control. The four roles are:

remote_control Enables the administrator to monitor the actions, to control the mouse actions, and to enter keyboard data on a remote workstation.

remote_monitor Enables the administrator to monitor the actions only on a remote workstation.

remote_probe Enables the administrator to run commands on a target from the command line interface.

remote_reboot Enables the administrator to restart a remote workstation.

When planning the authorization schema for Remote Control in the TMR, remember that specific roles can be assigned to administrators on a TMR-wide basis or for specific policy regions. With Remote Control, more administrators are typically added, requiring access only to a few Remote Control objects, each covering a subset of workstations in the organization.

Note

Administrators will only be allowed to take control over endpoints that are defined in the policy region(s) for which the administrator has been assigned one or more Remote Control roles unless the administrator has been granted TMR wide Remote Control roles.

18.2.2 Creating the RemoteControl Object

In each policy region, in which a RemoteControl object will be created, the object type RemoteControl has to be added to the list of managed resources for that region.

Having done so, the RemoteControl object can be created. This will be used by the administrators, that have been assigned a remote control role, to control session settings and initialize a session with an endpoint.

Creating the RemoteControl object can only be done through GUI using the dialog shown in Figure 226:

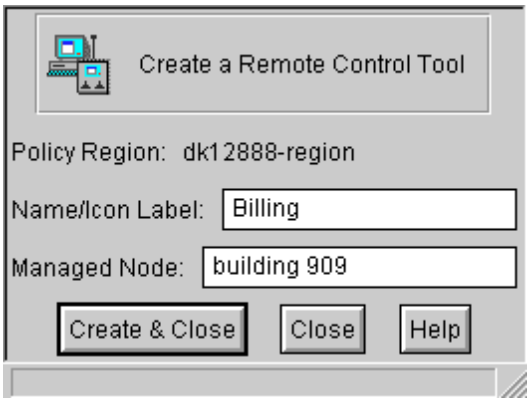


Figure 226. Remote Control Creation Dialog

The managed node specified when creating the RemoteControl object will be the node that controls all session initializations through the specific

RemoteControl object. In order to fulfill this task, the Remote Control Server has to be installed on the specified managed node.

18.2.3 Setting Default Policies

During installation of the Remote Control Server component on the TMR server, a Remote Control default policy object - RemoteControl_PDO - is built. This holds the default policies for all RemoteControl objects within the TMR. No validation policies are available for the RemoteControl object; hence, the default policies are the only way to control the way the session control parameters are manipulated by the authorized Remote Control administrators, and this will be on a TMR-wide basis.

The fact that the default policy applies to all RemoteControl objects within a policy region, or in the whole TMR, if no special policy objects have been created, means that implementation of each individual policy has to be considered and planned carefully. Remember that the endpoints available to each administrator, if the administrators are not granted TMR-wide authorizations, are those defined in the same policy regions as the RemoteControl object used to initialize the session.

To provide different customization to various Remote Control objects, they should be placed in different policy regions. For each region, a new Remote Control default policy object can be created and attached to the policy region.

To create a new default policy object, the following command can be used:

```
wcrtpol -d RemoteControl <new_default_RemoteControl_policy_object_name>
RemoteControl_PDO
```

The new Remote Control default policy object inherits all the policies of the Tivoli-provided Remote Control policy object (`RemoteControl_PDC`).

The new default policy object will have to be assigned to the RemoteControl objects of the policy region(s) where it will be used. In the policy region window, select **Properties - Managed Resource Policies...** and specify the new default policy for the region, as shown in Figure 227 on page 580:

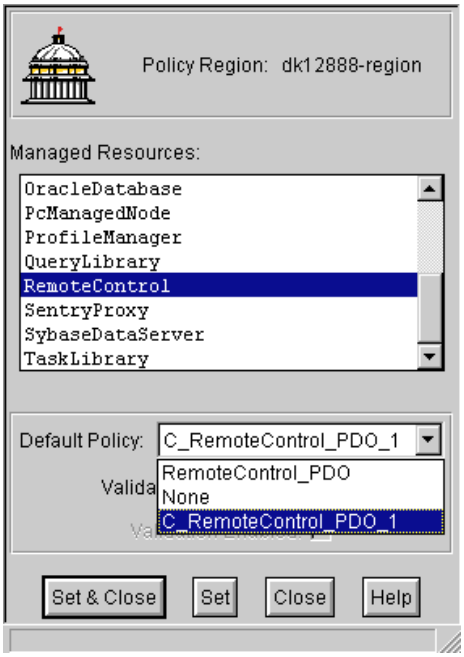


Figure 227. Setting New Remote Control Default Policy for a Policy Region

The new default policy object can now be tailored to the specific needs for each policy in the object.

TMR Default Policy Defaults

Remember that the base default policy object, `RemoteControl_PDO`, acts as the model from which the new default policy objects are created. This means that only policies that should be different from the standard policies should be implemented in the new default policy object(s).

In order to make changes to default policies for the entire TMR, the selected policy can be changed in the base default policy object, `RemoteControl_PDO`. If you change the base default policy object, you need to be sure that future product upgrades do not overwrite it.

The following default policies can be applied to Remote Control default policy objects:

`rc_def_alt_t` Determines whether the remote target user is able to press the <Alt+t> keys to change the session state.

`rc_def_command` Determines whether the default action that appears on the Remote Control dialog is to control, monitor, or reboot the target.

`rc_def_define` Determines whether the user can filter the client list by using the `rc_def_targets` policy method.

`rc_def_targets` Allows the user to define a list of clients.

`rc_def_filter_mode` Determines whether to display managed nodes, NetWare clients, PC managed nodes, TMAs, or all of the above.

`rc_def_polfilter_mode` Determines whether to display the client list for a specific region.

`rc_def_grace_time` Determines how many seconds to wait for the user to respond before beginning a remote control session.

`rc_def_timeout_op` Determines whether to begin a remote session if the user does not respond within the grace period.

`rc_def_gw` Determines whether to use the gateway between the controller and the target.

`rc_def_ports` Determines the port numbers that the target and the controller use to communicate.

`rc_def_backgrnd` Determines whether to disable the desktop background of the target workstation when the session starts.

`rc_def_color` Determines whether to limit to 16 colors the target screen displayed on the controller.

`rc_def_comp` Determines whether to compress the data transmitted to the controller.

`rc_def_rate` Determines the amount of time between the refreshes of the screen displayed on the controller.

Refer to *Tivoli Remote Control User's Guide* for details on each individual policy and which values are valid.

The policies are used to control the behavior of the panels used by users at Remote Control controller workstations when initiating a remote control session. The appearance of the panel depends on the value returned by the `rc_def_define` policy. This can take two values: DefinableTableList or FilteredList. Figure 228 on page 582 shows the panel, and related policies,

for a `rc_def_define` value of **FilteredList**, and Figure 229 on page 583 shows the same information for a value of **DefinableTargetList**.

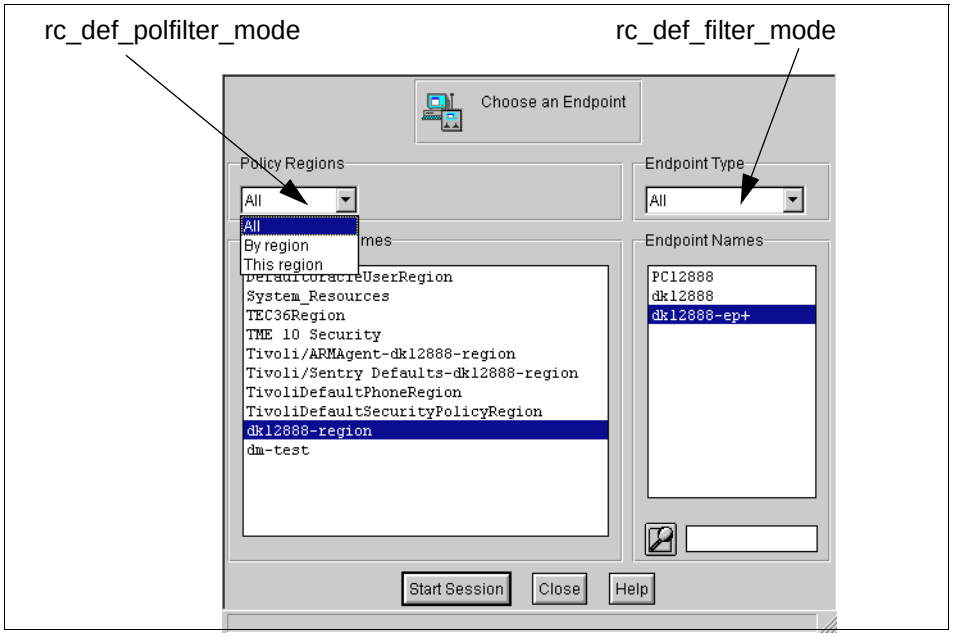


Figure 228. Remote Control FilteredList Endpoint Selection Policies

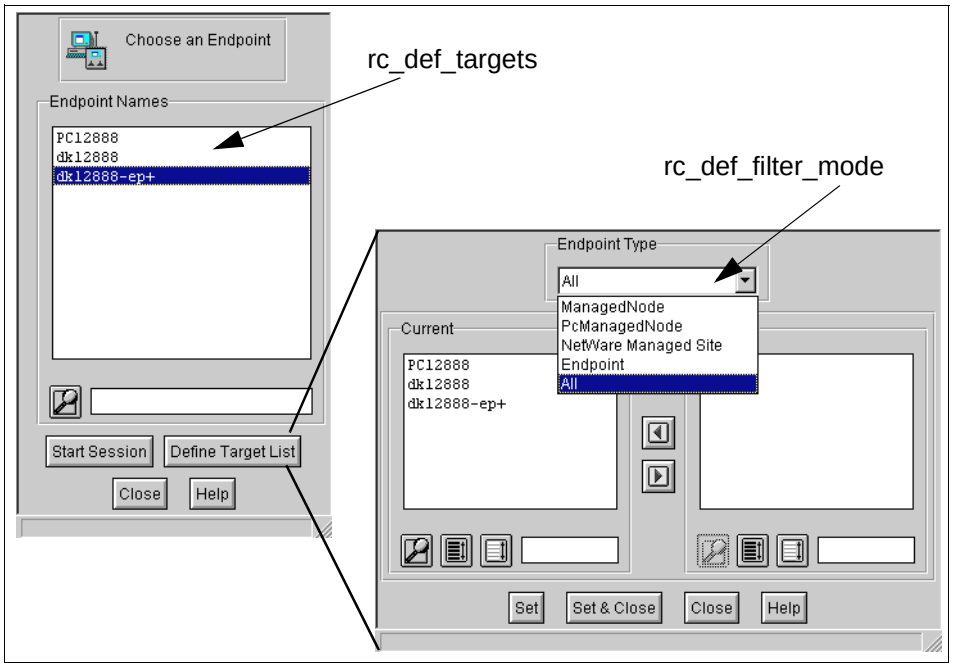


Figure 229. Remote Control DefinableTargetList Endpoint Selection Policies

Having selected the endpoint that is to be the target of the Remote Control session being setup, the session control panel is displayed. In this, the parameters controlling the session are specified. By means of default policy objects, default values can be provided for any field, and this can optionally be locked to prevent the user from modifying it.

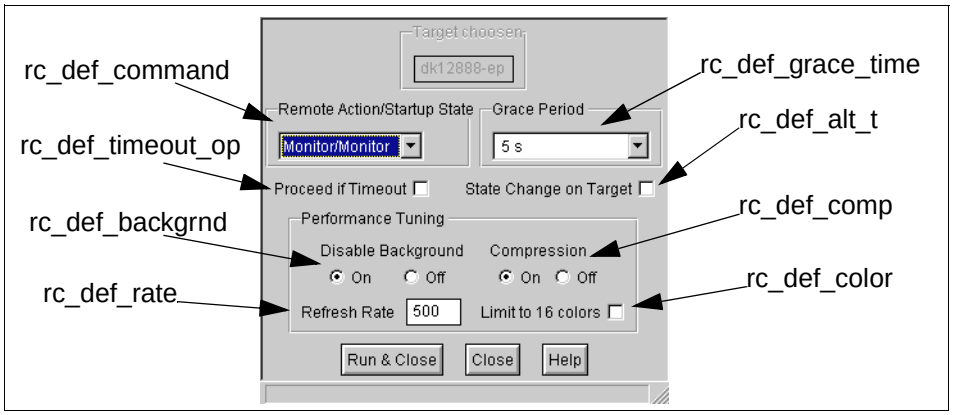


Figure 230. Default Policies for Controlling the Remote Control Session

To apply any changes to any default Remote Control policy, in any default policy object, use the `wgetpolm` and `wputpolm` commands:

1. Export the current policy to a file:

   ```
   wgetpolm RemoteControl <default_policy_object> <policy> > file
   ```

2. Modify the file using an editor.
3. Add the modified file to the custom default policy object:

   ```
   wputpolm RemoteControl <default_policy_object> <policy> < <file>
   ```

Refer to *Tivoli Framework Command Reference* for details on using the `wgetpolm` and `wputpolm` commands.

18.2.4 Defining Gateways for Remote Control

For each default policy object, the value of the `rc_def_gw` and `rc_def_ports` controls the use of gateways. Only one gateway can be assigned to each default policy region object. Note that the Remote Control gateway component is not in any way related to the Tivoli Framework gateway. Both the Tivoli gateway and the Remote Control gateway can be present at the same managed node, but there is no relationship between the two.

If the `rc_def_gw` is specified, the label of the managed node should be used as a Remote Control gateway. Also, specified here are the port used for incoming requests from controllers, the maximum number of simultaneous sessions allowed, the tcp/ip port number to use for communication with targets, and the IPX-port to be used to communicate with targets using the IPX protocol.

Note

Both the IP-port and the IPX-port have to be specified - even if one of the protocols is not used to communicate to targets. A value of `0` will let the system allocate the port number and should be applied when either of the protocols is not used.

Figure 231 shows an example of a `rc_deg_gw`:

```
#!/bin/sh
#
#  Default policy method for Remote Control Gateway
#
#  This method checks if you want to activate the Gateway or NOT.
#
#  Possible modes are:
#  YES ManagedNode-label GatewayPort MaxSesions IP:IP-Port XP:IPX-PORT
#       (i.e. YES MyMNode 3400 24 IP:2500 XP:0)
#       NO
#

# --- default -- echo "NO"

#
#
# use gateway 'itso2' port 9488 for TCP/IP
#
echo "YES itso2 9488 32 IP:8487 XP:0
#
#
exit 0
```

Figure 231. Using rc_def_gw Policy to Define a Remote Control Gateway

When the gateway is configured, the target usage of protocols has to be set up. This is obtained through the `rc_def_ports` policy. In this policy, the following parameters are controlled:

TIP Port number for target-to-gateway `tcp/ip` communication

CIP Port number for controller-gateway `tcp/ip` communication

TXP Port number for target-to-gateway `spx/ipx` communication

CXP Port number for controller-gateway `spx/ipx` communication

An example of this is shown in Figure 232 on page 586.

```
#!/bin/sh
#
#  Default policy method for Remote Control
#
#  This method allows to set the Target-Ip-Port/Target-ipX-Port
#  used from the tgt application to listen for the connection,
#  default value is 0 that means (TIP:2501, TXP:0x8771)
#  and the Controller-Ip-Port/Controller-ipX-Port used
#  from the Cntrl application to initiate the connection,
#  default value is 0 #  that means a port assigned by the system
#  will be used.
#
#  Possible modes are:
#           TIP:target-ip-port CIP:Controller-Ip-Port
#           TXP:target-ipx-port CXP:Controller-ipX-Port
#
#

# -- default -- echo "TIP:0 CIP:0 TXP:0 CXP:0"
#
#
echo "TIP:9433 CIP:9432 TXP:0 TXP:0 CXP:0"

exit 0
```

Figure 232. Using rc_def_ports Policy to Define Remote Control Gateway Ports

Like the `rc_def_gw` policy, the `rc_def_ports` requires all four parameters to be specified. Use a value of `0` for the protocols not used. This applies system defaults.

18.3 Taking Control of a Target

To take control over a target, the controller requests the session using the Tivoli infrastructure. On the managed node hosting the RemoteControl object (the Server in Figure 233), the `start_target` method is used to activate the remote control program on the target. If the target starts successfully, and the user did not reject the session, the managed node then activates the `start_controller` method on the method caller to let the controller executable start. From then on, the session between the controller and the target runs entirely on the TCP/IP protocol with no involvement from the Tivoli environment. This is shown in Figure 233 on page 587:

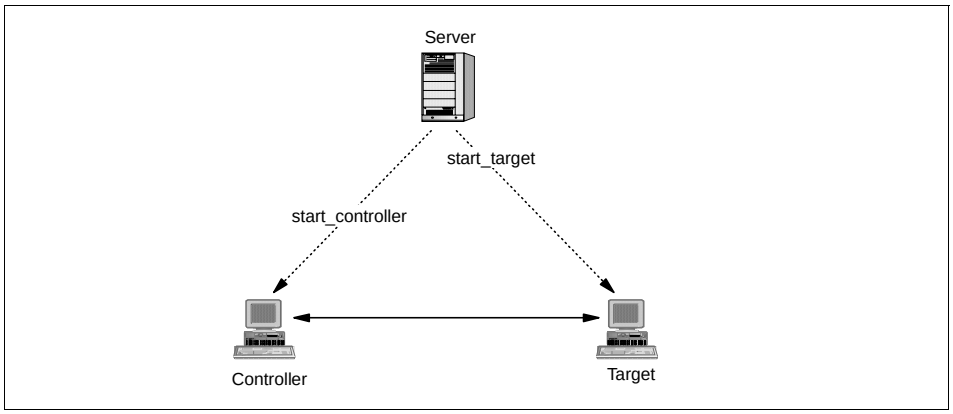


Figure 233. Remote Control Session Initialization without RC Gateway

When involving a gateway, having modified the rc_def_gw policy, the picture is slightly changed, as shown in Figure 234:

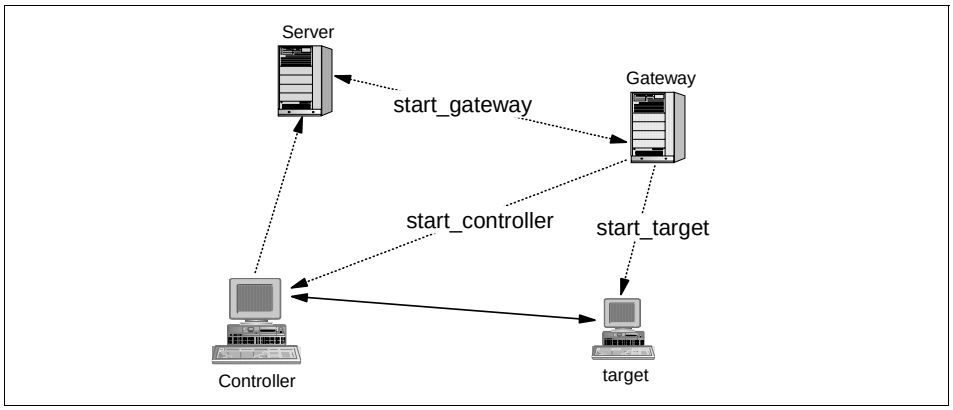


Figure 234. Remote Control Session Initialization with RC Gateway

Whenever the controller uses the RemoteControl object to initiate the session, the managed node owning the RemoteControl object will make sure that the Remote Control gateway is active. This managed node looks up information for the gateway and issues the start_gateway method. When this runs successfully, the process of selecting a target can continue as before.

Having selected a target for the Remote Control session, the controller method sends a request to the gateway (using the port specified on rc_def_gw and rc_def_ports policies), and the gateway issues the start_target to prepare the target end of the session. The start_controller

method is now used to activate the Controller end, and the session is established.

18.3.1 Remote Control Trace

To find out what is really happening behind the scenes of Remote Control, a number of facilities are available.

As for any other Tivoli applications, the `oserv` command gives a first indication of any problems. To further investigate problems, Tivoli Remote Control allows the generation of a unique trace file for each session. This option is available only for Windows NT, Windows 95/98, and OS/2 controllers and targets.

On Windows NT and Windows 95/98 platforms, tracing is controlled using a registry key. The registry paths are:

Target:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Target\trace length
```

Controller:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Controller\trace
length
```

Setting the value to `0` disables tracing. Any other value will enable tracing, dumping the first number of bytes - specified in *trace length* - from each communication buffer to the trace file. The trace file is named RCxxxxx.TRC, where xxxxx indicates the process ID of the current session.

Refer to the *Tivoli Remote User's Guide* for details on setting up the trace on OS/2 Controllers and Targets.

Documentation Correction

The *Tivoli Remote Control User's Guide 3.6* states that on Windows NT and Windows 95 the trace files (RCxxxxx.TRC) will be stored in the location specified by the value of the register-key:
`HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Controller\RCpath`
or
`HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Target\RCpath`

For Remote Control 3.6 on Windows NT, this is false. No matter what value is specified for the RCpath, the trace files are stored in the directory `%windir%\system32`.

Besides the trace files, Tivoli Remote Control offers the option of logging the start and stop of sessions. This option is available only for Windows NT and Windows 95/98, and like the tracing options, it is controlled by settings in the local registry of the individual machine. The paths are:

Target:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Target\logging
```

Controller:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Remote Control Controller\logging
```

18.4 Troubleshooting Remote Control

Troubleshooting the Remote Control environment involves the usual Framework troubleshooting procedures - looking in the output from `odstat` and/or `wtrace`. In addition, on Windows NT and Windows 95/98, information can be gathered from the eventlog and the trace files. Tracing is also an option on OS/2.

18.4.1 Framework Troubleshooting

From the TMR server, or the Remote Control gateway, the `odstat` output will contain information with respect to starting and stopping the various components. Look for the following methods:

- `start_gateway`
- `start_target`
- `start_controller`
- `close_gateway`

There are no references to the stop of controller-target sessions.

18.4.2 Windows Eventlog

If logging is enabled, the eventlog on controllers will show an event for all session starts and stops. The information is stored locally and includes the type of event (start/stop) and the tcp/ip address of the target, as shown in Figure 235 on page 590.

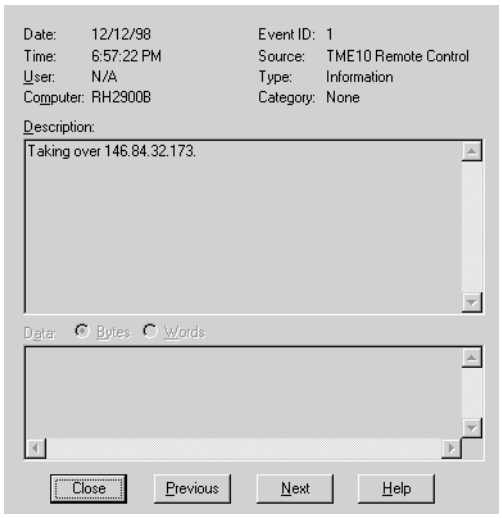


Figure 235. Remote Control Controller Event

18.4.3 Trace Files

As stated in “Remote Control Trace” on page 588, the trace files for Remote Control 3.6 on Windows NT are stored in `%windir%\system32` and not, as stated in the documentation, a location controlled by the RCPath registry key.

The information in the trace files is intended for internal usage, and is not documented. In order to interpret the information, a deep knowledge of the protocol used by Remote Control to send/receive data is needed.

For practical purposes, the trace file will be of use to determine when and how much data is passing between the controller and the target.

An example of a trace file from a Controller is shown in Figure 236 on page 591:

```
Open succeeded. --> rc = 0000000000

Send starting. --> rc = 0000000000

Send succeeded. --> rc = 0000000000
buffer lenght = 11
      dumping = 11
0B 00 00 41 67 6F 50 61 6F 6C 6F

Send starting. --> rc = 0000000000

Send succeeded. --> rc = 0000000000
buffer lenght = 26
      dumping = 26
1A 00 00 12 00 01 02 00 02 01 6F 02 00 45 51 4E
4B 42 45 4E 55 2E 44 41 54 00

First Receive. --> rc = 0000000002
buffer lenght = 2
      dumping = 2
0C 03

Receive succeeded. --> rc = 0000000000
buffer lenght = 780
      dumping = 200
0C 03 00 12 00 24 00 00 04 00 03 03 00 01 00 00
00 00 00 80 02 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF FF 00 00
80 00 00 00 80 80 00 80 00 00 00 80 00 80 00 80
80 00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00
FF 00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF
FF 00 00 FF FF FF 00 0A AA AA AA 3B 00 43 00 C8
05 0F 00 00 10 49 00 B8 00 14 00 B8 00 14 00 43
00 3A 00 5C 00 57 00 49 00 4E 00 4E 00 54 00 5C
00 73 00 79 00 73 00 74 00 65 00 6D 00 33 00 32
00 3B 00 43 00 3A 00 5C 00 57 00 49 00 4E 00 5C
00 3F 00 3F 00 5C 00 43 00 3A 00 5C 00 57 00 49
00 4E 00 4E 00 54 00 5C

First Receive. --> rc = 0000000002
buffer lenght = 2
      dumping = 2
10 00

Receive succeeded. --> rc = 0000000000
buffer lenght = 16
      dumping = 16
10 00 00 12 00 02 14 00 05 73 00 02 00 03 03 00
```

Figure 236. Extract from a Remote Control Trace File

Part 3. Additional Information

Appendix A. Tivoli's Use of Windows NT

This appendix comes from a White Paper produced in Tivoli's support organization. It contains a technical discussion of Tivoli Enterprise running on Microsoft's Workstation and Server release of NT Version 3.5.1 and 4.0.

This re-emphasizes many of the tips, techniques, and other information presented throughout the rest of this publication and adds more detail specific to the Windows NT platform.

A.1 Introduction

The information in this appendix is organized as follows:

- Tivoli Authentication Package (TAP)
- Tivoli Enterprise and NT User/System Accounts
- Security concerns
- Installation and uninstall of Tivoli Enterprise
- Environmental Issues (DLL conflict, WINS, Name Resolution)
- Tivoli Commands Specific to NT
- Microsoft and Third Party utilities
- Issues
- PcAgent
- A list of methods that utilize the `$root_user` idmap

A.1.1 Intended Audience

This is a technical document intended for those responsible for planning, implementing, and supporting Tivoli Enterprise in an environment with NT Workstations and Servers.

A.1.2 Scope

This appendix was written to cover Version 3.1.x, 3.2, and 3.6 of Tivoli Enterprise. This will address both the Tivoli Management Framework (TMF) and the Tivoli Management Agent (TMA). Although the PC agent does not utilize TAP, section A.10, "PC Agent Overview" on page 637 will address its implementation on NT.

This appendix focuses only on the basic services provided by TMF/TMA. As these basic services are responsible for the spawning of processes and

enforcing of security, this will provide a good foundation in how Tivoli applications interact with Microsoft Windows NT. Refer to the application chapters in this redbook for more information specific to those applications working with Windows NT.

A.1.3 Conventions

This document will address both the classic Tivoli Management Framework as well as the new Tivoli Management Agent. The following abbreviations will have the following meanings:

TMF Tivoli Management Framework (based on the oserv service)

TMA Tivoli Management Agent (based on the lcfd service)

Terminology Note

Prior to the General Availability release of v3.6, the TMA endpoint was referred to as LCFD. If one issues the command `net start`, you will see Tivoli Lightweight Client (which is the service name). However, using other commands or viewing certain registry keys will show `lcfd` (which is the process name). Because this is a technical document, and the purpose is to provide a clear picture of the implementation, `lcfd` will be used to describe the process running on the NT. Otherwise, TMA will be used to describe the general implementation of Tivoli's Endpoint technology.

This document is available in PDF form at `http://www.support.tivoli.com`.

A.1.4 Other Resources

For a good review of NT's security model, refer to Mark Russinovich's articles in the May and June Windows NT Magazine (`http://www.winntmag.com`). See also Microsoft documents Q96005, Q102716, Q122422 from TechNet as well as Chapter 6 of the NT Resource Kit 4.0 for additional information on security and Microsoft NT.

A.1.5 Acknowledgments

This document could not have been written without the help of Conrad Johnson, Tivoli development. Other valuable contributions from Viki Stevens, Rob Tulloh, Sean Allen, Ian Willoughby, Simon Allen, Mark Adams, Bowman Hall, Gary Hamilton, Shane Frensley, Brian Graham, Sean Larkin, Angelo DeRise, Jerry Saulman (Tivoli), and the many people that contributed on the tme10 listserv, including Jamie Carl, Pete Meechan, Gene Martin, Paul Claridge and John Chapin.

A.2 Tivoli Authentication Package

One of the fundamental difference between the UNIX and NT implementation of Tivoli Enterprise is the Tivoli Authentication Package (TAP). This section will explore the purpose and implementation of TAP.

A.2.1 Why TAP Is Needed

A requirement of the Tivoli Object Request Broker (OSERV) and Tivoli Management Agent (LCFD) is that it be able to run methods in the context of a given user associated with the method. That is, the resources accessible to the method are those accessible to the given user. Such methods are known as setuid methods. The Tivoli Authentication Package (TAP) is installed and loaded by the Local Security Authentication (LSA) subsystem of NT allowing setuid methods to work on NT.

A.2.2 Understanding TAP

NT supports impersonation, which is the ability to spawn a process as a user other than the parent process. Like the implementation of NT's file services, Tivoli utilizes these services to implement TMF and TMA. Tivoli Authentication Package is the cornerstone of this impersonation.

Impersonation could be enabled using two WIN32 API calls available: `LsaLoginUser()` and `CreateProcessAsUser()`. The `LsaLoginUser()` function creates a logon token for a given user obtained by accessing the Microsoft Authentication Package (msv1_0). Two parameters required are the user and the clear text logon password of the given user.

This would be an issue, as the Tivoli oserv/lcfd does not store passwords with methods, and any password changes for users would need to be updated to allow the oserv/lcfd to impersonate. Furthermore, transmitting passwords securely would also be a concern.

Microsoft also provides impersonation by allowing users to install their own authentication package that can create logon tokens that can then be passed to the `CreateProcessAsUser()` function. This is the method that Tivoli uses. The oserv/lcfd calls the Tivoli Authentication Package to obtain a logon token. Because the Tivoli Authentication Package trusts that the oserv/lcfd has already authenticated the calling method (a basic feature of Tivoli), it will pass a logon token that is suitable to the `CreateProcessAsUser()` function. This eliminates the need for passwords, which can be changed and not impact the Tivoli processes.

`%SYSTEMROOT%\system32\TivoliAP.dll` is TAP's implementation that provides this authentication. It is registered with NT's LSA (Local Security Authentication) under the registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\AuthenticationPackages
```

When LSASS.EXE (LSA Service) starts, it will load TivoliAP.dll. This is the reason the reboot is necessary after an installation.

A.2.3 How TAP Works

The oserv/lcfd service runs as NT AUTHORITY\SYSTEM. This is a special group account that has privileges for the local machine and authenticates to the local Security Accounts Manager (SAM). It is not possible to change the user that the oserv/lcfd runs as. Below is an example of how Tivoli interacts with NT to initiate a setuid method:

1. A request is made from a (Tivoli) client to initiate a method.
2. The method's user is identified.
3. The oserv/lcfd service passes the user name and domain (if any), and that it wants to use the Tivoli Authentication Package to the Local Security Authority (LSA)
4. The LSA forwards this information to TAP.
5. If there was no domain name initially passed to LSA from the oserv/lcfd, TAP calls first the local SAM database to retrieve the account SID and global group SIDs. If the account is not found, it proceeds to the domain SAM.
6. If the user passed was qualified as a domain account (DOMAIN\user), TAP will proceed directly to the domain controller.
7. The account SID and global group SIDs are returned to TAP. At this point, there is still no token generated.
8. Before passing the SID back to LSA, if the account is not tmersrvd or the local Administrator, TAP makes sure the account is a normal account that is not disabled, expired, or locked out. The account's password must not be expired. If these checks fail, TAP tells LSA not to create a logon token for the account.
9. TAP passes the collection of SIDs to LSA. LSA checks the local LSA policy database for any user rights associated with the user.
10. LSA places all this collected information into a structure that is called the token or access token. LSA passes this token back to the oserv/lcfd process.

11. The oserv/lcfd process calls CreateProcessAsUser() and passes the token along with the executable that it will start.

This method of impersonation allows processes access to the local resource of that NT running the oserv/lcfd service. Although the token structure may contain domain SIDs, it is suitable only to access local resources as the user rights incorporated in the token structure were from the local LSA policy database. Tokens generated from TAP on a Primary/Backup Domain Controller will also be limited to local resources on that domain controller.

A.2.4 Understanding the Tivoli Remote Access Account

The purpose of the Tivoli Remote Access Account (TRAA) is to control access to remote NT resources, like Netshares, through the oserv/lcfd. Although the installation and the documentation refer to the TRAA account being used to remotely share binaries, the TRAA account has much farther-reaching consequences.

When a program runs in NT, a logon token is associated with that program. Logon tokens are created when a user logs in, or are created when the `LogonUser()` function is called, or are created by authentication packages, such as TAP. Logon tokens contain a user account name used to access local resources and zero or more credentials used to access remote resources.

When an NT program accesses a local resource, it uses the user account referenced in the program's logon token to determine if it can access the local resource. When an NT program accesses a remote resource, such as a Netshare, the LAN Manager authenticates that the logon token associated with the requesting program has the correct set of credentials that allows it to access that remote resource. A credential contains an account name and that account's encrypted password. A user logging in, or a call to the `LsaLoginUser()` function, creates a logon token that initially has one credential. This one credential contains the user and password for the account used to created the logon token. Additional credentials may be added to a logon token. When you specify an account and password to connect to a remote disk drive, a credential containing that account name and password are added to your logon token.

Logon tokens created by TAP do not have any credentials that contain the user and password for the account used to create the logon token because the password is not used to create the token. Instead, if a TRAA has been specified using the `-r` option of `wsettap.exe` (TMF) or `wlcftap.exe` (TMA), TAP creates a logon token and adds to that token the credentials of the TRAA. Therefore, a program with a logon token created by TAP accesses local

resources accessible to the user specified to TAP when the token was created (the setuid user) and will access remote resources using the TRAA (if any).

The `-r` option of the `wsettap.exe`/`wlcftap.exe` command is used to set and change the TRAA. The account information is stored in the registry key:

```
HKEY_LOCAL_MACHINE\Security\Policy\SecretsTivoli Remote Access Account Credential\
```

By default, `HKEY_LOCAL_MACHINE\Security` is unreadable. The TRAA does not have any impact on the local ManagedNode/Endpoint and does not need to be set in order for the standard Tivoli Enterprise functions.

If design requirements of the Tivoli Management Region require that some application will need access to a remote resource, the TRAA account will only need the necessary access to these remote resources. An example of using the TRAA account would be a FilePackage's after-script that copies the install log to a remote share. The TRAA account could be set to DOMAIN\tmeuser, and the share would allow write access to this share by the DOMAIN\tmeuser account. Another example would be Tasks created to start a particular service remotely using the `sc.exe` or `netsvc` command (part of the NT Resource Kit) if a TEC event arrived saying that the service was down. One design of the task could be to run the task locally on the machine as a local or domain NT administrator. Another solution would be to set the TRAA account on one of the TMF/TMA nodes to a domain level administrator and execute the task on this node.

While TAP has no dependencies on password changes, the TMF/TMA nodes will be impacted if the TRAA account's password is changed. Because each NT ManagedNode/Endpoint's TRAA account is stored locally and keeps the password, a change in the account would require that the `wsettap.exe`/`wlcftap.exe` be run on each node to update the password for TRAA. If the account or password has changed in the domain, yet the TRAA account has not been updated, oserv/lcfd will not restart.

If the TRAA account is to be used in the environment, it is best that the account and password remain static.

A.2.5 Order of Account Selection

The Tivoli Authentication Package will first attempt to resolve an unqualified user account using the local SAM database, and then if unsuccessful, resolve the name in the domain. An example of an unqualified account would be where a user is specified as `fred`, instead of `BEDROCK\fred`. In this case, when the task's batch, script, or executable was invoked on the NT endpoint, TAP

first tried to spawn the process as MACHINE\fred. If this account did not exist, it then would attempt to resolve it in the domain. Especially, in complex domains with various trusts, it is best to fully qualify the user ID.

A.2.6 wsettap.exe and wlcftap.exe

The commands `wsettap`/`wlcftap` manage the TMF/TMA node in 2 ways:

1. Enables and disables the Tivoli Authentication Package from the LSA list of Authentication packages.
2. Enables and modifies the TRAA account.

It is often said that the TAP account has been modified using `wsettap`/`wlcftap`. This is not technically correct. When the `wsettap –r`/`wlcftap –r` is issued, the TRAA account is being modified. When `wsettap –a|d`/`wlcftap –a|d` is issued, TAP is being enabled (or disabled).

A.3 Tivoli Accounts

This section will discuss the accounts created by the Tivoli installation process, the differences that Tivoli Enterprise 3.6 introduces compared to version 3.2 ‘SuperPatch’ and earlier, and how certain enterprise policies affect TMF/TMA on NT.

A.3.1 Accounts Created

Tivoli introduces two accounts at installation time. The accounts are the user tmersrvd and the group Tivoli_Admin_Privileges. These accounts are created locally on the NT target’s Security Account Manager (SAM) database and are configured the same for TMF/TMA. User Right changes can be viewed in the UserManager under the Policies menu option. User rights can also be granularly changed using the `ntrights.exe` application in the resource kit.

A.3.1.1 tmersrvd Account

The tmersrvd account is an unprivileged account. A password is randomly generated at installation, and the account can be disabled without affecting the framework. Many of the Tivoli methods will run in the context of tmersrvd. The password can also be changed with no adverse effect on Tivoli.

The User Rights required for this account are:

- Bypass Traverse Checking (SeChangeNotifyPrivilege)

  The tmersrvd account does not get assigned to this User Right directly. A new install of NT will assign the special group Everyone to Bypass

Traverse Checking. (In non-US versions of NT, this group will be referred to as that local language equivalent.) This allows a user to traverse a directory tree even if the user has no other rights to access that directory. If security policies in the corporate enterprise disallow Bypass Traverse Checking, the tmersrvd account needs to be added directly to this user right.

- Log on Locally (`SeInteractiveLogonRight`)

  This is assigned to the tmersrvd account during installation of the framework.

A.3.1.2 Tivoli_Admin_Privileges

This group is by default assigned to the built-in administrator. Unless the TMR server is an NT, in which case, the account used to install the TMR server will be assigned to this group.

It has three required advanced User Rights:

- Act as Part of the Operating System (SE_TCB_NAME, SeTcbPrivilege).

  The user can act as a trusted part of the operating system.

- Increase Quotas (SE_INCREASE_QUOTA_NAME, SeIncreaseQuotaPrivilege).

  The user can increase object quotas. Each object has a quota assigned to it.

- Replace a process level token (SE_ASSIGNPRIMARYTOKEN_NAME, SeAssignPrimaryTokenPrivilege).
- The user can modify a process' access token.

The Act as Part of the Operating System privilege is required when running the `wsettap`/`wlcftap` command with no options specified because it communicates with the LSA to retrieve the current configuration of TAP. Other operations of the `wsettap`/`wlcftap` command communicate with the registry and not with the LSA; so, they do not require a special privilege. (Any invocation of the `wsettap`/`wlcftap` command must be from a member of the Administrators group).

The privileges Increase Quota and Replace a Process Level Token assigned to the Tivoli_Admin_Privileges group are the privileges required to start a process as a different user. The run_task and sentry_engine methods, for example, require these privileges. These methods run as the built-in administrator (Version 3.2 or earlier) or $root_user (Version 3.6 and later).

Note

If you change the value of the $root_user idmap for the Windows NT interpreter type (w32-ix86), you must ensure that the account is a member of the Tivoli_Admin_Privileges group. If the account is not part of the Tivoli_Admin_Privileges group, TMF/TMA nodes will receive tap_call_init failed, error 38.

A.3.2 Accounts Used by Tivoli Enterprise

There are seven types of accounts that Tivoli Enterprise will use in the environment: Initial installation of TMF/TMA, system, privileged, unprivileged, idmap, TRAA, and any user-defined accounts.

A.3.2.1 Installation Account

The installation account is used at the time that the TMF/TMA files are being installed. For TMF, this is when the initial framework is being installed through the classic install or SIS. For TMA, this would be when the Endpoint software is being installed through SIS or `winstlcf`.

This installation account can be a local account defined in the NT's SAM database or a domain account that is resolvable by this node. This account has two requirements:

- It must be in the Administrators group.
- It must have the user right Logon Local.

Once the framework (TMF) or Lightweight Client (TMA) has been installed, this installation account is not used. Therefore, the account can be revoked or disabled.

Note

The account used for installation of an NT TMR Server does have a special role. When installation completes, the $root_user idmap will map that Installation account to the w32-ix86 interp. Therefore, prior to removing this account, set the $root_user idmap to another Administrator and map the new login to the Tivoli Root Administrator (see also the command `wauthadmin`). Also, be sure to include this new Administrator in the Tivoli_Admin_Privileges group.

For example, a company creates a domain administrator MASTER\tivinstall. The MASTER domain is the top level domain, and there are several resource domains that have a two-way trust relationship with MASTER. When

TMF/TMA nodes are being installed, an NT Administrator in the MASTER domain will enable the account and provide a password to the Tivoli Administrator responsible for installation. This Tivoli Administrator logs into the TMR as their normal account and specifies this user and password while installing the nodes. After completing the installations, the NT Administrator will disable MASTER\tivinstall until additional installations are required.

A.3.2.2 System Account

Tivoli Enterprise runs the `oserv(TMF)/lcfd(TMA)` service as `NT AUTHORITY\SYSTEM`. In addition, TMF installations will also have the `spider.exe` (HTML server) and possibly the `gateway.exe` (used to communicate to TMA endpoints) processes also running as `NT AUTHORITY\SYSTEM`.

A.3.2.3 Privileged Account

Tivoli Enterprise uses the privileged account when a management function of the Tivoli Enterprise needs access to privileged resources on the NT. Tivoli Enterprise will request that a certain program be started as this privileged account to access this resource.

See section A.3.5, “Privileged Account Tivoli Version Comparison” on page 610 for information on what account is used based on the version of Tivoli Enterprise.

A.3.2.4 Unprivileged Account

The unprivileged account is the `tmersrvd` account. Tivoli Enterprise will attempt to run as many executables as this user to minimize any possible security compromises. If the `tmersrvd` account is not found on the local NT’s SAM database, it will look to the domain for this account. This `tmersrvd` account must exist in order for Tivoli to function properly.

As noted earlier, this account has no real access and can be disabled if desired. Unlike other user accounts, the `tmersrvd` account can be disabled (but not deleted) with no adverse effect on Tivoli.

The `tmersrvd` account should not be deleted from the node unless the account exists in the domain. However, due to the load that would be placed on the Primary Domain Controller (see section A.3.6, “Domain Controllers” on page 612), this is not a good alternative.

A.3.2.5 ID Map Account

Because Tivoli Enterprise spans a heterogeneous environment, the idmap was introduced to provide a means of mapping a special ID (referred to as an idmap) to an OS-specific user account. On NT, the idmap may contain a

reference to w32-ix86, which is the definition within Tivoli to describe an NT node.

The idmap `$root_user` is a pre-configured idmap that resolves to Administrator on NT. This map is used for various processes on NT. (See section A.3.5, “Privileged Account Tivoli Version Comparison” on page 610 and 7.1.5, “ID Mapping” on page 199 for more information on this map). When the oserv/lcfd service is asked to resolve the method that is to run as `$root_user`, it will look for the string Administrator.

Note

The idmap `$root_group` or another group ID must exist for an administrator beginning with NT 4.0. Under NT 3.51, this requirement had no impact if the group ID was blank. Installations that upgrade from 3.51 to 4.0, which previously had no group identifiers for their administrators, will suddenly find themselves unable to log in to Tivoli except as the root administrator who, by default, has the group ID `$root_group`.

Idmaps can be modified to reflect a naming convention with an enterprise using the `widmap` command.

The `root_group` idmap for NT is not actually used when a process starts. However, it is important that the `root_group` idmap has a group listed for NT although it does not need to be a privileged group.

A.3.2.6 TRAA Account

As described in Section A.2.4, “Understanding the Tivoli Remote Access Account” on page 599, the TRAA account is used when a Tivoli process must access a remote resource. By default, no Tivoli process requires that the TRAA account be defined.

When defining the TRAA account, it is important to identify the reason for this account (such as to enable a task to run a domain command or having a software distribution package after-script write a file to a remote share). With these needs identified, create this domain account with the necessary rights. Again, the TRAA account needs only access to the resource defined. If a password change has occurred without updating the node(s) where the TRAA is set, oserv/lcfd nodes will fail to restart.

An example would be a company that creates the domain account `MASTER\tivuser`. `MASTER\tivuser` is granted write access to a Netshare called `\\SERVER\tivfiles`. This file allows only the user `MASTER\tivuser` write access, and the Administrators group would have full access.

The MASTER\tivuser password is set to not expire, and only the Tivoli Administrator responsible for installation knows the password. Because MASTER\tivuser only has this limited set of rights, there is a low risk of this account being compromised and attacking other resources.

A.3.2.7 User Defined Accounts

Tivoli Enterprise can be configured to use other user accounts as design needs warrant. There are several areas that allow a Tivoli administrator to define a certain action to run as a certain user. A task, for example, is created to manage MSSQL, and the task is defined to run as a given MSSQL Administrator.

Please refer to section A.3.6, “Domain Controllers” on page 612 for information on the use of domain accounts.

A.3.3 Identifying Under Which User a Given Process Will Run

There will be times when it is necessary to identify what a process would be running so as to identify possible permission or troubleshooting a failed action. Below are several methods to identify the user a given method will run as.

A.3.3.1 TMR, ManagedNode, or Gateway Using odstat and wtrace

This outlines how one identifies the user of a process that runs on a TMF node. For this example, the Tivoli action was to do the following:

```
root#wrunquery -h freedom -f c:/tmp/test  GetNode
```

Identify a method in an odstat and its associated method:

```
334 M hdoq  1-984  done     6     0 15:14:24
1721656771.4.7#TMF_ManagedNode::Managed_Node# write_to_file
```

Issue the command resolve OID method:

```
root#resolve  1721656771.4.7 write_to_file
1721656771.1.345
```

At this point, you have identified the behavior object where the method is defined. Issue the command `objcall BEHAVIOR OID om_stat methodname`:

```
root#objcall $TMR.1.345 om_stat write_to_file
CATALOG=
SET_USER=*
SET_GROUP=*
EXPORT=TRUE
EXECUTE=FALSE
default
```

At this point, you have identified who the method will run using the SET_USER flag. In this case, it is '*'. Section A.3.4, "Options for the SET_USER" on page 609 describes the flags for SET_USER.

Issue the command `objcall BEHAVIOR OID om_get_definition methodname default`:

```
root#objcall $TMR.1.345 om_get_definition write_to_file default
STORAGE=/TAS/MANAGED_NODE/man_node_skel1
MODEL=queued-obj-daemon
```

At this point, the process `man_node_skel1` has been identified as the implementation of the `write_to_file` method.

Because the SET_USER has a '*', we must identify who the user will be. To do this, we identify the Tivoli Administrator that would start this process and reference one of their logins:

```
root#objcall $TMR.0.0 get_principal_id root@opus.support.tivoli.com
$root_user
$root_group
```

At this point, we know that the method `write_to_file` will run as the `root_user` idmap. The file was written to an NT machine, so we must resolve the `root_user` idmap:

```
root#widmap resolve_entry root_user w32-ix86
Administrator
```

The process `man_node_skel1` will run as the user Administrator.

A.3.3.2 TMA Endpoint

Identifying what user a process is to run as in an NT TMA node is not the same as with TMF nodes. With the advent of caching methods on gateways, it will be difficult to identify methods acting on the endpoints using `odstat`. Below is an alternative means of identifying the method that is running against the endpoint and then identifying the user and process.

This example will look at the distribution of a FilePackage to a TMA node.

Set the debug level to 6 of the gateway:

```
wgateway gatewayname set_debug_level 6
```

Restart the gateway:

```
wgateway gatewayname restart
```

Once the gateway is up, execute the action on an endpoint that is assigned to that gateway:

```
wdistfp –a –d @FilePackage:StdConfig @Endpoint:binkley
```

Once completed, locate the `%DBDIR%\gatelog` (`$DBDIR/gatelog` on UNIX) on the gateway. In the gatelog file, locate the methods that the endpoint was verifying that it had already cached or needed or where an MDist flag is referenced, such as below:

```
1998/11/24 11:53:40 +06: mdist: distribution ID = 27,method =
fps_install,size = 0
```

Note, on a busy gateway, this will be next to impossible as there could be multiple actions occurring. It is recommended that this be done on a test TMR. Once the method has been identified, locate the method in the Tivoli Object Repository:

```
odbls -M fps_install -k $DBDIR
1721656771.1.330
method:
   fps_install
```

Note: There is a possibility that a method may be overloaded, in which case, `odbls` will return several possible instances. This is beyond the scope of this document, but if there are duplicate methods listed from the above command, the best thing to do is check all instances, and, hopefully, they return the same value.

Now that the behavior object has been identified, execute the command:

```
objcall Behavior Object om_stat method
objcall $TMR.1.330 om_stat fps_install
SET_USER=$root_user
```

A.3.3.3 Identifying the User Using ADE *.ist Files

Another method of identifying a user is to install the ADE files that are part of the Framework CD and match the method (from an `odstat`/`wtrace`) to the appropriate *.ist file. This is useful for the basic framework methods. These *.ist files do not exist for the applications at this time.

As an example, a Tivoli Administrator creates a Policy Region. The `wtrace` reveals this information:

```
Object ID:   1721656771.1.195#SharedPolicyRegions::Engine#
Method:      create_policy_region
Principal:   CRITSIT-LAB\mhahn (36458574/0)
```

Search the ist files for the method `create_policy_region`:

```
>grep –i create_policy_region *.ist
PolicyGUI.ist
```

Identify the user that the method will run as:

```
                    TMF_imp_PolicyRegion::GUI::create_policy_region
          } = {"default", "/TAS/PRDO/Policy_GUI";};
  };
```

The options are noted in section A.3.4, “Options for the SET_USER” on page 609.

A.3.4 Options for the SET_USER

These are as follows:

- Privileged: `SET_USER=root`

  On NT, this will map to the built-in Administrator account (See section 3.5 for more information). It is important to stress that this built-in Administrator account can be renamed and not affect any privileged methods.

- Unprivileged: `SET_USER=`

  (If viewing *.ist files from ADE, this is referred to as default.)

  On NT, this will map to the tmersrvd account.

- Idmap: `SET_USER=$value`

  The `$` has a special meaning to the oserv/lcfd. This will refer to an idmap. To view the idmap, use the command widmap. An example of this is:

  ```
  SET_USER=$root_user
  #widmap resolve_entry root_user w32-ix86
  Administrator
  ```

  Idmaps are managed on a TMR level and are designed to provide a means of mapping certain accounts based on the OS.

  When an idmap references an account name, like Administrator, TAP must locate that name first in the local SAM database or in the domain SAM. If an account Administrator is not found, the process will not start. This behavior is different with `SET_USER=root` as root is mapped to a SID directly, which will map to whatever the built-in Administrator account has been renamed to.

- User-defined: `SET_USER=*` or Tivoli applications that support setting a UID, such as tasks and SentryProfiles

As Tivoli Enterprise is deployed, there are many areas in the products that allow customization. Tasks and SentryProfiles are two of the areas that allow Tivoli Administrators to define what user these customized programs will run as.

A.3.5 Privileged Account Tivoli Version Comparison

This section is extremely important as there have been changes in the use of the privileged accounts between Version 3.2 and earlier Version 3.6.

A.3.5.1 TMF/TMA when SET_USER=root

Prior to Version 3.6, the built-in administrator account was used for most processes needing to run in the context of a privileged user. So `SET_USER=root` maps to the local, built-in, NT Administrator. This is a special account reserved by NT and has full rights to the system. This account is defined as SID 500. Tivoli calls this special SID rather than the name Administrator.

This account can be renamed, and oserv/lcfd will run as the renamed account since the SID is the same. This account can't be demoted in privilege, therefore, all Tivoli privileged processes will have access to the local resources.

A.3.5.2 TMF/TMA when SET_USER=$root_user

Version 3.6 will run most privileged methods as the user name obtained using the `$root_user`. This provides the Tivoli Administrator with the ability to define the user for all Tivoli privileged processes. Several points need to be understood, however, to provide a smooth roll-out of the Tivoli environment:

`SET_USER=$root_user` does not use SID500, so account name in `$root_user` idmap must map correctly to a user account in local or domain SAM. Failure to map an account name will cause tap_get_sid_logon_token failed error.

The account must be part of the NT Administrators and Tivoli_Admin_Privileges group, and it must have the UserRight LogonLocally.

If the MACHINE\Administrator account is not renamed, there are no modifications to the `$root_user` idmap necessary unless desired to run the Tivoli Enterprise privileged programs as another local or domain account.

If the MACHINE\Administrator account is renamed, or the design of the TMR dictates using a domain account for privileged accounts, then:

- MACHINE\Administrator renaming must be consistent on all TMF/TMA nodes, or a local Administrator account is created on all TMF/TMA nodes. The account would not have to be called Administrator. Rather, it could be

called anything and would need to be consistent on all TMF/TMA endpoints and the root_user idmap would be updated to reflect the new account name.

- If a domain account is used, and there is a failure in communicating to the primary domain controller, TMF/TMA could adversely be effected as the account's SID could not be obtained to place in the token structure. This would cause the oserv/lcfd to not be able to spawn the requested process.
- The current design of TAP will query the Primary Domain Controller to authenticate domain accounts. It will bypass all local Backup Domain Controllers. If using domain accounts for root_user, the PDC will be queried for every invocation of a privileged Tivoli program.

A.11, "Version 3.6 Methods Using $root_user idmap" on page 638 outlines some of the common methods and their respective executable that will run using this idmap.

A.3.5.3 Example of v3.6 Using a Local SAM Account

Spinal Tap Incorporated has several NT domains. Each of these NT domains are located in given geographical locations and are linked to the corporate office through a 128 KB FrameRelay. In the corporate office, the master domain has two-way trusts with each of the geography domains. Each geography manages their domain separately, and there is no way to enforce consistent naming conventions in all the domains. As well, the company does not have any consistent naming convention for the built-in Administrator account other than that it is a mandate to rename the account.

One solution is to use a domain account in the master domain for the $root_user idmap. However, because of the slow links, using a domain account could impact the performance and reliability of Tivoli Enterprise. Instead, the company creates a new local account on each NT as NT is installed and configured named Nigel. This account is then added to the Administrators and Tivoli_Admin_Privileges group. The $root_user idmap is then changed to reflect this name for w32-ix86.

In this case, because the account is local, all authentication for the privileged account will occur in the local NT.

A.3.5.4 Example of v3.6 Using a Domain SAM Account

Acme Sprockets' NT domain design consists of a Multiple Master Domain with two-way trusts between the various master and resource domains. Each domain is part of the campus Network and is centrally managed. The corporate policy is to rename the built in Administrator account on each NT workstation and server along with the domain SAMs.

Acme creates a domain account on the master domain called TivPriv. The idmap is then set to the new user:

```
widmap rm_entry root_user w32-ix86  (Remove the Administrator reference)
widmap add_entry root_user w32-ix86 MASTER\\TivPriv
```

(Note the double \ . This is to escape the \)

In this case, for every process running as privileged, it will authenticate to the MASTER domain SAM and get the SID value for TivPriv.

A.3.6 Domain Controllers

This section will discuss issues particular to Primary and Backup domain controllers.

A.3.6.1 Authentication to Primary Domain Controller

As pointed out in section 3.5.2, TAP's current design will request domain user authentication from the Primary Domain Controller and bypass any local Backup Domain Controllers. This potentially can flood the PDC with authentication requests if a domain account is used for the `$root_user` idmap or applications, such as Distributed Monitoring, that can execute large number of processes in a short span of time.

Many NT environments use several NT domains to manage the environment. One common design is the use of a Master domain and then resource domains that are two-way trusted with the Master domain. If design requirements demand the use of a domain account for Tivoli Enterprise and the NT domains are configured similarly to the model described above, one could create an account in each of the domains with the same name:

- MASTER\TivAdmin
- US\TivAdmin
- EUROPE\TivAdmin
- JAPAN\TivAdmin

The `$root_user` idmap would map w32-ix86 to TivAdmin. When a TMF/TMA node runs a privileged process, TAP will see that the idmap references TivAdmin. It will first look to the local SAM database. Not finding the account there, TAP will then request from the node's primary domain controller. So, a node in the JAPAN domain will only authenticate to the JAPAN Primary Domain Controller rather than the MASTER domain controller.

Using the same model, assume a given task must run as MASTER\TivAdmin since this is part of the MSSQL login list. One would need to define

specifically MASTER\TivAdmin in the UID field of the task or create a new idmap that resolves to this account. Had MASTER not been part of the specified account, the same node in the JAPAN domain would get the SID for the JAPAN\TivAdmin rather than the MASTER\TivAdmin.

A.3.6.2 Account Creation on PDC/BDC

Another issue pertaining to primary and backup domain controllers will be the creation of the Tivoli accounts at installation. It is recommended to install TMF/TMA on the Primary Domain Controller first, then synchronize the backup domain controllers to allow the newly created accounts to propagate. If an installation is attempted first on a Backup Domain Controller, the installation will fail because the accounts have not been updated on the Primary Domain Controller. Either wait 15 minutes for the domain servers to re-synchronize and attempt the installation again or force the synchronization.

The Primary and Backup domain controllers are a special case regarding account management. Their accounts are actually considered a domain account. When Tivoli runs on either a PDC or BDC, the authentication will still take place on the local SAM database with no impact to the network or other domain controllers. Also, Tivoli will not force partial or full synchronization within the domain.

A.4 Security

This section will address concerns regarding how Tivoli is configured for security on TMF/TMA nodes.

A.4.1 Changes to NT Accounts Used by Tivoli Enterprise

Because of the TMF/TMA design based on the `CreateProcessAsUser()` system call, any process spawned to access local resources will not be affected by passwords. The exception is the TRAA account. Changing the password will require resetting of the password for any TMF/TMA's TRAA affected by the change. One method of automating this change is to create a task that will execute as the Administrator and issue the `wsettap/wlcftap` command with the new account.

With the exception of the tmersrvd account, accounts that have been disabled, expired, or locked out will fail when the oserv attempts to start a process as this user. This is due to TAP checking the status of this account prior to passing SID information back to LSA.

A.4.2 File System Issues

TMF requires that it be installed on an NTFS file system. During the installation, Tivoli will check that the target drive is an NTFS drive. TRIP does not require NTFS. It is installed on the C drive by way of the remote installation with the designated CurrentNtRepeat

TMA does not require installation on an NTFS file system. However, due to the insecure nature of the FAT file system, it is advisable to install the TMA files on an NTFS file system.

A.4.3 Permissions on Installation Directories

This section discusses the permissions required on the Tivoli directories.

A.4.3.1 Base Tivoli/LCF Directory

The base directory for Tivoli Enterprise's TMF/TMA install will be installed with the following directory permissions:

Administrators	Full Control
System	Full Control
Creator Owner	Full Control
Everyone	Change
Server Operators	Change

A.4.3.2 Tivoli DB Directory (TMF Only)

Prior the TMP 3.1.3, the group Everyone was denied access to the DB Directory. This caused issues when processes running as non-admin users attempted to access files in `%DBDIR%\tmp`. This is also an issue with TMP 3.2, and is addressed with the Framework SuperPatch. Version 3.6 is not effected.

A.4.3.3 %SYSTEMROOT%\system32

The tmersrvd account will require read access to `%SYSTEMROOT%\system32`. If this directory is restricted to only Administrators, and the Everyone group is not part of the access control list for `%SYSTEMROOT%\system32`, TMF/TMA will fail.

To correct this, add Bypass Traverse Checking to the tmersrvd account. Bypass Traverse Checking is referred to differently on non-US versions of NT. For example, the French equivalent is Outrepasser le contrôle de parcours.

A.4.3.4 Registry Access

There is no Tivoli process by default that manipulates the NT Registry. The commands `wsettap` and `wlcftap` will add or remove a value in LSA's AuthenticationPackage key (see section 2.2). The commands `wsettap`, `wlcftap`, `wmailhost`, `wlocalhost`, and the Desktop For Windows will add and modify Tivoli specific keys as well. However, applications like Software Distribution and Distributed Monitoring can use other accounts to edit/view the registry depending on modifications done by the Tivoli administrator when configuring the profile.

A.4.3.5 Creating Tasks

If there are Tivoli Administrators with the correct roles to create a Task, that user's ID will be needed to read the source script/executable (see section A.3.3, "Identifying Under Which User a Given Process Will Run" on page 606 to identify what that user ID will be).

A.4.4 Location of the oserv.exe

With TMF nodes, the `oserv.exe` process is in two locations; the `%BINDIR%\bin` and `%DBDIR%` directories. The `oserv.exe` that is used is the one in `%DBDIR%`.

A.4.5 Changes in the NT Domain

If in the event the NT Domain naming convention changes, there are several areas within Tivoli that could be effected:

- TRAA account
- idmaps using a qualified DOMAIN\user format
- Administrator logins
- Tasks
- Distributed Monitoring
- TEC Adapters using an account other than SYSTEM

A.5 Tivoli Enterprise Install and Removal

This section will discuss the specifics of an NT TMF/TMA installation and uninstall. Section A.5.5, "Preparing an NT for a Tivoli Installation" on page 624 will provide information on how to prepare an NT in advance for a Tivoli installation.

A.5.1 Installation of the Tivoli Remote Installation Package

The Tivoli TMF (and TMA through SIS) remote installation requires that the target is running either `rexec` or `rsh`. As NT does not provide for either, Tivoli introduced the Tivoli Remote Installation Package (TRIP) as part of the installation sequence to remotely install an `rexec` process on the target NT.

When the Tivoli Administrator creating the TMF/TMA node selects **Install**, the installation process takes these steps:

1. The TMR server looks up the CurrentNtRepeat machine, which is an NT ManagedNode that already has TRIP installed (TRIP does not need to be running on the CurrentNtRepeat to remotely install the rexec service).
2. Using the NT API `OpenService()`, the CurrentNtRepeat machine will check to see whether TRIP is already running on the target node. If so, it will proceed to the creation of the directories (section A.5.2, “Creation of a Tivoli Managed Node” on page 617 in this document).
3. The CurrentNtRepeat node will attempt to map the \\NODE\c$ drive using Server Message Block requests.
4. The CurrentNtRepeat will attempt the mapping of the drive as the Default Access Account user specified in the Client Install window.
5. Once mapped, the CurrentNtRepeat node will copy the necessary files to c:\Tivoli\TRIP. Once copied over, the service will be created with the command `trip -machine <target>` that is executed from the CurrentNtRepeat machine. At this point, the target NT is running `rexec` as SYSTEM.

Knowing the domain that the CurrentNtRepeat machine is important when creating new NT TMF/TMA. If the CurrentNtRepeat node is in a domain that is not trusted by the target node’s domain, the installation will fail.

Conflicts on the target NT will occur if there is already an `rexec` package or if another process has port 512. NT 3.51 SP5 introduced an enhanced spooler process that would use port 512. In this scenario, the conflicting service will need to be shutdown until after the installation is complete. In addition, other software packages, such as XSM, use their own `rexec` process which you should be aware of to look for reasons for failure to start service (Service Specific Error 8.)

REXEC’s assigned port is 512. `%SYSTEMROOT%\system32\drivers\etc\services` is an NT file that has among other services the entry for `rexec` (exec tcp512). If this entry is modified to another port, the installation will fail as the installation expects port 512.

A.5.2 Creation of a Tivoli Managed Node

Once TRIP is installed, the installation at the TMR will next send a series of scripts to the target to create the directories defined by the Tivoli administrator. The main check done here is that the file system that the install will be placed is NTFS and that there is enough disk space available. This sequence is the same using the classic installation method or the new Software Installation Service (SIS). With SIS, the Tivoli administrator is prompted for the TRAA account along with the installation directories.

A.5.2.1 Creation of Tivoli-specific Accounts on Target Node

The last step for the pre-installation is the creation of the Tivoli-specific accounts on the target NT. An executable `ntconfig.exe` is executed on the target and will run as the user defined in the Default Access Account. It will be responsible for creating the tmersrvd account as well as the TIVOLI_ADMIN_PRIVILEGE group and assign the required user rights.

A.5.2.2 Setting the TRAA Account

If this is the classic installation method, the Tivoli administrator will be prompted to enter the Tivoli Remote Access Account (TRAA) after selecting **Continue** in the preliminary portion of the install. As noted above, this account is used when a Tivoli framework process accesses a remote NT object or if you install the binaries and libraries on a remote drive. By default, no Framework processes will access remote objects, but if there is a user-customized script, or if a task is executed on the NT, the TRAA account will be necessary for the process to have access.

There are three options:

- None

  This provides no TRAA account. Beware that if this a reinstall of an NT ManagedNode, and there was a TRAA account set prior, selecting **NONE** will keep the old TRAA account intact.

- Use Default

  This will use the same account and password that was specified in the Default Access Account. This is not an ideal choice as this will grant the TRAA account the full rights that the Default Access Account has and is a potential security risk.

- Different

  Define an account other than the Default Access Account.

A.5.2.3 Installation of the Framework Files

The framework files are installed by a process on the target node called sapack. This process is rexec'd to the target and runs from the Tivoli database directory. This process runs as the user defined in the Default Access Account.

The Tivoli installation groups the files by their type, such as Binaries, message catalogs and OS-independent files. During this installation, the sapack will lay down the files directly from the incoming Network stream. There is no staging done on any of the drives.

A.5.2.4 Creation of the Client Database

The creation of the client database is broken up into two portions. The first deals with just starting the oserv on the target, and the second is the creation of the various objects on the target's database to make it a TMF node.

A.5.2.5 Starting the oserv for the First Time

The start of the oserv on the target is managed with the command `$BINDIR/TAS/INSTALL/install2.cfg`, which:

- Checks for any dispatchers on port 94
- Determines the database directory
- Creates the oserv service using the command:

  ```
  oinstall -install <drive:\path>
  ```

- Copies `$BINDIR/bin/TivoliAP.dll` to `%SYSTEMROOT%/SYSTEM32/TivoliAP.dll`
- Checks to see if TAP is available (see section 5.5 on tips to preload TAP to eliminate the initial reboot). If not, will start the oserv with a `-u` flag.
- If the host name does not match the label that Tivoli assigns to the target, it will add the name to the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform\wlocalhost`
- Starts the oserv and attempts to contact TMR server. It uses the following switches:

  i Initializes the client database

  h TMR Server label

  k Path to database directory

  b Binary directory

  l Library Directory

  u Used for NT. Bypasses TAP for the initial install.

At this point, the oserv service is running as SYSTEM. If the `-u` flag was used, all other Tivoli processes will also run as SYSTEM.

Configuring the Client Database

Once the oserv service is running, the next step is to create the objects locally to populate the database. These processes are started from the TMR server using `$BINDIR/TMF/BASESRVC/client.cfg` and will start as the built-in Administrator (v3.2 or earlier) or the `$root_user` idmap if the `-u` option is passed to the lcfd/oserv. If this a reinstalled NT, and TAP is enabled, there will be several processes running as tmersrvd.

A.5.3 Un-installing TMF

In the event that the TMF node installation failed, or a user needs to remove a fully-populated TMF node, it is important to follow the steps below to clean up all references to the TMF node.

A.5.3.1 Removing the Node From the Tivoli Database

First, before removing the node, identify the dispatcher number assigned to the NT with the `odadmin odlist` command. Then, issue the command:

```
wrmnode Name of Node
```

This command will clean up all references in the database, which includes the various subscriptions to Profile managers, ProfileManagers, and so on. If this command is completed successfully, then issue the command:

```
wchknode -ncsxvu dispatcher (number noted above)
```

This will verify that all references of the removed node no longer exist in the Tivoli Object Repository.

A.5.3.2 If the wrmnode Command Failed

If a failure in removing the TMF node occurred, there is a likelihood that all references to the TMF node have not been completely removed. Issue the following commands:

```
wrmnode Name of node -d dispatcher number
```

If this fails, then:

```
odadmin odlist objects dispatch number | wc -l
```

If the number of objects remaining are three or less, then issue the command:

```
odadmin odlist rm_od disp number
```

(If there were more than three objects, it still may be necessary to remove the node with the `odadmin odlist rm_od <disp>` command although more than seven objects is highly suspect of other problems, and it would be best to contact your support provider.

Verification of a Removed Managed Node in the Tivoli Database

There are three important locations that a TMF node is referenced. If after removing a TMF node, a Tivoli administrator could verify that the removed ManagedNode does not exist in the three locations:

```
wls -l /Library/ManagedNode
wlookup -ar ManagedNode
odadmin odlist
```

(This does not take into account subscriptions to Profile Managers, and so on.)

Removing Files From the NT

Once the Tivoli database has been cleaned up, the client oserv should be stopped. If not, stop the process. Once the oserv is stopped, it is necessary to clean up some registry entries made by Tivoli.

Remove the oserv from the service list. The key still exists, but the values are nulled:

```
oinstall -remove
```

Remove the entry in LSA for the Tivoli authentication package:

```
wsettap -d
```

Remove `trip` as a service:

```
trip -remove
```

You can keep `trip` running (and not do this step) as this will eliminate one step in the reinstall.

Issue the command `wlocalhost`. If it returns a host name, and this name is not valid, reset the value to the host name that is applicable to this machine:

```
wlocalhost new name
```

or remove this key from the registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform\localhost
```

Remove the files from the NT directory:

```
%SYSTEMROOT%\system32\drivers\etc\Tivoli
```

Path to base directory of Tivoli\Tivoli (except if you are leaving TRIP you would remove the other directories from inside the Tivoli\ directory structure, leaving the Tivoli\trip directory.)

A.5.4 Installation of the Tivoli Management Agent

This section discusses the installation of the Tivoli Management Agent (TMA) for NT. The two methods discussed are SIS and a command line style installation similar to TRIP using `winstlcf`.

A.5.4.1 Installation of TMA Using SIS

The SIS installation is similar to the creation of a TMF node. It requires that TRIP be installed on the endpoint. If TRIP does not exist, it will utilize the CurrentNtRepeat method like the TMF installation does (see section A.5.1, “Installation of the Tivoli Remote Installation Package” on page 616).

Interrogating the Target

With TRIP functioning, SIS will first push several executables into `c:\temp\deploy`. The first is `sis_sh.exe`, which is a Borne Shell for NT. The second is `worldname.exe`, which will identify the local language equivalent of Everyone.

Next, it will identify whether the `%SYSTEMROOT%\system32\kbdus.dll` exists.

Finally, a remote execution of ping will check that the node is able to resolve the TMR server’s host name.

Creation of Tivoli-Specific Accounts

The executable `ntconfig.exe` (found in the TRIP directory) will be executed, creating the tmersrvd account and the Tivoli_Admin_Privileges group.

Seed File Created

Using an executable called `sapack.exe`, a file called w32-ix86_seed is sent to `c:\temp\deploy`. This file is used for the initial start of the `lcfd` service.

lcfd Installed

Once the seed file is in place, the `lcfd.exe` is executed with the `–I` option enabling it as a service. Once completed, it will place several files outside of the base TMA directory:

```
%SYSTEMROOT%\system32\TivoliAP.dll

%SYSTEMROOT%\Tivoli\lcf\1\lcf_env.cmd
```

Finally, it will register the Tivoli Authentication Package with LSA using `wlcftap.exe -a`.

lcfd Starts and Logs in to Gateway

Once TAP has been enabled, `lcfd` is started with the parameters passed to it by the seed file as well as any user-defined parameters. TMA will attempt to log into a gateway.

Once completed, the NT will need to be rebooted for TAP to be loaded by LSA. If this was a previous installation of TMA or is a ManagedNode being migrated to TMA, the NT will not need to be rebooted since TAP has already been recognized by the LSA.

A.5.4.2 Installation Using winstlcf

`winstlcf` is a `perl` script that can be issued from the TMR server or any gateway/TMF node. This method is similar to a TRIP install as it uses the Server Message Block (SMB) protocol to install and enable TMA on NT endpoints.

Sequence of Events

The command is issued similarly to this:

```
winstlcf -l 3656 -g 1.1.1.2+3646 -N binkley serene
```

`Binkley` is the system that copies the files to the target. It is called the proxy node. This proxy node must already be defined as an endpoint. `Serene` is the target NT.

The `-g` option is to identify a gateway. If `-g` is not used, the `lcfd` will broadcast through the UDP to attempt to contact a gateway.

The steps for the installation are as follows:

1. Gateway that manages the proxy endpoint responsible for copying the files is contacted.
2. Installing system contacts target node using the account passed to the `winstlcf` script.
3. The drive is mapped to the proxy, and the Tivoli directories are created on the target node. The executables are copied over to the `lcf` directories:
   - `bin\w32-ix86\mrt\lcfd.exe`
   - `bin\w32-ix86\mrt\lcfep.exe`
   - `bin\w32-ix86\mrt\libcpl.dll`
   - `bin\w32-ix86\mrt\libdes.dll`
   - `bin\w32-ix86\mrt\libmrt.dll`
   - `bin\w32-ix86\mrt\msvcrt40.dll`
4. Through the Server Message Block (SMB), the `lcfd` is installed as a service and is started as the user used for the installation.

5. At this point, the installing endpoint is complete and logs off.
6. Once the `lcfd` on the target is started, it will contact the gateway and the login starts. Once checked in, the gateway downloads the following files:
   - `bin/w32-ix86/endpoint/ NTLCFInst.exe`
   - `bin/w32-ix86/endpoint/wlcftap.exe`
   - `/bin/w32-ix86/endpoint/TivoliAP.dll`
   - `/bin/w32-ix86/endpoint/ntconfig.exe`
   - `/bin/w32-ix86/endpoint/libacct.dll`
   - `/bin/w32-ix86/endpoint/reboot.exe`
7. The `lcfd` executes ntconfig.exe to create the accounts
8. `lcfd` executes `wlcftap` to enable TAP and set the TRAA account if required. If TAP was previously installed, the TRAA account may still exist from the previous installation.
9. `Lcfd` will reboot the system if it was instructed to

There is a limited amount of logging available using this command. If there are failures, check `lcfhost.err`. This file is located in the present working directory where `winstlcf` is issued from. If the `lcfd` process is not starting, set the trace level to 3 on the TMA node. To do this, either manually start `lcfd` with the option `–d 3`, or change the log_threshold =3 in `last.cfg`.

A.5.4.3 Removing TMA

In the event that TMA needs to be removed, follow these steps to remove TMA from the NT node:

Removing the Endpoint from the Tivoli Enterprise Database

First, remove the endpoint by issuing the command:

```
wdelep EndpointName
```

Removing the Files

Once the endpoint has been removed from the Tivoli database, issue the following commands to remove the TMA files and modifications.

If TMA was installed from the CD (InstallShield), then from the `%LCFROOT%` directory:

```
uninst.bat
```

If TMA was installed remotely, there will be no `uninst.bat` file. Follow these steps to manually remove the TMA installation. Remove TAP from the LSA:

```
wlcftap –r
```

If TMA and TMF are installed on the same machine, do not remove TAP; this will effect the TMF installation on the NT.

Remove `lcfd` from the services list

```
lcfd –r “lcfd”
```

Remove the icon from the TaskBar

```
lcfep –s
```

Finally, remove the files from the NT located in:

- `%SYSTEMROOT%\system32\drivers\etc\Tivoli`
- `%SYSTEMROOT%\Tivoli\lcf`
- Directory to `lcf base\lcf`

A.5.5 Preparing an NT for a Tivoli Installation

If possible, there are several steps that can be taken during the creation of the NT workstations/servers that will assist in the time to create a TMF node and to not have to reboot the NT when it becomes a TMF/TMA node.

A.5.5.1 Loading TAP

The reason that a newly created TMF/TMA node must reboot is for the `TivoliAP.dll` to be loaded by `LSASS.exe` (LSA subsystem). Once loaded, the Tivoli Authentication Package is available to generate login tokens for the `oserv/lcfd` service. If a gold build of NT is developed to provide a consistent level of software to all NT server/workstations being deployed, it is possible to have the `TivoliAP.dll` loaded and enabled as well. This will allow the NT to become a TMF/TMA node later on, and not have to reboot after the installation, since TAP is available already. To do this, copy the `TivoliAP.dll` to `%SYSTEMROOT%\system32` and issue `wsettap -a` (or, create a script that would edit the registry directly). The command `wsettap -a` does not require an `oserv/lcfd`; so, this could be done at the time the NT image is being installed.

A.5.5.2 Loading Tivoli Files Prior to Using SIS/Classic Install

In environments where slow links exist, creation of Managed Nodes can be problematic due to line speed and time. To anticipate a creation of a Managed Node, the NT node being shipped to the remote site can have the entire Tivoli directory copied from an existing Managed Node. Once done, the Database directory would be deleted. When the NT is to be created as a Managed Node, and the directory paths have been specified to match the target’s already-installed files, the only portion that the install needs to do is the database portion. This can significantly decrease the time required for installing the Managed Node.

A.6 Environment Issues

There are several environmental concerns pertaining to NT that may effect TMF/TMA.

A.6.1 DLL Conflicts

There are incompatibilities with different versions of MSVCRT40.DLL shipped from Microsoft (which seems to vary from software package to software package). Tivoli installs Version 4.0 of the DLL. If Tivoli processes start to fail or processes, such as `ntprocinfo.exe` ,utilize 100 percent of the CPU, this would suggest a version of MSVCRT40.DLL that is not backwards compatible has been introduced into the `%SYSTEMROOT%\system32` directory.

This is an issue only with the TMF nodes. TMA nodes have the MSVCRT40.dll installed in the TMA directories and, thereby, eliminate possible conflicts.

To correct this on NTs that see this issue, follow these steps:

Version 3.1.3 or 3.1.4, 3.2 SuperPatch, 3.6:

```
copy %BINDIR%\mslib\msvcrt40.dll %DBDIR%
copy %BINDIR%\mslib\msvcrt40.dll %BINDIR%\tools
copy %BINDIR%\mslib\msvcrt40.dll %BINDIR%\bin
```

Version 3.1.x (other than 3.1.3) and Version 3.2 Framework:

```
copy %BINDIR%\mslib\msvcrt40.dll %DBDIR%
copy %BINDIR%\mslib\msvcrt40.dll %BINDIR%\tools
copy %BINDIR%\mslib\msvcrt40.dll %BINDIR%\bin
cacls %DBDIR%\msvcrt40.dll /g everyone:r  (Grants read rights to the file)
```

At this point, Tivoli processes will be insulated to the changed MSVCRT40.DLL in the `%SYSTEMROOT%\system32` (after a restart of the oserv).

Loading of DLLs works differently in Windows NT than previous versions of Windows. NT loads a DLL separately for each process because each application has its own address space in Windows NT; the address space is shared in 16-bit Windows. (See Microsoft Development Network, or Article ID Q100635).

The way the `LoadLibrary()` works in NT is as follows: When no path is specified, the function searches for the file in the following sequence:

1. The directory from which the application loaded.
2. The current directory.

3. The 32-bit Windows system directory. Use the **GetSystemDirectory** function to get the path of this directory. The name of this directory is SYSTEM32.
4. The 16-bit Windows system directory.
5. There is no Win32 function that obtains the path of this directory, but it is searched. The name of this directory is SYSTEM.
6. The Windows directory. Use the **GetWindowsDirectory** function to get the path of this directory.
7. The directories that are listed in the PATH environment variable.

To confirm this behavior, use a tool, such as `listdlls.exe`, from `http://www.sysinternals.com`. Processes like `lsass` will use the `%SYSTEMROOT%\system32\msvcrt40.dll`, and Tivoli processes like `oserv` will use `%DBDIR%\msvcrt40.dll`.

Another method is to use `tlist <Process Name>` and locate the version of the DLL in question. Tivoli ships Version 4.0.0.5270.

A.6.2 How Shell and Perl Scripts Work on NT

Shell and Perl scripts within Tivoli start with the line `#!/bin/sh` or `#!/etc/Tivoli/bin/perl`. Although these paths do not exist on NT, Tivoli Enterprise will catch these references and direct them to the correct executable.

On TMF, the `oserv` process will read the line and if it sees either `sh` or `perl`, it will use the `perl` and `sh` found in `%BINDIR%\tools` (TMF). TMA will rely on the dependencies when `sh` or `perl` are encountered (see section A.6.3).

Because Tivoli uses `#!/bin/sh`, it is important to not replace Tivoli's version of `bash.exe` and `sh.exe`. If Enterprise needs dictate another version of `sh.exe`, place it in a separate location and adjust the NT PATH environment to include that version. Tivoli's implementation will use the Tivoli-supplied `bash.exe/sh.exe`.

A.6.3 Dependencies and TMA

Because TMA endpoints do not contain the various tools included with TMF (`%BINDIR%\tools`), it will be necessary to create dependencies to provide support to scripts using commands not found with the standard NT/TMA release.

On TMA, the Framework 3.6 release notes describe how to create dependencies. For example, the method run_task needs to have a dependency on bash (sh) in order for tasks to execute a script using `#!/bin/sh`. To set this up:

```
wdepset -c task-library-tools -a w32-ix86 bin/w32-ix86/tools/sh.exe +a +p %TOOLS%
wdepset -e @DependencyMgr:task-library-tools -a \
   w32-ix86 bin/w32-ix86/tools/win32gnu.dll +a +p %TOOLS%
wchdep @Classes:TaskEndpoint @DependencyMgr:task-library-tools run_task
wgateway your_gateway dbcheck
```

You can issue the `wdepset -e` command for each tool you want to be downloaded when a task is run, such as `sed` and `awk`, or for `perl`. You must issue a `wgateway dbcheck` against all gateways whenever you update dependencies so that the new method headers are cached.

If Enterprise requirements dictate that Tivoli products like tasks will use a set of tools on the TMA nodes, it may be best to create a FilePackage that is then distributed to each node after its initial creation. This may prove an easier method of providing tools to the nodes rather than using dependencies.

A.6.4 Name Resolution/WINS

NT offers several means to resolve host names. Tivoli utilizes the standard `gethostbyname()` and `gethostbyaddr()` system calls to resolve name and IP address. When this is passed to NT, it will use not only DNS (if configured), but WINS, hosts, and LMHOSTS. If a failure occurs, be sure that the NT is properly configured for name resolution, such as Enable DNS for Windows Resolution in the TCP/IP properties. One note about WINS: Although WINS is a valid resolver, WINS database is not static by default and should not be relied on as the primary resolver when the NT is a ManagedNode.

There are several reports of problems with using Fully Qualified Domain Names. From a post on `NTBUGTRAQ's` listserv (`http://www.ntbugtraq.com`) reports that a 15 character FQDN will fail to resolve on NT due to NetBios naming conflicts.

A.6.5 Sourcing the Tivoli Environment

If a user wishes to source the Tivoli environment, there are setup files located on the node:

TMF `%SYSTEMROOT%\system32\drivers\etc\Tivoli\setup_env.cmd`

TMA `%SYSTEMROOT%\Tivoli\lcf\1\lcf_env.cmd`

If one would like to have a Command shell that sources the Tivoli Environment when invoked, create a shortcut to the cmd.exe executable and

select **Properties**. Under Shortcut, append to `cmd.exe /k <path to file>` in the Target Field. For example, a TMF file being sourced would use:

```
cmd /k %SYSTEMROOT%\system32\drivers\etc\Tivoli\setup_env.cmd
```

A.6.6 Tivoli Desktop for TMF

The Tivoli desktop is not installed with the NT TMF or TMA. To install the Windows version of the desktop, initiate `<CDROM>\PC\Desktop\Disk1\setup.exe`. This will install the Tivoli desktop application.

It is still possible to point an existing Windows Tivoli desktop at an NT Managed Node even if the Desktop For Windows is not installed on that NT. The Managed Node is capable of supporting remote desktops.

TMA nodes do not support remote desktops.

A.6.7 Performance Tuning for Tivoli

Although not important to TMA, there are several NT tuning parameters useful to TMF nodes serving as repeaters, home hosts to PC managed nodes, gateways, or to an NT TMR or TEC server.

ControlPanel>Network>Services>Server>Properties. This section allows the tuning of system-wide caches. The ideal setting is Maximize Throughput for Network Applications.

NT 4.0 - **ControlPanel>System>Performance.** Set the **Performance Boost** to the low setting if possible.

NT 3.5 - **ControlPanel>System>Tasking** Set to **Foreground and Background are equally responsive**.

Screensaver Note

GL Pipes are useful, but this will impact the performance of NT (It appears to utilize all 100 percent of the CPU as seen using `perfmon`.

A.6.8 Non-US Keyboard Issue

If an NT TMF/TMA install fails with the error:

```
Creating separate WindowStation and DeskTop. Create WindowStation failed.
The specified module could not be found.
```

This is due to the input locales. Copy `kbdus.dll` to `%SYSTEMROOT%\system32`. The US keyboard or locale does not have to be selected in Windows configuration; the DLL just needs to be present in the system32 directory.

A.6.9 Port Restriction Causes TIME_WAIT to Last 169 Seconds

Tivoli provides a means to restrict port usage to a given range (`odadmin` will reveal whether port restrictions are in use). On NT, the restrictions will cause closed TCP connections to persist 169 seconds in a TIME_WAIT state.

To minimize the delay, one can edit the NT's TCP settings as follows:

1. Locate the key:

   ```
   HKEY_LOCAL_MACHINE/System/CurrentControlSet/Services/Tcpip/Parameters
   ```

2. Click on **Edit/Add Value**.
3. You then enter the Value Name: `TcpTimedWaitDelay`.
4. Change the data type from its default REG_SZ to REG_DWORD.
5. When you click on **OK**, it will then ask you for Data:

   ```
   60 (decimal) for 60 seconds
   ```

Then reboot. All TIMED_WAIT'ed ports will disappear in 60 seconds.

A.6.10 Tivoli Files Placed Under %SYSTEMROOT%

This section lists various files that are placed by Tivoli under the %SYSTEMROOT% directory.

Table 39. Tivoli Files Placed in %SYSTEMROOT%

Component	Files (size file name)
Tivoli Remote Installation Package	None
Tivoli Management Agent	%SYSTEMROOT%\system32 496,640 sis_sh.exe, 32,256 TivoliAP.dll, 7,168 worldname.exe
	%SYSTEMROOT%\Tivoli\1\lcf 727 lcf_env.cmd, 1,782 lcf_env.sh (Sizes of files will vary)

Component	Files (size file name)
Tivoli Management Framework	%SYSTEMROOT%\system32 496,640 sis_sh.exe, 32,256 TivoliAP.dll, 7,168 worldname.exe
	%SYSTEMROOT\system32\drivers\etc\Tivoli 857 setup_env.cmd, 1,120 setup_env.sh, 510 tll.conf\arg, 443 tll.conf\layout, 496 tll.conf\library, tll.conf\task (Sizes of files will vary)
Remote Control Server	None
Remote Control Controller	%SYSTEMROOT% 46,080 RCSERV.EXE
	%SYSTEMROOT%\system32 4,096 EQNMSG.DLL
Remote Control Target	%SYSTEMROOT% 26,624 eqnhook.dll, 46,080 RCSERV.EXE
	%SYSTEMROOT%\system32 1,457 Command Prompt.lnk Modified, 17,408 VDD.DLL, 7,168 VDDFIFO.DLL, 12,288 VDDHOOK.DLL
	%SYSTEMROOT%\system32\drivers 21,024 keyex.sys, 20,288 mouex.sys, 8,288 tgrab.sys
Security Management	None
User Administration	None
Software Distribution	None
Distributed Monitoring	None
Enterprise Console	None
Inventory	None

A.7 Tivoli Specific Commands and Terminology for NT

Below are a list of commands and concepts that are unique to the NT environment:

Table 40. Tivoli-Specific Commands and Terminology for NT

Command/Entity	Description
wrunui.exe (TMF only)	Executable that allows a script, sentry monitor, or task to execute graphic-based application on the NT console. (Such as wrunui notepad)
bash.exe	Bourne Again Shell. Provides full bourne shell facilities. This is located in %BINDIR%\tools. There is also sh.exe, which is the renamed bash executable.
wsettap.exe(TMF) or wlcftap.exe(TMA)	Has 2 roles: Sets the TRAA account for access to remote access of NT objects: wsettap -r DOMAIN\fred sets TRAA to that domain account wlcftap –r "" nulls TRAA account Activates and disables the Tivoli Authentication Package wsettap -d disables TAP wlcftap -a activates TAP
smtp_client.exe (TMF only)	Provides a mail client for NT. Useful for scripts that want to initiate a mail message as a result of a script. For example, smtp_client mhahn@support < c:\temp\file. If smtp_client is issued with no email address, it will look for the value in: HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform\mailhost .This key can be set using wmailhost.exe.
TRAA (Tivoli Remote Access Account)	An account and password stored locally on NT that is used when a Tivoli process accesses a remote object, or if the Tivoli binaries and libraries are installed on a remote share. TRAA account is not needed unless customer design dictates access to these remote NT objects (like Netshares).
TRIP (Tivoli Remote Installation Package)	Provides rexec functionality for installation of ManagedNode or TMA using SIS or the classic install method. Once installed, provides remote start-up of the oserv (odadmin start <disp>) and for remote interconnects of TMRs. TRIP can be disabled after installation if remote start-up of oserv service is not needed.
wlocalhost.exe (TMF only)	Command sets the wlocalhost key and value in registry: HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform\wlocalhost. Key is used when NT is a failover machine or when the host name differs from the Tivoli label of the ManagedNode.

Command/Entity	Description
wmailhost.exe (TMF only)	Command to specify a SMTP server. TMF nodes use this reference when the wsupport or smtp_client command is used as well as applications like Distributed Monitoring. It stores the SMTP server in: HKEY_LOCAL_MACHINE\SOFTWARE\Tivoli\Platform\mailhost When attempting to connect to SMTP server, will first try the host referenced with wmailhost, then the local host, and then the host that the given email address specifies (fred@bedrock)
ntprocinfo.exe	Provides a list of processes running on the NT. There are significant enhancements in the ntprocinfo command for 3.2 SuperPatch, and 3.6. ntprocinfo.exe will now correctly show the user that the process runs as, much like nthandleex.exe.

A.8 Useful Microsoft and Third Party NT Commands

There are several utilities available that assist and enhance the Tivoli environment.

A.8.1 Built-in NT Commands

`ipconfig -all` Provides Network information related to TCP/IP.

`netstat` Provides router and port information.

`nbtstat` Displays protocol statistics and current TCP/IP connections using NBT (NetBIOS over TCP/IP).

`perfmon` Useful to obtain overall system resource usage.

A.8.2 Other Utilities

Two excellent resources are: Microsoft's Resource Kit (NTResKit) and the collection of shareware at `http://www.sysinternal.com`.

Some useful commands from the `NTResKit`:

`tlist.exe` process list.

`telnetd.exe` Telnet service.

`sc.exe` NT Service Controller.

`netsvc.exe` NT Service Controller.

Some useful commands from `http://www.sysinternals.com`:

`listdll.exe` Shows a list of processes and the DLLs each process has loaded.

`nthandleex.exe` Shows the processes running, the user the process is running as, and open handles.

`ntregmon.exe` Shows what processes are accessing the registry.

An excellent remote command line interface called XLNT is available from Sunbelt Software. This allows you to have a completely command line based interface into your NT systems with support for multiple simultaneous logins. See `http://www.ntsoftdist.com/axlnt.htm` for more details.

A.9 General Issues

Below is a general list of Tivoli issues encountered on NT.

Note

Due to the recent release of TMA, this list currently has no real issues specific to TMA. The original paper from which this appendix came will continue to be updated to address 3.6 issues. Updates can be obtained from `http://www.support.tivoli.com`.

A.9.1 Issues with TAP

TAP issues typically fall into two areas: Initial start-up of the oserv and spawning processes as invalid accounts. The command `wsettap`/`wlcftap` can be issued regardless of the oserv's state (up or down). To issue `wsettap`/`wlcftap`, the caller must be a member of both the Administrators group and the Tivoli_Admin_Privileges group. If a user gets the error `Access is denied`, this is a result of the user not being part of the two groups.

A.9.1.1 Failure to Start oserv

Failures starting the oserv will generally receive error 1067. In all cases, review the `%DBDIR%\oservlog` on the failing NT for a better explanation. Below are several errors and the solution to restart the oserv:

- Tivoli Authentication Package is not properly installed or loaded by LSA. The error is:

  ```
  !tap_init_failed A specified authentication package is unknown.
  TAP is not known to the LSA subsystem.
  ```

  TAP is not listed in the Authentication Package key, issue the command:

  ```
  wsettap -a or wlcftap –a
  ```

then reboot the machine.

- Another Authentication Package is listed before the Tivoli Authentication Package.

  If this is the case, move TAP to the second position (After msv1_0) in:
  `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\AuthenticationPackages`

- `tap_call_init` failed, Error 2221

  The TRAA account no longer exists. Reset the TRAA account to a valid account:

  ```
  wsettap -r <DOMAIN>\<user>
  ```

  or

  ```
  wsettap -r ""   (This will null the TRAA account)
  ```

- `tap_init_failed` Error 38:

  The privileged account is not part of the Tivoli_Admin_Privileges group.

- `!tap_call_init` failed The user must change their password before they log on the first time:

  The TRAA account is being forced to reset password. Disable this feature.

- `tap_get_sid_logon_token` failed: No mapping between account names and security IDs was done.

  The tmersrvd account has been removed.

- `tap_get_sid_logon_token` failed: Logon failure: the user has not been granted the requested logon type at this computer.

  The tmersrvd account does not have `Log on Locally` as a UserRight

- `CreateWindowStation` failed: Access is denied:

  `tmersrvd` is denied access to the `%SYSTEMROOT%` directory. Verify that `tmersrvd` has Bypass Traverse Checking on and that the account is not specifically denied access.

A.9.1.2 Executing Processes Using Invalid Account

If the Tivoli Framework attempts to spawn a process as an invalid user, such as invoking a task or a sentry monitor, the following errors can occur:

- `user_string_to_sid` failed with code `77`

  Account used does not exist, or that the idmap does not have the correct entry for the w32-ix86 interpreter (widmap). This is common with tasks where the task is specified to run within the context of a given UID.

- `tap_make_sid_logon_token` failed with code `77`

Either the account is not allowed to log on locally, in which case, you need to add this User Right so that account in question, or the account is disabled. Be sure account has **Password Never Expired** set, or the account has **User Must Change Password** set. Disable this.

- Logon failure: unknown user name or bad password:

  Reset the account on the NT or Synchronize the domain.

- Logon failure: the user has not been granted the requested logon type at this computer:

  Either the TRAA user is not allowed to access a share, or the TRAA user is not able to access the computer from the Network. Check the share permissions and/or the UserRights. This can also occur if a task's UID does not have the **Logon Locally** set.

- `LookUpAccount` failed:

Account does not exist on local NT or in the domain

A.9.2 Start-Up of oserv

Examples of possible problems and their solutions are as follows:

- `@odinit: Unable to establish connection to ALI !oserv: odlist init failed. IPC Shutdown (67)`.

  There is already a connection from the NT to the TMR server that is currently in a TIME_WAIT state. Issue `netstat -a` and verify the TIME_WAIT state with the TMR server's port 94.

  This is a problem with TMP3.2 when port restrictions are in use. Apply patch `3.2-TMP-0028`.

- `!oserv: odlist init failed. requested resource not found (30)`

  This can cause three possible problems:

  - TAP has the TRAA account incorrectly set. Reset the TRAA account.
  - Name resolution for the NT is incorrect. The NT is unable to lookup its IP address and host name correctly. Add a line to `%SYSTEMROOT%\system32\drivers\etc\hosts` with the correct host name and IP address for the NT.
  - If running Version 3.2 of the framework, apply patch **3.2-TMF-0028**.

- `Bind failed winsock_comm.c No such File or directory:`

  The `wlocalhost` registry key is incorrectly set. Issue the command:

  `wlocalhost <correct name of NT>`

- `Application <TMF_Sched, TNR_prog1, mannode_skel> failed to initialize`

This is a DLL conflict with `MSVCRT40.DLL`. See A.6.1, “DLL Conflicts” on page 625.

A.9.3 Using TRAA with Tasks

Possible errors include:

- `Access is denied executing task`

The TRAA account does not have permissions to access a remote share.

- User sees a phantom mapped drive.

  If a task is run on an NT, and the script maps a drive using the net use `DRIVE` command, and the script does not delete the map, a user will see the drive mapped and will be unable to delete the mapped drive.

  Although mapping and deleting the map for a task is acceptable, the other alternative that will alleviate p599

  roblems with the ghost drive to an existing drive letter is to use the `\\MACHINE\share` format.

A.9.4 General Framework

Possible errors include:

- Unhandled exception error in oservlog, oserv goes down

  This is often encountered when an NT is a home host for PcManagedNodes, and an inventory scan was initiated. Setting **max_conn** to 20 resolves the issue. The problem is that the number of active method threads approaches 200. At this point, the oserv is unable to maintain this high number of processes.

  The fix is in the 3.2 SuperPatch and Version 3.6.

- Error in oservlog regarding `hurl`

  This problem can occur if the NT serves as a TMR server and a Network intensive task or a `wchkdb -ux` was invoked. Increase the number of rpc_max_threads to 20 more than the total number of your ManagedNodes (`wlookup -ar ManagedNode`).

A.10 PC Agent Overview

This section will discuss the PC agent implementation on NT. It is important to stress that the PC agent does not use TAP and has no functional equivalent to TRAA.

A.10.1 PC Agent Design

The PC agent was introduced in Version 1.0 of Tivoli Management Environment (precursor to Tivoli Enterprise). This product allowed limited management of PC-class systems, including Windows 3.1/95, OS2, and Windows NT. Software Distribution, Inventory, Remote Control, UserAdmin, and Tasks were applications that were written to communicate to the PC agent and execute actions on these endpoints.

A.10.2 PC Agent Running as a Console Application

The PC agent on NT can be installed as either a service or as a Console Application. When the PC agent is installed as a console application, it is started up with the SID of the user that logged in and assumes all credentials as the user. If the user logged in with a domain account, or has mapped a remote share with a domain account, the PC agent will have that user's rights on the local and remote resources.

If that user logs out, the PcAgent will be stopped and no communication will exist between the PcAgent and Tivoli Enterprise.

A.10.3 PC Agent Running as Service

In many environments, the PC agent is usually installed as a NT Service, as this provides consistent communication between the PCManagedNode and the TMR since the PC agent is never shutdown when users log out. The PC agent runs as NT AUTHORITY\SYSTEM. This account provides privileged rights to the local system. However, if the PC agent attempts to access a remote share, the agent will fail, as the NT AUTHORITY\SYSTEM account has no privileges to remote resources.

A.10.4 PC Agent Running as a User-Defined Account

One way to get around the inability of the PC agent to access remote resources is to install the PC agent as a service and set it to run as another user. The user would need to be a domain Administrator in order for the PC agent to properly access these remote shares. Furthermore, the defined user's password could not change, as NT statically stores the password in the local NT registry. Therefore, once set in the Services Control Panel, if that

user account's password changed, the PC agent would not be able to start again until it was modified in the NT service control panel.

A.11 Version 3.6 Methods Using $root_user idmap

This section lists the TMF/TMA methods that use the `$root_user` idmap for Version 3.6. This list was compiled with the following applications: Inventory, Security Management, Distributed Monitoring, Software Distribution, User Administration, Remote Control, Adapter Configuration Facility, and Enterprise Console.

Table 41. Methods That Use the $root_user ID Map

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
Adapter Configuration Facility			
ACPEP			
	acpEp	Edit adapter files	/TME/ACP/acpep
	acpEpRmFiles	Removes adapter files	/TME/ACP/acpep
	install_gateway	Installs upcall collector on Gateway/TME/ACP/acpep_install	
	install_logfile	Installs logfile adapter	/TME/ACP/acpep_install
	install_nt	Installs the NT Event Adapter	TME/ACP/acpep_install
	install_snmp	Installs the SNMP Adapter	/TME/ACP/acpep_install
TMA nodes			
Endpoint			
	admin	Admin related methods	/endpoint/admin
	write_html	HTML management	/endpoint/msg_bind
Inventory Profiles on TMR Server			
InventoryProfile			
	_get_validation_enabled	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	abort_lock_acquisition	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
	copy_all_actions	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	copy_records	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	disable_validation	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	edit_acl_check	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	enable_validation	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	get_default_policies	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	get_validation_policies	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	initializ	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	ip_push	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	ip_sched_push	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	move_records	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	populate	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	push	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	remove	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	set_default_policies	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	set_excluded_dirs	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	set_included_dirs	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
	set_validation_policies	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
	validate	Creation/push of profile	/TME/INVENTORY/INV_PROF/inv_prof_meths
Inventory Endpoint			
InventoryProfileEndpoint			
	ip_discover	Scan	/TME/INVENTORY/INV_ENDPT/inv_endpt_meths
	ip_reset_last_mif_file	MIF	/TME/INVENTORY/INV_ENDPT/inv_endpt_meths
Inventory Scan from ManagedNode Serving as NWMS Host			
InventoryProfileNWMSEndpoint			
	ip_discover	Scan	/TME/INVENTORY/INV_ENDPT/inv_nwms_endpt_meths
	ip_reset_last_mif_file	MIF	/TME/INVENTORY/INV_ENDPT/inv_nwms_endpt_meths
Inventory Scan from ManagedNode Serving as PCMN Homehost			
InventoryProfilePCEndpoint			
	ip_discover	Scan	/TME/INVENTORY/INV_ENDPT/inv_pc_endpt_meths
	ip_reset_last_mif_file	MIF	/TME/INVENTORY/INV_ENDPT/inv_pc_endpt_meths
Endpoint			
InventoryUserLink			
	lcf_create_dir		/generic/TME/INVENTORY/USERLINK/lcf_create_dir
	lcf_send_file		/generic/TME/INVENTORY/USERLINK/lcf_send_file
	lcf_view_run_dir		/generic/TME/INVENTORY/USERLINK/lcf_view_run_dir

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
Installation of Endpoint Using winstlcf. Methods Running from Proxy Endpoint			
LCF-NtLcfInstall			
	configure	Setup lcf.dat/lcf_env.cmd files	/endpoint/NtLcfInst
	inspect	Initial inspection	/endpoint/NtLcfInst
	install	Copy of endpoint code	/endpoint/NtLcfInst
	reboot	Reboot system	/endpoint/NtLcfInst
Modification of Distributed Monitoring Endpoint Engine			
SentryEndpoint			
	boot_engine	Start engine	/TME/SENTRY/dogEndpoint
	engineUpdate	Update monitors on engine	/TME/SENTRY/dogEndpoint
Creation/Modification/push/execution of Distributed Monitoring Engine (All methods for class SentryEngine will use idmap, so run_engine is only one listed as this is representative of all other methods)			
SentryEngine			
	run_engine	SentryEngine running	/TME/SENTRY/sentry_engine
Software Distribution			
TMF_FP			
	fp_dist	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fp_operation	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fpblock_dist	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fpblock_target_preview	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_cancel	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_fpblock	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_install	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
	fps_list	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_size	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_uninstall	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
	fps_want_list	Distribution modification/action	/TAS/MANAGED_NODE/fp_endpoint
Task Execution on TMA/TMF Nodes			
TaskEndpoint			
	batch_run_task	Execute task	/TAS/TASK_LIBRARY/task_endpoint
	run_task	Execute task	/TAS/TASK_LIBRARY/task_endpoint
Task Management			
TaskRepository			
	change_task		/TAS/TASK_LIBRARY/repository_skel1
	create_job		/TAS/TASK_LIBRARY/repository_skel1
	create_task		/TAS/TASK_LIBRARY/repository_skel1
	delete_task		/TAS/TASK_LIBRARY/repository_skel1
	disconnect		/TAS/TASK_LIBRARY/repository_skel1
	dump_tasks		/TAS/TASK_LIBRARY/repository_skel1
	export_task		/TAS/TASK_LIBRARY/repository_skel1
	import_task		/TAS/TASK_LIBRARY/repository_skel1

CLASS	METHOD	USER ACTION	EXECUTABLE relative to %BINDIR%/%LCF_BINDIR%
	lookup_task		/TAS/TASK_LIBRARY/repository_skel1
	update_resources		/TAS/TASK_LIBRARY/repository_skel1
UserProfiles			
UserManagement			
	UserProfile_synchronize	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	UserProfile_verify	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	change_password	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_discover	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_discover_ext	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_gen_strlist	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_runcmd	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_set_login	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1
	um_update	Manage UserProfiles	/TME/USERMANAGEMENT/umbo_skel1

Appendix B. RDBMS Management

Note

This chapter has NOT been updated from the previous edition of this book. The ESM Database Management Modules are now called Tivoli Management Module for Oracle (and Sybase, DB2, and so on).

This appendix is included as much of the information still applies to the current products.

For some time, a UK company called DBMX provided TME 10/Plus modules to manage database applications. Since the merger of DBMX with Tivoli in 1997, the company's line of products has grown and strengthened, and at the time of writing, were known collectively as Enterprise Server Management (ESM) Database Management. There are four different databases supported by TME 10 modules, and for each module, there are selectively-installable management components:

Table 42. Supported Databases and Components for ESM

Databases	Framework	User Administration	Distributed Monitoring
Oracle	Yes	Yes	Yes
Sybase	Yes	No	Yes
Microsoft SQL	Yes	No	Yes
Informix	Yes	No	Yes

The ESM product names are:

- TME 10 Module for Sybase
- TME 10 Module for Oracle
- TME 10 Module for MS SQL
- TME 10 Module for Informix

Note

The names for files, commands, and directory locations have slightly changed since the pre-merger versions. In this chapter, we are referring to the new TME 10 product versions only.

In this chapter, we use Oracle 7 as the database for all the examples. Much of what is stated also applies to the other database products.

As with any interaction with database products, the installation and configuration of the ESM products is likely to require the assistance of the Database Administrator (DBA) if one is designated.

B.1 Installation

Each component that is going to be used has to be installed on the TMR server and on all clients where you want to manage a database. The client where the database resides has to be a managed node. The installation process is documented in each product's Framework guide, for example, the *ESM Sybase SQL Server Framework Guide*. This section contains an outline of the installation process in order to highlight important details.

After the installation of the code, there will not be a new icon on the desktop. The new objects will be created after the registration to the database. To do this, it is recommended to create a new policy region for the database management.

To work with the database, the administrator needs the proper roles. There are some new TMR administrator roles for these products, such as sybase_dba, which should be given to the database administrator.

Without having these TMR roles applied to the database administrator, you will not be able to register the database.

Also, the database management policy region needs to have special managed resources assigned to get access to the database, such as OracleDatabase. To activate Sentry profiles, the managed resources ProfileManager and SentryProfile must be set as well.

To work with the database, a database object needs to be created by registering the database with the TMR server. The installation added the objects to the Create menu for a policy region, as shown in Figure 237:

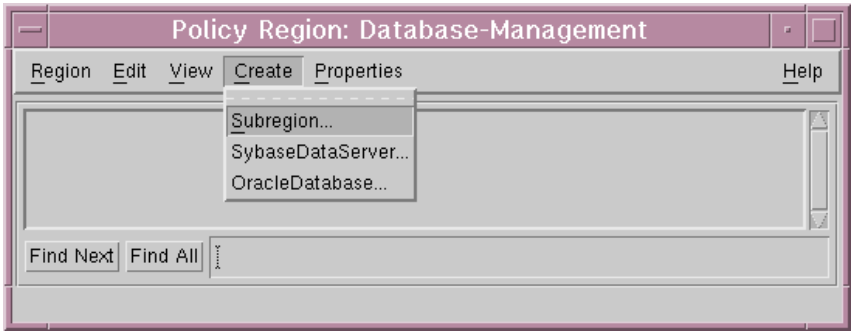


Figure 237. Create ESM Database Object

The menu items in figure 237 will appear only when the proper managed resources are given to the Database Management Policy Region. Clicking, for example, the **Oracle Database** menu item opens a dialog box similar to the one shown in Figure 238:

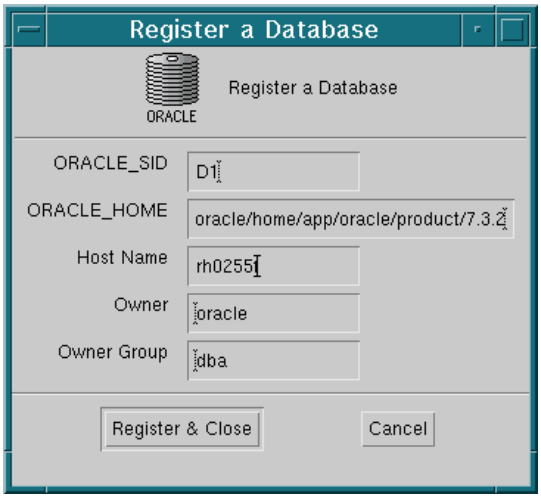


Figure 238. Registering Dialog for Oracle Database

Figure 238 is an example dialog for the Oracle database registration. There are five fields, and all of them are mandatory and important. More information on Oracle can be found in Appendix C, “RDBMS Install Examples” on page 679.

- **ORACLE_SID**

This is an Oracle environment variable and can be set uniquely by the database administrator. You can find it in several files:

- `/home/oracle.profile`
- `$ORACLE_HOME/network/admin/tnsname.ora`
- `$ORACLE_HOME/network/listener.ora`

This applies for UNIX and NT.

- **ORACLE_HOME**

  This is also an Oracle environment variable, set by the database administrator, and you can find this in `/home/oracle.profile`.

  Look for the *tnslsnr** file and go back one step in the directory path, which will give you the ORACLE_HOME directory.

- **Host Name**

  This is the IP-hostname of the machine hosting the Oracle server.

- **Owner**

  On UNIX systems the Owner usually defaults to oracle because this is the Oracle user ID. On Windows NT systems, you should leave this field unchanged.

- **Owner Group**

  On UNIX systems, the group usually defaults to `dba`, because this is recommended by the Oracle installation guide. It is the group given to the Oracle user when the user is being generated. On Windows NT systems, Oracle does not use the group concept and the Oracle *internal* user password should go in here. The internal users are `sys`, `sysadmin` ,or `admin`. Problems typically show up at registration time. If you type in the right parameters and problems still occur, refer to Section B.6.4, “ESM Database Registration” on page 664.

At this point you can create a profile manager inside the Database Management Region. You can have four different types of profiles:

- Resource profile
- Role profile
- User profile
- Sentry profile

Each has its own icon representation in a profile manager.

The resource, role, and user profile types come with the ESM product, and the Sentry profiles created use the TME 10 Distributed Monitoring Sentry profile. When you first look at the resource, role, and user profiles, there is nothing useful in there, but you can populate them by double-clicking the profile and choosing **Populate...** from the Profile pull-down menu. To get all the information already defined, the Validation Policies have to be disabled by double clicking the profile and choosing **Edit** from the pull-down menu and then disable the **Validation Policies**.

After this has been done, you can choose the **Populate...** menu item to get all information including Oracle administrator information. After that, it is recommended that you turn back the Validation Policies to **enabled** to avoid overwriting any required administrator information.

To implement some useful TME 10 Distributed Monitoring monitors, we used a script file, which allows the creation of some Sentry profiles in a given profile manager.

```
#!/bin/sh
#
# Script to create sample monitoring profiles
#
# sample_monitors.sh
# Provided by DBMX for demonstration use only
# - Does not perform any error checking
# Creates five profiles with sample monitors. Requires the Profile Manager
# as an argument.
#
# When this script has completed, set the User and Group ID for the profiles, and
# distribute to Database and Instance endpoints as appropriate
if [ $# -ne 1 ]
then
echo "Format is: sample_monitors.sh ProfileManager"
exit 1
fi
PRFMAN=$1
# Create the profiles
wcrtprf @ProfileManager:$PRFMAN SentryProfile Database_Alarms
wcrtprf @ProfileManager:$PRFMAN SentryProfile Database_Warnings
wcrtprf @ProfileManager:$PRFMAN SentryProfile Oracle_Caches
wcrtprf @ProfileManager:$PRFMAN SentryProfile Instance_Warnings
wcrtprf @ProfileManager:$PRFMAN SentryProfile Capacity_Planning
#Add the monitors for profile Database_Alarms
waddmon OracleDatabase archivespacerl -a no -t "30 minutes" \
    -c critical -R "<"  3 -n "Oracle Sentry" \
    -c severe   -R "<"  5 -n "Oracle Sentry" \
    -c warning  -R "<" 10 -n "Oracle Sentry" \
    Database_Alarms
waddmon OracleDatabase nettwolistener -a no -a "" -t "5 minutes" \
    -c critical -R "==" "down" -n "Oracle Sentry" \
    Database_Alarms
waddmon OracleInstance rdbmsstate -t "5 minutes" \
    -c critical -R "==" "failed"  -n "Oracle Sentry" \
    Database_Alarms
# Add the monitors for profile Database_Warnings
waddmon OracleDatabase alerts -a no -t "10 minutes" \
    -c critical -R "==" 1 -n "Oracle Sentry" \
    -c severe   -R "==" 2 -n "Oracle Sentry" \
    -c warning  -R ">"  2 -n "Oracle Sentry" \
    Database_Warnings
waddmon OracleDatabase backgrounddumpspace -a no -t "30 minutes" \
    -c critical -R ">" 95 -n "Oracle Sentry" \
    -c severe   -R ">" 90 -n "Oracle Sentry" \
    -c warning  -R ">" 85 -n "Oracle Sentry" \
    Database_Warnings
waddmon OracleDatabase freespacedef -a no -a no -t "1 hour" \
    -c critical -R "<"  0 -n "Oracle Sentry" \
    Database_Warnings
```

Figure 239. Script to Create ESM Sentry Profiles, Page 1

```
waddmon OracleDatabase freespacefragmentation -a no -t "1 day" \
    -c critical -R "<" 10 -n "Oracle Sentry" \
    -c severe   -R "<" 20 -n "Oracle Sentry" \
    -c warning  -R "<" 30 -n "Oracle Sentry" \
    Database_Warnings
waddmon OracleDatabase freetablespace -a no -a no -t "1 hour" \
    -c critical -R "<"  5 -n "Oracle Sentry" \
    -c severe   -R "<" 10 -n "Oracle Sentry" \
    -c warning  -R "<" 20 -n "Oracle Sentry" \
    Database_Warnings
waddmon OracleDatabase maximumextents -a no -a no -t "1 days" \
    -c critical -R "<"  1 -n "Oracle Sentry" \
    -c severe   -R "<"  3 -n "Oracle Sentry" \
    -c warning  -R "<"  5 -n "Oracle Sentry" \
    Database_Warnings
waddmon OracleDatabase rowsindual -a no -t "1 hour" \
    -c critical -R "!=" 1 -n "Oracle Sentry" \
    Database_Warnings
# Add the monitors for profile Oracle_Caches
waddmon OracleInstance bufcachehitratioi -a no -t "15 minutes" \
    -c critical -R "<" 70 -n "Oracle Sentry" \
    -c severe   -R "<" 80 -n "Oracle Sentry" \
    -c warning  -R "<" 90 -n "Oracle Sentry" \
    Oracle_Caches
waddmon OracleInstance dictcachehitratioi -a no -t "15 minutes" \
    -c critical -R "<" 70 -n "Oracle Sentry" \
    -c severe   -R "<" 80 -n "Oracle Sentry" \
    -c warning  -R "<" 90 -n "Oracle Sentry" \
    Oracle_Caches
waddmon OracleInstance libcachehitratioi -a no -t "15 minutes" \
    -c critical -R "<" 90 -n "Oracle Sentry" \
    -c severe   -R "<" 92 -n "Oracle Sentry" \
    -c warning  -R "<" 95 -n "Oracle Sentry" \
    Oracle_Caches
# Add the monitors for profile Instance_Warnings
waddmon OracleInstance continuedrowratio -a no -t "2 hours" \
    -c critical -R ">" 1   -n "Oracle Sentry" \
    -c severe   -R ">" 0.5 -n "Oracle Sentry" \
    -c warning  -R ">" 0   -n "Oracle Sentry" \
    Instance_Warnings
waddmon OracleInstance enqueuetimeouts -a no -t "2 hours" \
    -c critical -R "- >=" 3 -n "Oracle Sentry" \
    -c severe   -R "- >=" 2 -n "Oracle Sentry" \
    -c warning  -R "- >=" 1 -n "Oracle Sentry" \
    Instance_Warnings
waddmon OracleInstance freelistwaits -a no -t "2 hours" \
    -c critical -R ">" 2   -n "Oracle Sentry" \
    -c severe   -R ">" 1   -n "Oracle Sentry" \
    -c warning  -R ">" 0.5 -n "Oracle Sentry" \
    Instance_Warnings
-c critical -R ">" 2   -n "Oracle Sentry" \
    -c severe   -R ">" 1   -n "Oracle Sentry" \
    -c warning  -R ">" 0.5 -n "Oracle Sentry" \
    Instance_Warnings
waddmon OracleInstance readsysstatd -a no -t "2 hours" \
    -c critical -R "- >=" 5 -n "Oracle Sentry" \
    -c severe   -R "- >=" 3 -n "Oracle Sentry" \
    -c warning  -R "- >=" 1 -n "Oracle Sentry" \
    Instance_Warnings
waddmon OracleInstance processratio -a no -t "2 hours" \
    -c critical -R ">" 95 -n "Oracle Sentry" \
    -c severe   -R ">" 90 -n "Oracle Sentry" \
    -c warning  -R ">" 85 -n "Oracle Sentry" \
    Instance_Warnings
waddmon OracleInstance redologwaits -a no -t "10 minutes" \
    -c critical -R "- >=" 5 -n "Oracle Sentry" \
    -c severe   -R "- >=" 3 -n "Oracle Sentry" \
    -c warning  -R "- >=" 1 -n "Oracle Sentry" \
    Instance_Warnings
```

Figure 240. Script to Create ESM Profiles, Page 2

```
waddmon OracleInstance rollbackwaits -a no -t "1 hour" \
    -c critical -R ">" 5 -n "Oracle Sentry" \
    -c severe   -R ">" 3 -n "Oracle Sentry" \
    -c warning  -R ">" 2 -n "Oracle Sentry" \
    Instance_Warnings
# Add the monitors for profile Capacity_Planning
# Note that these monitors store information in the ESM$MONITOR table
waddmon OracleInstance bufcachehitratio -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance callrate -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance dictcachehitratio -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance libcachehitratio -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance physicalreads -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance physicalwrites -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
waddmon OracleInstance sortoverflowratio  -a yes -t "1 hour" \
    -c always -n "Oracle Sentry" \
    Capacity_Planning
exit
```

Figure 241. Script to Create ESM Profiles, Page 3

The above script will generate the following profiles:

- Capacity_Planning
- Database_Alarms
- Database_Warnings
- Instance_Warnings
- Oracle_Caches

Figure 242 shows an example of what is added in the SentryProfiles by running the above script file:

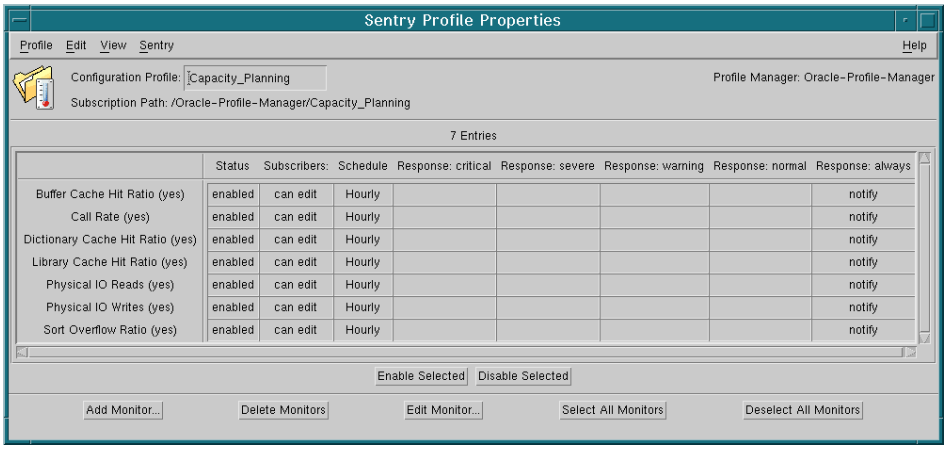


Figure 242. ESM Capacity_Planning Monitors

The script file to generate these monitors applies to UNIX and NT TMR servers. On NT, you have to run the script file in the bash environment. The previous examples should help you to create similar monitors for the other database modules.

B.2 Directories for ESM Database Management Files

The following is a list of extra files provided that you may need to use along with their locations. The files listed below might be necessary for problem determination purposes, executing from the command line, or for adding something to an existing rule base. They are all contained within `$BINDIR/bin`:

- `M7MSSQLServerSentry.pl`
- `M7OracleSentry1.pl`
- `M7OracleSentry2.pl`
- `M7OracleSentry3.pl`
- `M7OracleSentry4.pl`
- `M7OracleSentry5.pl`
- `M7SQLEnginev70`
- `M7SQLEnginev71`
- `M7SQLEnginev72`
- `M7SQLEnginev73`
- `M7SybaseSentry1.pl`
- `M7SybaseSentry2.pl`
- `M7SybaseSentry3.pl`
- `M7SybaseSentry4.pl`
- `mcsl`

- mextract
- sregmonsvr
- sregsvr
- ostartup
- oshutdown

B.3 Adding ESM Tasks

ESM for Oracle provides the following task-related files:

- /usr/local/Tivoli/bin/generic/OracleFrameworkTasks

  OracleFrameworkTasks.tlf

- /usr/local/Tivoli/bin/generic/OracleSentryTasks

  OracleSentryTasks.tlf

The second line in the file OracleFrameworkTasks.tlf has the distribution mode for the tasks:

```
Distribute = "ALI";
```

This means that when the task library is loaded using the wtll command, the task executables are not copied to each managed node.

To run a task as a response to a monitor, the distribution mode needs to be LOCAL; so, this file will need editing, and the task library will need to be reloaded.There are no such tasks for the other databases. These are the Task File Libraries, which are necessary to create the task library on the desktop within a given policy region. In the current release, the task libraries are not generated during installation but by running the following commands:

```
wtll -p Database-Management -P cat OracleFrameworkTasks.tlf

wtll -p Database-Management -P cat OracleSentryTasks.tlf
```

The -p (lower-case) parameter indicates the policy region where the task libraries are going to be created, and the -P (upper-case) defines a preprocessor that needs to be run before creating the libraries. Both parameters are mandatory; otherwise, you will get a failure.

If you specify the label for the policy region incorrectly you will get an error:

```
Syntax errors:
tll error in Management, line 1: Cannot run preprocessor "cpp"
Improper task library statement (missing "TaskLibrary" keyword; saw Cannot instead)
```

Figure 243. Possible Error Message when Creating ESM Task Library

The same error message occurs when missing one of the mandatory parameters, which can be misleading.

Refer to Section 7.4, “Task Library” on page 238 for more information about task libraries and adding tasks.

B.3.1 ChangeOracleHome Task

This task changes the ORACLE_HOME value stored on the Database Endpoint Object. It does not effect the location of the physical database; it changes where ESM expects that location to be. As such, this task should only be used to effectively notify the ESM subsystem that the ORACLE_HOME location has been successfully changed for the target database. This should only be run once the database has been moved. The task updates the object with the following `idlcall`:

```
idlcall $EndpointOid _set_Home "$NewOracleHome"
```

To check the value of the attribute, you can do the following:

```
idlcall $EndpointOid _get_Home
```

Where `$EndPointOid` is the OID of the managed node containing the database. The task also updates the VersionNumber attribute if the Oracle software has been upgraded.

B.3.2 DiscoverOracleDB Task

This task attempts to auto-discover and register Oracle databases on the target managed node. For all UNIX implementations, the `oratab` file is scanned. To allow for possible connection failures and down databases during registration, the process is NOT deemed as a recoverable transaction. This means that if one database registration fails, the discovery process reports the failure and continues trying to register other databases.

The files it reads on UNIX are `/var/opt/oracle/oratab` and `/etc/oratab`. Which one is used depends on the UNIX type; so, it looks at both to make sure it finds the entries.

Note

This task is not supported on Windows NT, as the user would have to supply an internal password for each database.

For each database it finds, `oregdb` is used to register the database.

B.4 TME 10 Enterprise Console Operations

ESM provides a number of collections of files that help integrate the product with Distributed Monitoring and TEC as follows:

- `/usr/local/Tivoli/bin/generic/OracleSentry`:

  ```
  ESMSentry.baroc
  M7OracleDatabase.baroc
  M7OracleDatabase.col
  M7OracleDatabaseDefaults.sh
  M7OracleDatabaseDefaultsDrop.sh
  M7OracleInstance.baroc
  M7OracleInstance.col
  M7OracleInstanceDefaults.sh
  M7OracleInstanceDefaultsDrop.sh
  ```

- `/usr/local/Tivoli/bin/generic/MSSQLSentry`:

  ```
  ESMSentry.baroc
  M7DropMSSQLDatabaseDefaults.sh
  M7DropMSSQLServerDefaults.sh
  M7MSSQLDatabaseDefaults.sh
  M7Profile pull down menu.MSSQLDatabaseSentry.baroc
  M7MSSQLDatabaseSentry.col
  M7MSSQLServerDefaults.sh
  M7MSSQLServerSentry.baroc
  M7MSSQLServerSentry.col
  ```

- `/usr/local/Tivoli/bin/generic/SybaseSentry`:

  ```
  ESMSentry.baroc
  M7SybaseDatabase.baroc
  M7SybaseDatabase.col
  M7SybaseDatabaseDefaults.sh
  M7SybaseDatabaseDefaultsDrop.sh
  M7SybaseServer.baroc
  M7SybaseServer.col
  M7SybaseServerDefaults.sh
  M7SybaseServerDefaultsDrop.sh
  SetErrorCategory.sh
  ```

To get database management events into TEC, a new rule base is necessary with the imported Distributed Monitoring and ESM .baroc files as illustrated in figure:

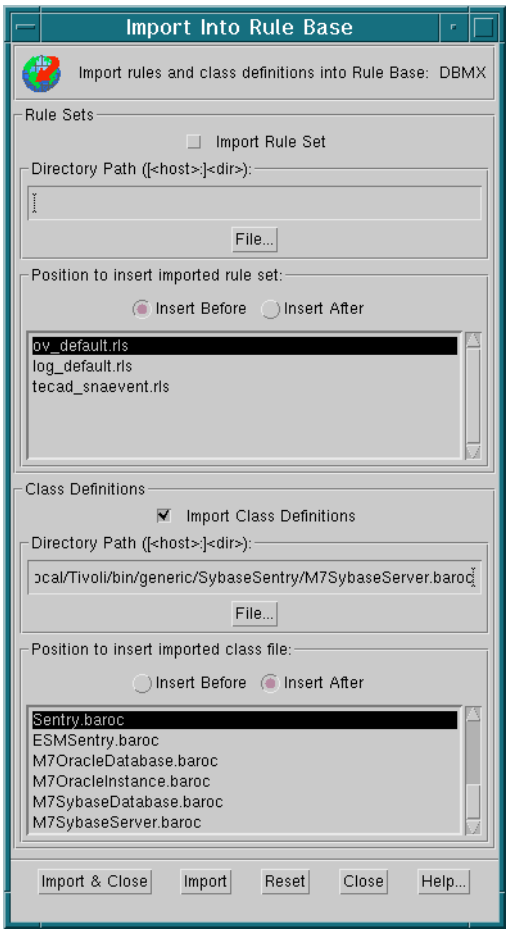


Figure 244. Import of Oracle and Sybase .baroc Files into a New Rule Base

The new .baroc files should be added at the end of the existing .baroc files of the actual rule base. You have to have the right sequence when adding these .baroc files. For the database management the following hierarchy applies:

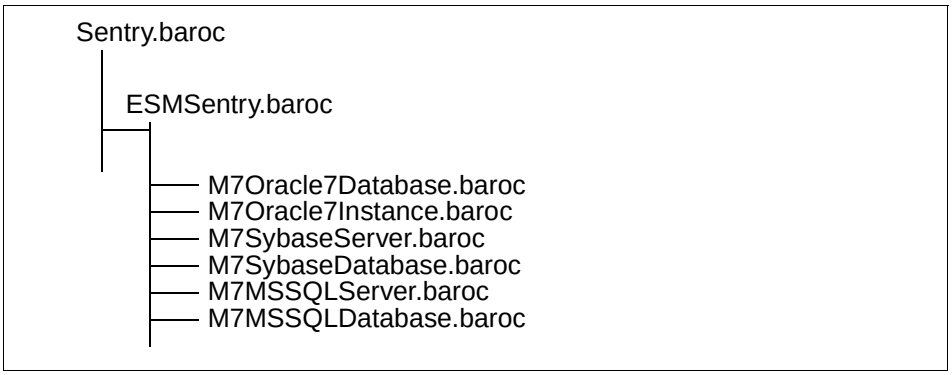


Figure 245. TEC .baroc File Hierarchy for Database Management

B.5 ESM Frequently Asked Questions

The following sections relate some of the more important questions and answers about ESM database management.

B.5.1 Oracle Framework

Q:***Can I register a database that isn't running?***

A: No, but this is planned to be an option in a future release of ESM Framework.

Q:***Do I have to register all my databases through the GUI?***

A: No, you can use the **DiscoverOracleDB** task to register databases on target Managed Node. This does not currently work on Windows NT, as the Oracle INTERNAL account requires a password that ESM cannot discover.

Q: ***Can I register a database in an interconnected TMR?***

A: Yes, as long as the database and policy region are in the same TMR.

Q: ***What happens if I upgrade Oracle, and ORACLE_HOME changes?***

A: Run the **ChangeOracleHome** task to change the ORACLE_HOME attribute stored on the Oracle7 object.

Q: ***Can I add a datafile or rollback segment to more than one database at a time?***

A: No, all of the actions in the Framework software apply to an individual endpoint. You could write a task to apply the actions to multiple databases.

Q: ***When editing a parameter, does ESM save the old version of the parameter file?***

A: Yes, the `init<sid>.ora` file is saved as `init<sid>.ora.old`. A comment is added to the `init<sid>.ora` file to stay what the ID value of the parameter was and when the file was changed.

Q: ***Are there any plans to have a SQL*Net listener endpoint?***

A: This may be provided in a future release of ESM.

B.5.2 Oracle7 Distributed Monitoring

Q: ***What are the most-likely causes of a monitor failing to run?***

A: The two most common causes of problems are: The user ID and group ID have not been set for the profile. The administrator does not have the oracle_monitor role.

Q: ***Why do I need to set user-ID and group-ID to get a monitor to run?***

A: Because ESM methods run under the control of a particular administrator.

Q: ***Why is the oracle_dba role needed to run a monitor?***

A: This is needed because StoreHistory and the Chained Rows monitors use the ExecuteSQL method, which requires the `oracle_dba` role.

Q: ***If I have a mix of UNIX and NT machines, how can I set the user and group ID for a profile and distribute to both machine types?***

A: You cannot do this. Different profiles have to be created for UNIX and NT subscribers.

Q: ***Is there a TEC Adapter for ESM?***

A: No, ESM uses the TME 10 Distributed Monitoring mechanism to forward events to the TEC.

Q: ***Should I reload ESM baroc files when I upgrade?***

A: Yes, as there may be changes to the classes or new monitors may be added.

Q: ***How does SQL*Net V1 TCPI/IP listener monitor work?***

A: By running the command `tcpctl stat` and parsing the output to see that the listener is running. A timeout is set on the command to see if the listener has hung.

Q: ***How does the SQL*Net V2 listener monitor work?***

A: By running the command `lsnrctl stat` and parsing the output to see that the listener is running. A timeout is set on the command to see if the listener has hung.

Q: ***Should I distribute the listener monitors to each endpoint?***

A: You can, but you only need to distribute these monitors to a single endpoint.

Q: ***How can I turn off monitors when the database is shut down?***

A: Use the DisableMonitor task in the ESM OracleSentryTasks task library.

Q: ***Can I distribute a profile containing instance monitors to a database endpoint, and conversely?***

A: Yes, but you shouldn't, and the ability to do this may be removed in a future release.

Q: ***Why do I get an exception when running chained rows monitors?***

A: Probably because the `utlchain.sql` script has not been run as `SYS`. This creates a table called `CHAINED_ROWS` that stores the output from the `ANALYZE TABLE LIST CHAINED ROWS` command.

Q: ***Does the ESM$MONITOR table have to be owned by the user DBMX?***

A: Yes, as ESM runs the command:

```
INSERT INTO DBMX.ESM$MONITOR VALUES(...);
```

Q: ***Why does my Run Program response to a monitor not work?***

A: When using the `Run Program` response, there are a number of attributes of the file to be run that may cause things not to work as desired. When specifying a file to run, provide the full path name to the file. What you select for the path name will depend on which type of managed node to which you will be distributing the profile. Another consideration is the permission of the file. On a UNIX system, this can be checked through standard CLI commands, while on Windows NT, you can use the File Manager or other CLI interfaces. The permission you choose for the file will depend on who is going to be running the response, that is, the user-ID/group-ID of the profile, which is set through the Edit menu in the Sentry GUI.

Q: ***Why is the Profile_OID incorrect in the history table?***

A: Probably because the Sentry 3.0.1 patch has not been installed. In previous releases, the Profile_OID environment variable did not exist, and ESM inserted a dummy value into the table.

Q: ***Can you advise on setting up monitoring profiles?***

A: It is difficult to give general advice since every DBA has their own monitors and SQL scripts that they want to run. The default thresholds and monitoring schedule for each ESM monitor should be adequate for most situations. If ESM does not have a monitor that a customer wants, it is possible to create a monitor using the Freeform SQL monitors.

Q: ***Can I add my own list of alerts to the Alerts monitor?***

A: Not with this release. In a feature release of ESM, you will be able to specify which alerts you want included and excluded from the list of alerts that ESM checks.

Q: ***How does the free space fragmentation work?***

A: The information below is taken from the *Oracle DBA Handbook*. To judge whether a table space could benefit from a free space rebuild, it is necessary to establish a baseline on an arbitrary scoring system. Since free space fragmentation is made up of several components (number of extents, size of largest extent), the scoring index considers both. The weight used is arbitrary and is chosen to reflect the potential for the database to acquire a large extent. Thus, the number of extents is given little importance. The critical factor is the size of the largest extent as a percentage of the total free space. This is referred to as the Free Space Fragmentation Index (FSFI), and is calculated as follows:

$$FSFI = 100 * \sqrt{\left(\frac{\text{largest extent}}{\text{sum all extents}}\right)} * \frac{1}{(\text{number of extents})^{1/4}}$$

Figure 246. Equation to Calculate the Free Space Fragmentation Index

The largest possible FSFI is 100. As the number of extents increases, the FSFI rating drops rapidly. You should rarely encounter free space availability problems in table spaces that had adequate free space available and FSFI ratings over 30.

Q: ***How does RDBMS state monitor know if a database has crashed or has been shutdown by a database administrator?***

A: By looking at the end of the alert log for a shutdown message. If Oracle is shutdown by a database administrator, a message will be written to the alert log, but if a database crashes, no message will be written.

Q: ***Do any of the monitors modify a database?***

A: If the StoreHistory option is set to Yes for a monitor, a row will be written DBMX.ESM$MONITOR every time the monitor fires. The Chained rows (cluster)

and Chained rows (table) monitors run the ANALYZE command and will update the SYS.CHAINED_ROWS table.

B.5.3 Oracle User Management

Q: ***How does ESM populate the Set Tablespace dialog ?***

A: This is initially populated with SYSTEM, and the following four table spaces exist if a database is created by Oracle on a UNIX system when the Oracle software is installed: RBS, TEMP, TOOLS and USER. When a ESM User Profile is populated, all the table spaces in the endpoint database are added to the tablespaces list. A user can manually add table spaces on the list through the Set Tablespace dialog.

Q: ***Is the Tablespace pick list available on a global basis or a per-profile basis?***

A: On a per-profile basis.

Q: ***Can I set policy to stop showing invalid subscribers to ESM User Administration Profiles in the Subscribers dialog?***

A: Yes, using the following steps:

1. `wcrtpol -d ProfileManager Subs`
2. `wgetpolm -d ProfileManager Subs pm_def_subscribers > subs.tst`
3. `vi subs.tst`

   Now change:

   `wgetallinst -l ProfileEndpoint`

   To the following:
   `wgetallinst -l OracleDatabase`
4. `wputpolm -d ProfileManager Subs pm_def_subscribers < sub.tst`
5. `Set Default Managed Resource Policy for ProfileManager to Subs`

B.6 Troubleshooting the ESM Framework

This section contains information on dealing with problems that can arise with the ESM Framework feature. Refer to Chapter 3, “The Tivoli Core Installation Process” on page 59 for general advice regarding the installation of products.

B.6.1 Troubleshooting ESM TMR Server Installs

Check the output files and error logs listed in table 43:

Table 43. Output Files and Error Logs for ESM Oracle Server Installation on UNIX

Files (UNIX)	Location
/tmp/tivoli.sinstall	server
/var/spool/Tivoli/nodename.db/oservlog	server
/tmp/install.cfg.error and /tmp/install.cfg.output	server

On NT, the files will be in the directory indicated by the system variable TMP, which by default is %DBDIR%\tmp.

B.6.2 Reinstalling Failed Server Installations

If the product is only partially installed, you may need to remove the marker files before reinstalling the product. The following list shows the marker files on UNIX and NT:

- In the /usr/local/Tivoli/bin/aix4-r1/.installed directory on UNIX or in the tivoli\bin\w32-ix86\.installed directory on NT:
  - OracleFramework_BIN
  - OracleSentry_BIN
  - OracleUser_BIN
- In the /usr/local/Tivoli/lib/aix4-r1/.installed directory on UNIX or in the tivoli\lib\w32-ix86\.installed directory on NT:
  - OracleFramework_LIB
  - OracleUser_LIB
- In the /usr/local/Tivoli/bin/generic/.installed directory on UNIX in the tivoli\bin\generic\.installed directory on NT:
  - OracleFramework_GBIN
  - OracleSentry_GBIN
- In the /var/spooll/Tivoli/db/nodename.db/.installed directory on UNIX or in the tivoli\db\nodename.db\.installed directory on NT:
  - OracleFramework_ALIDB
  - OracleUser_ALIDB

- In the `/usr/local/Tivoli/man/aix4-r1/.installed` directory on UNIX (no entry on NT):
  - OracleFramework_MAN
  - OracleSentry_MAN
  - OracleUser_MAN
- In the `/usr/local/Tivoli/msg_cat/.installed` directory on UNIX or in the `tivoli\msg_cat\.installed` directory on NT:
  - OracleFramework_CAT
  - OracleUser_CAT

B.6.3 Troubleshooting ESM Managed Node Installs

Check the following output files and error logs, as shown in Table 44:

Table 44. Output Files and Error Logs for ESM Oracle Client Installation on UNIX

Files (UNIX)	Location
/tmp/tivoli.cinstall	Server
/var/spool/Tivoli/nodename.db/oservlog	Server and client
/tmp/install2.cfg.error and /tmp/install2.cfg.output	Client
/tmp/client.cfg.error and /tmp/client.cfg.output	Client

On NT the files will be in the directory indicated by the system variable *TMP*, which, by default, is `%DBDIR%\tmp`.

B.6.4 ESM Database Registration

This is more likely to encounter problems than anything else. Check that the Oracle database is up and running before trying to register the database and that the administrator has the *oracle_dba* and *senior* roles. OracleDatabase must be a managed resource in the policy region used for database management. Check that the information in the dialog is correct, that is, no spaces or control characters.

NIS (Yellow Pages) can mess up user name and group, particularly in IBMs AIX. You may need to create an entry in the `/etc/group` file for the `dba` group on the local machine.

For Windows NT, enter the database internal user-id password in the group field.

Use `oregdb` to do a command line registration. The following shows the usage and an example of how to do so:

```
oregdb Hostname ORACLE_SID ORACLEHOME Owner OwnerGROUP Policy_Region
```

In our environment this would be:

```
oregdb rh0255f D1 /oracle/home/app/oracle/product/7.3.2 oracle
 dba Database-Management
```

The registration process connects to the database to ensure it is running, and this can be where problems are indicated.

B.6.4.1 Registration over an Interconnected TMR

The database and policy region must be in the same TMR; otherwise, registration will fail.

If you have two TMRs, the local one (called local, for example) and a remote one (called remote, for example), the remote TMR will have a policy region, called remote-region that contains databases that have been registered on the remote TMR.

Suppose we make a two-way inter-TMR link initiated from our local TMR.

You may wish to verify updates have taken place or ensure they have by manually running the update process. Go on to each TMR and do an update of all resources from the GUI (or `wupdate -r All` on the command line) on both the local and remote TMRs. Then you have to update the resources to which your local administrator has access, as described in the following list:

1. Select **TMR Connections -> Top Level Policy Regions** from your local desktop menu to display the Top Level Policy Regions window. One of the region is the remote-region.
2. Double-click on the **Administrator** icon in your local-region dialog to open your local Administrators dialog.
3. Drag and drop the **remote-region** icon in the Top Level Policy Regions dialog over and onto the local administrators icon in the local Administrators dialog. Be careful to drop it on to your **local Administrators** icon because the remote Administrators icon will also be visible.

   After a few seconds, the remote-region policy region icon will appear in your local desktop dialog.

Therefore, from your local desktop, you will have both local-region and remote-region policy regions accessible through the icons. However, you

cannot copy or drag and drop the database icons from one region to the other.

A profile manager created in the local policy region will be able to subscribe to databases from both local-region and remote-region

B.6.5 Removing a Database Object

Don't just delete the object in the GUI, you must unsubscribe first. Make sure you don't leave monitors running. Check this with the `wlseng` command.

You may need to do this if registration fails. It may fail after it has created an object in CCMS. Find the OID and use `idlcall` to remove the objects. This needs to be done for the database object and the instance object:

```
wlookup -ar OracleDatabase
wlookup -ar OracleInstance
idlcall OID remove
```

You should be familiar with Chapter 2, "Tivoli Object Database Architecture" on page 9 before manipulating objects with IDL calls and be sure to take note of any warnings given in that chapter.

B.6.6 ESM Roles

This section summarizes the TMR roles provided by ESM:

- `oracle_dba`

  Can do anything to Oracle Databases.

- `oracle_admin`

  Distribute monitoring profiles and run TME 10 Distributed Monitoring monitors.

- `oracle_operator and oracle_user`

  These are currently equivalent. They can perform all read-only operations but no updates. Can call `SelectSQL` on Instance but not Database.

- `oracle_monitor`

  Very restricted.

  - Can call `SelectSQL` on instance but not database.
  - Can call `StartSentryChannel` for `omonsql` based monitoring.
  - Can call `_set_state` on database to flip the icons.

B.6.7 ESM Notice Group

At least one administrator should be assigned the notice group *Oracle Database Management,* but don't rely on this alone. You will need to review what other notice groups the administrator should have assigned based on the use of tasks, monitors, and so on.

B.6.8 Database Operations

There are two commands provided by ESM located in `/$BINDIR/bin`, which might be helpful doing database operations:

- `ostartup`
- `oshutdown`

`ostartup` This command enables you to start a database from the command line. This could be used as a Run Program response to the RDBMS state monitor if a database crashes.

ostartup Command Syntax

```
# ostartup
usage: ostartup [ -p ] [ -r ] [ -f ] [ -n | -m | -o ] [-I] database-name
        -p = parallel
        -r = restricted
        -f = force (abort)
        -n = nomount
        -m = mount
        -o = open (default)
        -I = to specify that database-name refers to an OracleInstance
```

Figure 247. ESM Command Line Database Startup

`oshutdown` This command enables you to start a database from the command line. This could be used as a Run Program response to other RDBMS monitors to initiate an automated shutdown of the database if needed.

oshutdown Command Syntax

```
# oshutdown
usage: oshutdown [ -n | -i | -a ] [-I] database-name
        -n = normal
        -i = immediate (default)
        -a = abort
        -I = to specify that database-name refers to an OracleInstance
```

Figure 248. ESM Command Line Database Shutdown

B.6.9 Symbolic Links

Note that Oracle does not support ORACLE_HOME as a symbolic link, and this is the same with ESM. You may get strange error messages starting up a database from within ESM if ORACLE_HOME is a symbolic link. The message will indicate that another instance already has the database mounted.

B.6.10 Background Daemons

Table 45 gives an overview of the daemons used by ESM:

Table 45. ESM Distributed Monitoring Background Daemons

Daemon	Object Type	Description
M7SQLEngineRun	M7SQLEngine	Establishes and maintains a connection to the Oracle database. Executes SQL and returns results.
M7DatabaseRunBase	Oracle7Database	Provides all non-GUI database functionality.
M7DatabaseRunGUI	Oracle7Database	Provides GUI for database objects.
M7InstanceRunBase	Oracle7Instance	Provides all non-GUI profile operations for Oracle instances.
M7InstanceRunGUI	Oracle7Instance	Provides GUI for instance objects.
M7InstanceRunSQL	Oracle7Instance	Controls the M7SQLEngine daemons for connection to the database.

B.7 Troubleshooting ESM Distributed Monitoring

This section contains information on dealing with problems that can arise with the ESM Distributed Monitoring usage.

B.7.1 ESM Distributed Monitoring Installation

If the product is only partially installed, you may need to remove the ESM marker files before reinstalling the product. For more information about the marker file locations refer to Section B.6.1, “Troubleshooting ESM TMR Server Installs” on page 663.

If the names of any monitors have changed, drop and re-create the profiles.

B.7.2 ESM Distributed Monitoring Notice Groups

Oracle Sentry can be used as the output from a monitor.

SentryStatus shows errors with monitors. Ensure that at least one administrator has this notice group subscribed.

Figure 249 is an example of an entry posted to the Oracle Sentry notice group in the case where **Store History** is chosen, but the table doesn’t exist:

```
Monitoring information:
   Sentry Capacity_Planning/Sort Overflow Ratio on host v722@toby
Mon Apr 07 13:58:00 1997
   Status: >>> E.EXEC <<<
   Sort Overflow Ratio (yes) (omonsql Returned Error : {
USER_EXCEPTION ExSqlError {
"Exception:UserException:SysAdminException::ExException:ExM7
SqlException:ExSqlError" "M7ExceptionCat" 1 "%5$t{%c} (%3$d):
SQL Error     : ‘%7$s’) Exit code 29
   (Previous:  Current:  Effective: )
```

Figure 249. Output Example for Oracle Sentry Notice Group

To see the monitor defaults in command line, issue the `wlsmon` command.

B.7.3 User and Group ID with Insufficient Access

This is a common problem with Distributed Monitoring. Profiles are created with a UID and GID of `nobody`. This must be changed to a valid UID and GID with appropriate access.

The `Set User/Group` ID controls the `user/group` ID under which the monitor probe runs. The idea is that the processes that manage monitoring only allow probes to be executed (with their inherent overhead) for people who are authorized to do so.

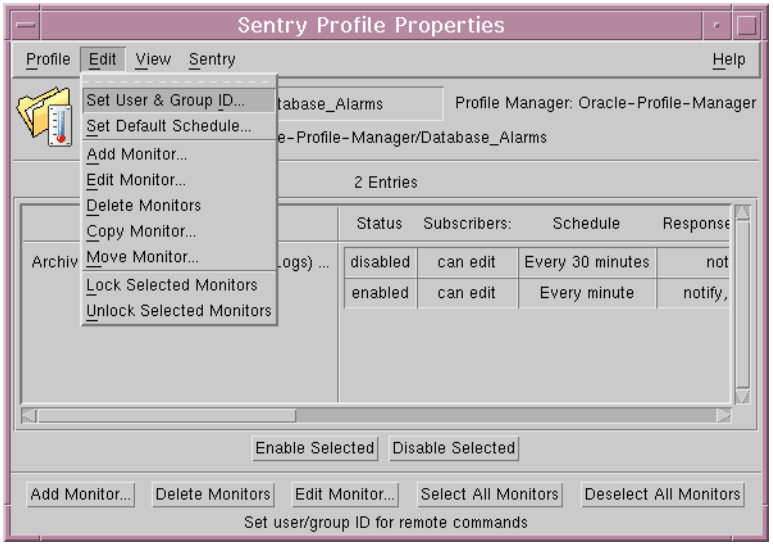


Figure 250. Panel to Set UID and GID in Profiles

Sentry only carries out authorization on a profile the first time it is pushed to that node or when the Sentry engine starts up. Therefore, if you change the user ID in the profile after you have already pushed it, you must stop the Sentry engine before re-pushing the profile with the new user ID.

The ability to change the `Set User/Group` ID requires the Tivoli admin role. This makes it possible to pre-configure the profile from an administrator desktop and have it used and pushed by someone without that role.

A good way to do this (one of many) is to create a designated OS user account on the target machine(s), such as esm_monitor, and treat it like the nobody account as far as the OS is concerned. Then create a TME 10 administrator desktop with at least resources, policy regions, and so on, but include the oracle_monitor role and then add the unqualified esm_monitor login (that is, not `esm_monitor@host`) to its login list.

B.7.4 Removing Monitors

Unsubscribe - Remove all profile copies or the monitors keep running.

Clear out the monitors using the `wclreng` command.

B.7.5 Required Roles

The oracle_monitor role or user role is needed to run monitors. Oracle_admin is needed to distribute a monitoring profile and to run a monitor.

B.7.6 Database and Instance Collection

Push profiles out of this collection to the database subscriber only. This is not enforced in the current release. Don't push profiles to a managed node subscriber. If you do, you will get an EXEC error with ORACLE_SID not defined. Software will be changed in future to prevent this. The same rule applies to profiles that are part of the instance collection. Push these profiles to a instance subscriber only.

B.7.7 Monitoring Tasks

This section summarizes the tasks provided.

- **CreateHistoryTable**

  This task creates the DBMX user and the ESM$MONITOR table used by ESM for storing historical monitoring results, thereafter, used by ESMChart. The task works as follows:

  - If the user DBMX and the table ESM$MONITOR exist, exit.
  - If the user DBMX doesn't exist, create the user using the parameters supplied on the GUI.
  - Create user DBMX identified by `&passworddefault` tablespace `&dtsnametemporary` tablespace `&ttsname`.
  - If the table `ESM$MONITOR` doesn't exist, create it.
  - Create table `dbmx.ESM$MONITOR` (monitor_id number, profile_oid varchar2(20), monitor_date date default SYSDATE, monitor_value number, monitor_title varchar2(100)).
  - If CONNECT privilege is granted to the user, grant connect and resource to `dbmx` or grant resource to `dbmx`.

- **PurgeHistoryTable**

  This task clears the `ESM$MONITOR` table, used by ESM for storing historical monitoring results, thereafter, used by ESMChart.

  - If all records are to be purged, the task drops and re-creates the table.
  - Drop table `dbmx.esm$monitor`.
  - Create table `dbmx.ESM$MONITOR` (monitor_id number, profile_oid varchar2(20), monitor_date date default SYSDATE, monitor_value number, monitor_title varchar2(100)).
  - Otherwise, delete all records prior to the date given on the GUI.
  - Delete from `dbmx.esm$monitor` where monitor_date < &date.

- **CurrentRunningSQL**

  This task shows the currently-running SQL for a specified user or for all users. The SQL used to display the SQL for all users is:

  ```
  select v$session.username, v$sqltext.piece, v$sqltext.sql_text
  from sys.v$session v$session, sys.v$sqltext v$sqltext where
  v$sqltext.address = v$session.sql_addressand v$sqltext.hash_value
  = v$session.sql_hash_valueorder by v$session.username,
  v$sqltext.piece;
  ```

- **DisableMonitoring:**

  This task disables all the currently-running monitors on the target database(s) and optionally shuts down the database(s).

  It uses `wdisprb` to disable the monitors, then calls `oshutdown` to shutdown the database in the requested mode.

- **EnableMonitoring:**

  This task enables all the currently-running monitors on the target database(s) and optionally starts up the database(s).

  It uses `ostartup` to start the database, then calls `wenlbprb` to enable the monitors.

B.7.8 Further Problem Determination at the Endpoint

The following a suggested sequence of steps to try. Refer also to Chapter 12, “Distributed Monitoring” on page 387.

- Check what's in the Sentry engine. This will cause an error if the engine isn’t running. Use `wlseng` to run the engine.
- Check the user and group ID of the Sentry profile.
- TME Administrator needs oracle_monitor role.
- OS user specified in Set User & Group ID needs to be in an Edit Logins list for a TME Administrator that has the oracle_monitor role.
- Check the SentryStatus notice group; it has the execution status and any problems with response actions will generate a message here.

B.8 Troubleshooting ESM Oracle User Managment.

This section deals with the management of users.

B.8.1 Installation of ESM User Management

If the product is only partially installed, you may need to remove the marker files before reinstalling the product. For more information about the marker file locations for ESM refer to B.6.1, “Troubleshooting ESM TMR Server Installs” on page 663.

B.8.2 User Management Notice Groups

At least one administrator should be assigned the notice group Oracle User Management. This will receive a notice for each error condition, such as a distribute failing.

B.8.3 User Management Roles

You will need a combination of Tivoli roles and ESM roles to use User Management. The roles required depend on the type of distribution. The following tables will show the dependencies:

Table 46. Roles Required for Distributing to Next Level of Subscribers

Activity	Context	Required Roles
Distribute one or more profiles from a profile manager.	Profile Manager	admin

Table 47. Roles Required for Distributing to All Levels of Subscribers

Activity	Context	Required Roles
Distribute one or more profiles from a profile manager.	Profile Manager	admin
Edit and distribute an endpoint profile.	Endpoint	admin
Update the database.	Endpoint	oracle_dba

Table 48. Roles Required for Distributing from the Endpoint

Activity	Context	Required Roles
Edit and distribute an endpoint profile.	Endpoint	admin
Update the database.	Endpoint	oracle_dba

B.8.4 Overview of Passwords in OracleUser Profiles

User management store the user password. It is held in two different formats depending on where the information comes from:

- If an administrator types in the password, it is held as a Tivoli encrypted string.

- If the password is populated from a database endpoint, it is held as an Oracle encrypted string.

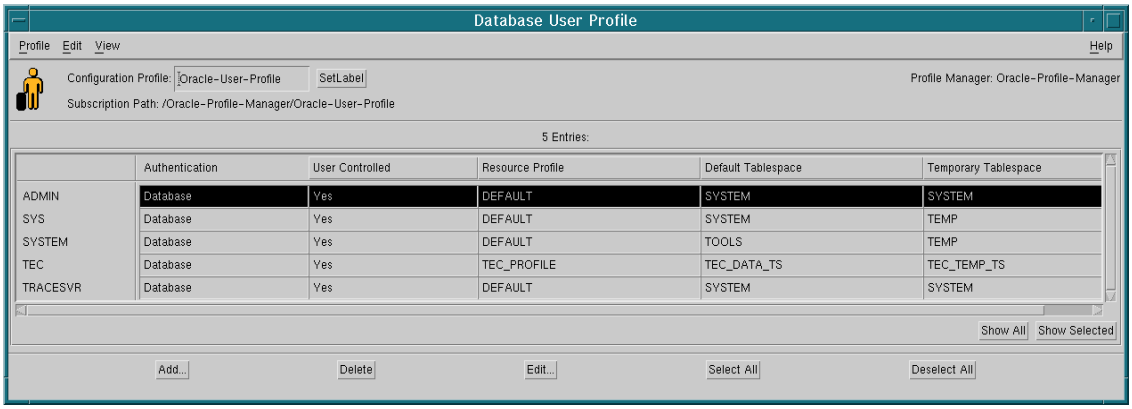


Figure 251. Oracle Database User Profile

In an ESM OracleUser profile, there are two flags associated with each user password:

- The first flag determines if the password is Tivoli-encrypted or Oracle-encrypted. This flag is not visible on the GUI as it is controlled internally.
- The second flag determines if the Oracle user (the actual person, not the user account) sets their own password. This is visible on the GUI, as the **User Controls Password** option.

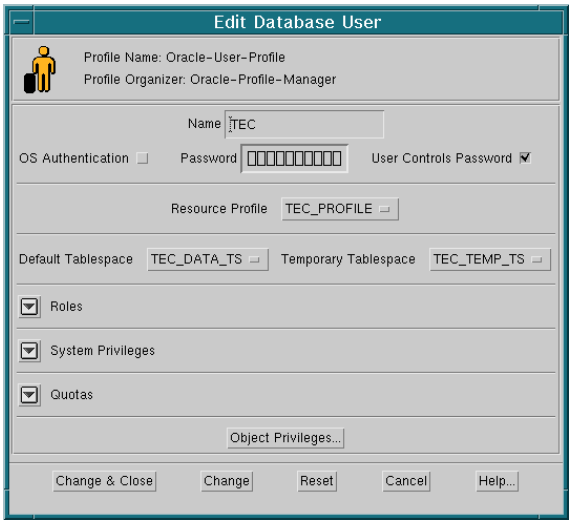


Figure 252. Edit TEC Oracle Database User

Note

If the password is user controlled, then no matter what the password is in the profile, it will not be changed during a push. If the password is not user controlled, then it will be changed during every push regardless of how it is stored (according to flag 1).

During a populate, all user controlled flags are set to `Yes` so that pushing out a currently-populated user will not change the password.

If the administrator wants to change a password, they must set the user controlled flag to `No` (or deselect it on the GUI) and then push the profile out.

B.8.5 Deleting Database User Records

If a user profile has been distributed, and a user record is deleted from the profile and pushed, it will de deleted at the endpoint, meaning that the user record will be removed from the Oracle7 database. If the profile has not been distributed, the record can be removed from the profile and pushed, and the user will not be deleted from the endpoint.

B.8.6 Background Daemons

Table 49 summarizes the daemons in use for ESM User Management:

Table 49. ESM User Profile Background Daemons

Daemon	Object Type	Description
M7UserDomainAgentRunBase	M7UserDomainAgent	Lists names and details of users, roles, and resources. Provides synchronization facility.
M7UserDomainAgentRunPush	M7UserDomainAgent	Changes and rolls back users, roles and resources pushed.
M7DatabaseResourceRunBase	Oracle7ResourceProfile	Provides all non-GUI profile operations for resource profiles.
M7DatabaseResourceRunGUI	Oracle7ResourceProfile	Provides all GUI profile operations for resource profiles.
M7DatabaseResourceRunIterator	Oracle7ResourceProfile	Caches CCMS data for performance.
M7DatabaseRoleRunBase	Oracle7RoleProfile	Provides all non-GUI profile operations for role profiles.
M7DatabaseRoleRunGUI	Oracle7RoleProfile	Provides all GUI profile operations for role profiles.
M7DatabaseRoleRunIterator	Oracle7RoleProfile	Caches CCMS data for performance.
M7DatabaseUserRunBase	Oracle7UserProfile	Provides all non-GUI profile operations for user profiles.
M7DatabaseUserRunGUI	Oracle7UserProfile	Provides all GUI profile operations for user profiles.
M7DatabaseUserRunIterator	Oracle7UserProfile	Caches CCMS data for performance.

B.9 Removing ESM Database Management Software

At the present time, there is no un-install program and no easy way to remove ESM Database Management Software. The binaries go into the Tivoli $BINDIR (%BINDIR%) directories. Always backup the Tivoli database before installing ESM Database Management software, then if the installation corrupts the database, or you decide to manually un-install the software, you have a backup that can be restored.

Refer to Section B.6.2, “Reinstalling Failed Server Installations” on page 663 for information regarding marker files that need removing if you intend to reinstall the products.

Appendix C. RDBMS Install Examples

Note

This appendix has NOT been updated from the previous edition of this book. We have included it as much of the information still applies to more current databases.

For any installation, you will need the assistance of the Database Administrator.

An excellent source for RDBMS installation and setup documentation is the *TME10 Inventory 3.2: New Features and Database Support* redbook, SG24-2135.

This appendix has been included as an example of installing an RDBMS for use with Tivoli. See also Chapter 9, “RDBMS Interface Module (RIM)” on page 313.

C.1 Installing an Oracle RDBMS

This appendix details the process for installing Oracle for use with TEC. The installation was performed on AIX, but this appendix includes a great deal of information applicable to all UNIX platforms and the use of Oracle for other TME 10 RIM applications. If a Database Administrator (DBA) exists for the database products, they should be involved in setting up this environment. Very often, who is installing TEC may also have to become the DBA.

C.1.1 Installing Oracle on UNIX 7.3.2.1

Perform the following steps.

1. Check for free disk space. A total of 550 MB is needed. 200 MB are only temporary.
2. Create `dba` and `oper` UNIX user groups. The only important thing is the group name itself; all the other parameters can be default using SMIT.
3. Create an Oracle software owner login. The name of the user must be oracle and the UID number should be greater than three and the group this user belongs to must be dba. Make the home directory `/home/oracle` and the login shell `/bin/ksh`.
4. Make a separate file system for the code and the temporary files. This makes it easier to identify the oracle things and to wipe out the temporary

files after the installation, which frees up a lot disk space. In our case, we called them:

- /oracle/home
- /oracle/temp

5. Now, the permissions for the two new file systems have to be granted by entering:

```
chown oracle.dba /oracle/home
chown oracle.dba /oracle/temp
chmod 755 /home/oracle
chmod 755 /home/temp
```

6. Create a local bin directory and change the permission by entering:

```
mkdir /usr/lbin
chmod 777 /usr/lbin
```

7. The next step is to set the variable for the Oracle owners environment. Therefore, you must login as `oracle` (or type `su - oracle`) and run the following commands:

```
umask 022
vi .profile
```

8. Enter the following variables in /profile with your site-specific values:

- ORACLE_HOME=/home/oracle
- export ORACLE_HOME
- ORACLE_SID=D1
- export ORACLE_SID
- ORACLE_TERM=lft
- export ORACLE_TERM
- PATH=$ORACLE_HOME/bin:/usr/lbin:/usr/local/bin:$PATH
- export PATH

9. Add the SQL `*Net V2` listener by adding the following line to the `/etc/services` file: `listener 1521/tcp #oracle`
10. Login as `oracle`.
11. su root.
12. Create a directory for the CDROM, for example, `/cdrom`.
13. Grant permission to this directory by typing `chmod 777 /cdrom`.
14. Grant permission for the temporary directory by typing `chmod 777 /oracle/temp`.
15. Mount the CDROM by issuing `mount -rv cdrfs /dev/cd0 /cdrom`.
16. Type `exit` to get to the Oracle user again.

17. Change to directory `/cdrom/orainst` and type `./start.sh` to run the startup script.
18. Running the `rootpre.sh` script:

```
su root
cd /oracle/temp/orainst
./rootpre.sh
exit
```

19. Reboot.
20. Login as `oracle` again and mount the Oracle `cd`.
21. Run the installer by entering: `/oracle/temp/orainst/orainst`. The installer posts a lot of dialog panels, and it takes some time to answer them. The first question without a dialog asks if you have run the `rootpre.sh`. If the procedure in this book has been followed, it's already done, that means the answer is `yes`.

Next, are the Installer dialog panels:

22. First is the Welcome panel. Select **OK** and hit **Enter**.
23. Installation Activity Choice? --> Select **Install/Upgrade** and **OK**.
24. Installation Options? --> Select I**nstall New Product**.
25. Mount Point panel? --> In our case, the mount point is `/oracle/home`.
26. Next prompt is for complete ORACLE_HOME location. It should be `/oracle/home/app/oracle/product/7.3.2`. If it is correct, enter `OK`.
27. Create DB Objects? --> Select **Yes**.
28. After this, there may be a few information prompts. Just answer **OK** to these screens.
29. OPS Install? --> Select **No**.
30. Question for successfully completion of running `rootpre.sh`? --> Answer: `Yes`.
31. Install Products on all nodes? --> `No`.
32. Install Source? --> Install from staging area (`/oracle/temp`).
33. OEACLE_SID?--> In our case, it is `D1`.
34. National Language Support --> We have chosen **American/English**.
35. Relinking Executable? --> `No`.

36. Location of Post-Installation File? --> In our case, it is `/oracle/home/app/oracle/product/7.3.2/orainst/root.sh`. Confirm with **OK** if it is correct.

37. Online Help Documentation? It is your choice, we said `No` because we wanted to save disk space. There are a few pages about documentation, and for all of them we did the same.

38. Software Asset Manager: We have chosen the following list:

    - SQL*NET (V2) 2.3.2.1.0
    - SQL*Plus 3.3.2.0.0
    - TCP/IP Protocol Adapter (V2)

39. The next prompt is for the official host name. In our case, `rh0255f`.

40. TCP service port. Earlier in the install, we have decided to use port #1521, that means `1521` goes in here.

41. The next few panels are password prompts. This is all your choice.

42. The next panels will ask for group names that will be assigned to Oracle users, enter `dba` in all cases.

43. The next important panel is to choose the storage type for the database. We have decided to use a file system-based DB instead of having raw repositories.

44. When you are asked if you want to spread database objects over three mount points, answer `NO`, because one mount point is good enough. In our case, it is `/oracle/home`.

45. The next few panels require individual input.

46. SQL*Net Listener automatically started? --> `Yes`.

47. Database Domain Name: In our case, `D1_Domain`.

48. The following panels ask for confirmation of default settings. The answer is `YES` to all of them.

49. The next questions are about having SQL help and demo facilities installed and can be answered with `NO`.

50. When you get the finish message, the installer can be exited.

51. At this point, the installation is not finished because it is necessary to run the Web-based installation facility to finish up the installation.

52. Start a Web browser and enter `http://IP-Name of the data base server:defined service port/ows-abin/boot`, which will skip the registration procedure.

53. At this Web page, you have to do three things:

1. Configure an Oracle Web Agent service called OWA_DBA
2. Configure an Oracle Web Listener
3. Configure a default Oracle Web Agent service called OWA_DEFAULT_SERVICE

After all this is finished, the Oracle installation is complete and you will see a success panel like that shown in Figure 253:

Figure 253. Web-Based Oracle Installation

To be able to connect with TEC to the Oracle database, an Oracle service must exist, and, therefore, the Oracle listener has to be started with this service. On Windows NT, the service and the related files can be created through a GUI. But on UNIX, everything has to be done manually, and for the `tnsnames.ora` file, there is no example provided with the Oracle product. The following screens will show examples for the necessary files called `tnsnames.ora` and `listener.ora`:

```
This screen.shows an example for a listener.ora file
LISTENER=
 (ADDRESS_LIST=
  (ADDRESS=
   (PROTOCOL=TCP)
   (HOST=rh0255f)
   (PORT=1523)
  )
 )

STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10

SID_LIST_LISTENER=
 (SID_LIST=
  (SID_DESC=
   (SID_NAME=D1)
   (ORACLE_HOME=/oracle/home/app/oracle/product/7.3.2)
  )
 )

TRACE_LEVEL_LISTENER = OFF
USE_CKPFILE_LISTENER = true
```

Figure 254. Listener.ora File for Oracle

```
This screen.shows an example for a tnsnames.ora file
D1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
(ADDRESS =
(PROTOCOL = TCP)
    (Host = rh0255f)
    (Port = 1523)
)
    )
    (CONNECT_DATA =
       (SID = D1)
    )
  )
```

Figure 255. Tnsnames.ora File for Oracle

These two files are very sensitive. We received one file as an example with comments in it. The error we got when we tried to connect the TEC to the database running the cr_tec_db.sh script is:

```
MGR-02073: an error occurred while connecting to a database
ORA-06401: NETCMN: invalid driver designator
```

Whenever there is an error like the one above, it is possible to get more information about it using the `oerr` command provided by Oracle. The syntax for this example is `oerr ORA 06401`. This is available in UNIX only. But in this case, it still doesn't tell you that the cause of the error is the comments in the `tnsnames.ora` file.

To avoid problems like this, you can verify the Oracle installation prior to the attempt to connect TEC to Oracle. See the next section "Oracle Installation Verification" on page 685.

C.1.2 Oracle Installation Verification

The Oracle Server Manager is one vehicle to verify that Oracle is working properly. To use this, root users on UNIX should invoke it from `$ORACLE_HOME/bin` by issuing the `svrmgrm` command, which is shown in Figure 256. When the Server Manager GUI pops up, you must enter an Oracle internal user ID and a proper password. By default, you can use `sys` as User ID and password `sys` as well. The service name is the name that has been defined in the `tnsnames.ora` file, and this is the name that will be used by the RIM module when TEC is attempting to connect to the Oracle database.

In Windows NT, there is something similar called the Instance Manager. Selecting **Start -> Programs -> Oracle Enterprise Manager -> Instance Manager** from the NT GUI will present the same picture.

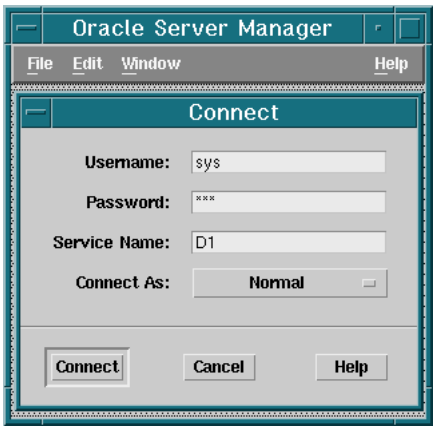


Figure 256. Oracle Server Manager GUI

If everything is OK, then you will get some information about the instance you have queried. If you get an error message here, the RIM module won't be able to connect to the database.

As an Oracle user, change to directory $ORACLE_HOME/bin and run the `tnsping service-name` command where the service name is the name of your database service. In our case, it is `D1`. This command checks if your database and the listener are up and running. If your listener or your database is not running, you get a response such as is shown in Figure 257 and Figure 258:

```
This is the output for a Oracle listener ping on AIX
oracle@rh0255f:product/7.3.2/bin$ tnsping D1

TNS Ping Utility for IBM/AIX RISC System/6000: Version 2.3.2.1.0 - Production on
18-NOV-97 11:12:06

Copyright (c) Oracle Corporation 1995.  All rights reserved.

Attempting to contact (ADDRESS=(PROTOCOL=TCP)(Host=rh0255f)(Port=1523))
TNS-12541: TNS:no listener
```

Figure 257. Tnsping Output for Oracle on AIX

```
This is the output for Oracle listener ping on NT
F:\ORANT\BIN>tnsping orcl
TNS Ping Utility for 32-bit Windows: Version 2.3.2.1.0 - Production on
18-NOV-9 17:10:27

Copyright , 1996(c) Oracle Corporation 1995.  All rights reserved.

Attempting to contact (ADDRESS=(COMMUNITY=tcp.world)(PROTOCOL=TCP)
(Host=tp760do)(Port=1521))
TNS-12541: TNS:no listener
```

Figure 258. Tnsping Output for Oracle on NT

When everything is OK, you will get something, such as `OK (290 msec)`, instead of the line `TNS-12541: TNS:no listener`.

If something is wrong, it is possible to verify if the database itself is not running or only the listener is in bad shape.

Another check is to use the Listener Control Program offered by Oracle. As an Oracle user, change to directory $ORACLE_HOME/bin and issue the command `lsnrctl`, which puts you into the listener control program. There, you can run commands to start and stop the listener and show the listener status. The `help` command shows the syntax for each of these commands. The output for a perfectly running listener looks like the following in Figure 259:

```
This is output for a Oracle listener status on NT.
LSNRCTL> status
Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=oracle.world))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for 32-bit Windows: Ver 2.3.2.1.4-Production
Start Date                18-NOV-97 17:38:49
Uptime                    0 days 0 hr. 21 min. 49 sec
Trace Level               admin
Security                  ON
SNMP                      OFF
Listener Parameter File   F:\ORANT\network\admin\listener.ora
Listener Log File         F:\ORANT\network\log\listener.log
Listener Trace File       F:\ORANT\network\trace\listener.trc
Services Summary...
ORCL          has 1 service handler(s)
The command completed successfully
```

Figure 259. Status Command of Oracle Listener Control Facility

The commands and the outputs are similar for UNIX and Windows NT.

To start the database on UNIX, login as a Oracle user and change to directory `$ORACLE_HOME/bin` and run the command `dbstart`. On NT, start the Oracle services.

You can also verify if the database is running or not by issuing the command `wtdbstat` from the RIM host or from the TMR Server.

Appendix D. Special Notices

This publication is intended to help technical users of Tivoli Enterprise products to understand more about the Tivoli Management Framework and applications. The information in this publication is not intended as the specification of any programming interfaces that are provided by the Tivoli Enterprise application suite. See the PUBLICATIONS section of the IBM Programming Announcement for the Tivoli Framework and 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 IBMs 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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.

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

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:

AS	OS/390
AS/400	OS/400
AIX	RISC System/6000
IBM ®	SPI
OS/2	400

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java, HotJava, and Sun Solaris are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT Performance Monitor, Windows NT, and the Windows 95 logo are trademarks
or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Appendix E. Related Publications

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

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 697.

- *The OS/390 Security Server Meets Tivoli*, SG24-5339
- *Introducing TME 10 Security Management*, SG24-2021
- *Tivoli Security Management Design Guide*, SG24-5101
- *Mass Installation Using SIS*, SG24-5109
- *Using Databases with Tivoli Application and RIM*, SG24-5112
- *New Features in Tivoli Software Distribution 3.6*, SG24-2045
- *A First Look at TME 10 Distributed Monitoring 3.5*, SG24-2112
- *TME 10 Inventory 3.2: New Features and Database Support*, SG24-2135
- *Getting Started with Tivoli User Administration*, SG24-2015
- *Tivoli User Administration Design Guide*, SG24-5108
- All About Tivoli Management Agents, SG24-5134

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041

CD-ROM Title	Subscription Number	Collection Kit Number
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

E.3 Other Publications

These publications are also relevant as further information sources:

The following Tivoli publications mentioned in this redbook are Product Documentation, which can only be obtained by purchasing the associated Tivoli product:

- *Tivoli User Administration NIS Management Guide*
- *Tivoli User Administration User and Group Management Guide*
- *Tivoli Framework Planning and Installation Guide*
- *Tivoli Framework Reference Manual*
- *Tivoli Software Installation Service User's Guide*
- *Tivoli Framework User's Guide*
- *Tivoli Advanced Development Environment Manuals*
- *Tivoli Framework Planning and Installation Guide, Version 3.2*
- *Tivoli Framework Reference Guide*
- *Tivoli Enterprise Console Adapters Guide*
- *Tivoli Console User's Guide*
- *Tivoli Inventory User's Guide*
- *Tivoli Inventory Release Notes*
- *Tivoli Distributed Monitoring User's Guide*
- *Tivoli Software Distribution Autopads User's Guide*
- *Tivoli Software Distribution User's Guide*
- *Tivoli ADE- Application Development for the Lightweight Client Framework, 3.6*
- *Tivoli Framework Planning and Installation Guide*
- *Tivoli Software Distribution Reference Manual*
- *Tivoli Security Management User's Guide*
- *Tivoli Security Management 3.6.1*

- *Tivoli Security Management Reference Manual for TACF*

The following Microsoft publications mentioned in this redbook are Microsoft Knowledge Base documents and can be found at the following Web site: `www.microsoft.com`

- *Q96005*
- *Q102716*
- *Q122422-From TechNet*

The following Microsoft publications mentioned in this redbook are Product Documentation, which can only be obtained by purchasing the associated Microsoft product:

- *NT Resources Kit 4.0*

The following Sybase publications mentioned in this redbook are Product Documentation, which can only be obtained by purchasing the associated Sybase product:

- *EMS Sybase SQL Server Framework Guide*

The following Oracle publications mentioned in this redbook are Product Documentation, which can only be obtained by purchasing the associated Oracle product:

- *Oracle DBA Handbook*

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO 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 via e-mail including information from the 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 for customer may be found at http://www.redbooks.ibm.com/ and for IBM employees at http://w3.itso.ibm.com/.

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 also view redbook. residency, and workshop announcements at http://inews.ibm.com/.

IBM Redbook 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.

List of Abbreviations

ACC	AutoPack Control Center
ACF	Adapter Configuration Facility
ACL	Access Control List
ADE	Advanced Development Environment
AEF	Application Extension Facility
ALI	Authentication, Location and Inheritance
ANSI	American National Standards Institute
API	Application Programming Interface
AS/400	Application S/400
BARC	Before, After, Remove & Configuration (script)
BDC	Backup Domain Controller
BDT	Bulk Data Transfer
BO	Behavior Object
BOA	Basic Object Adapter
CCMS	Configuration and Change Management System
CLI	Command Line Interface
CORBA	Common Object Request Broker Architecture
DBA	Database Administrator
DBDIR	DataBase Directory
DFW	Desktop for Windows
DHCP	Dynamic Host Configuration Protocol
DII	Dynamic Invocation Interface
DLL	Dynamically Linked Library
DM	Domain Manager
DMTF	Desktop Management Task Force
DO	Display Object
DPO	Default Policy Object
DRM	Default Routing Manager
ESM	Enterprise Server Management
FAT	File Allocation Table
FQHN	Fully Qualified Host Name
FSFI	Free Space Fragmentation Index
GCOS	GE Computer Operating System Field
GEM	Global Enterprise Manager
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
IANA	Internet Assigned Number Authority
IBM	International Business Machines Corporation
IDL	Interface Definition Language
IOM	Inter-Object Messaging
IPC	Inter-Process Communication

IPX/SPX Internet Packet Exchange/Sequenced Packet Exchange

ITSO International Technical Support Organization

LCF Lightweight Client Framework

LDAP Lightweight Directory Access Protocol

LSA Local Security Authentication

LQM Local Queue Manager

MCSL Monitoring Capabilities Subscription Language

MDist Multiplexed Distribution

MIF Management Information File

NAT Network Address Translation

NIS Network Information Service

NLS National Language Support

NQM Network Queue Manager

NTFS NT File System

OID Object ID

OMG Object Management Group

OOB Out of Band

ORB Object Request Broker

PAT Port Address Translation

PD/PSI Problem Determination/Problem Source Isolation

PDC Primary Domain Controller

PO Prototype Object

RACF Resource Access Control Facility

RCS Revision Control System

RDBMS Relational DataBase Management System/Server

RIM RDBMS Interface Module

RPC Remote Procedure Call

SAM Security Account Manager

SCD Spoolmate Configuration Database

SeOS Security Operating System

SMB Server Message Block

SIS Software Installation Service

TACF Tivoli Access Control Facility

TAP Tivoli Authentication Package

TCP/IP Transmission Control Protocol/Internet Protocol

TEC Tivoli Enterprise Console

TEIDL Tivoli-Extended IDL

TLI Transport Layer Interface

TLL Task Library Language

TMA Tivoli Management Agent

TMF Tivoli Management Framework

TME Tivoli Management Environment

TMR	Tivoli Management Region or TME 10 Management Region
TNR	Tivoli Name Registry
TNWR	TME 10 NetWare Repeater
TRAA	Tivoli Remote Access Account
TRIP	Used to Refer to the Tivoli Remote Execution Service
UED	Unison Enterprise Database
UID	User Identifier
VPO	Validation Policy Object
VPN	Virtual Private Network
WINS	Windows Internet Naming Service
XBO	Extended Behavior Object

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

ITSO Redbook Evaluation

Tivoli Enterprise Internals and Problem Determination
SG24-2034-01

Your feedback is very important to help us maintain the quality of ITSO 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!)

_____

_____

_____

_____

SG24-2034-01
Printed in the U.S.A.

Tivoli Enterprise Internals and Problem Determination

SG24-2034-01