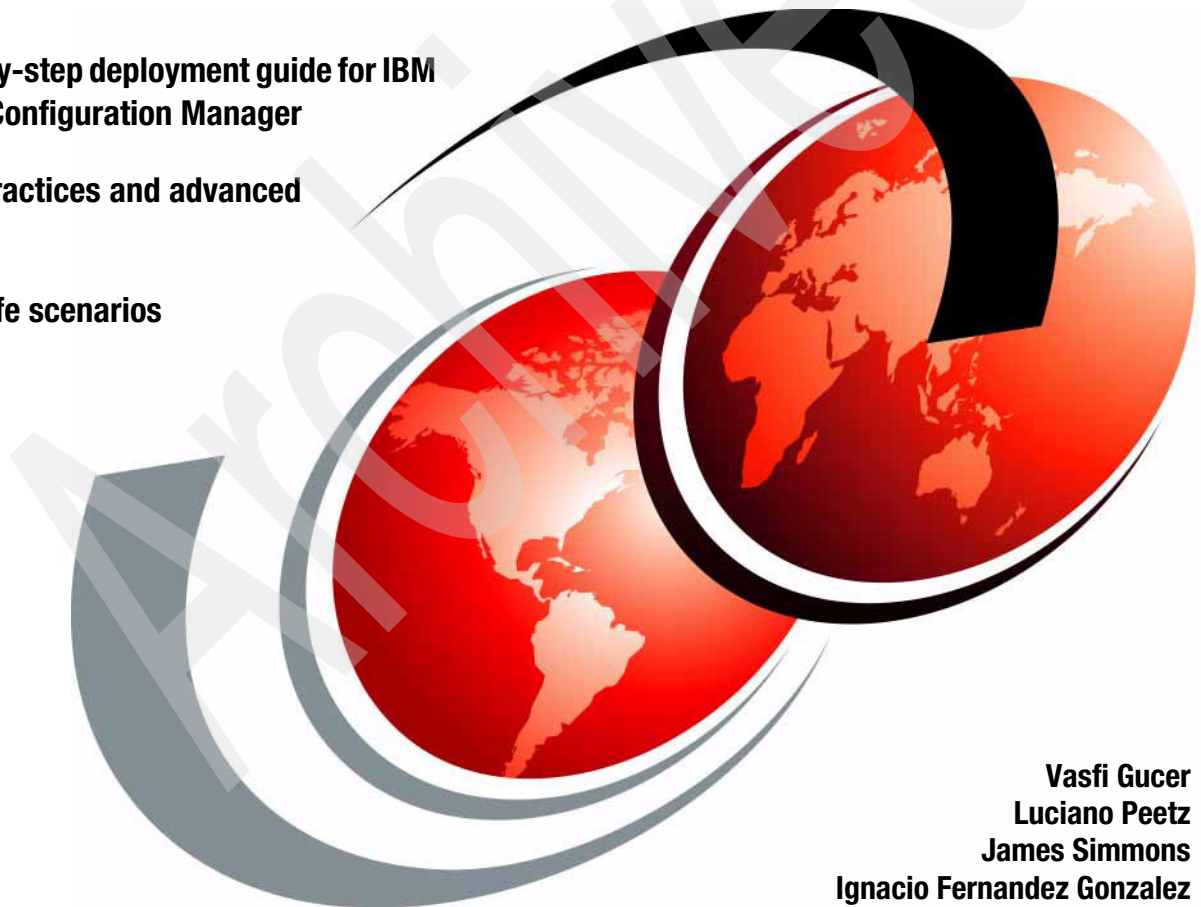


Deployment Guide Series: IBM Tivoli Configuration Manager

Step-by-step deployment guide for IBM
Tivoli Configuration Manager

Best practices and advanced
tuning

Real-life scenarios



Vasfi Gucer
Luciano Peetz
James Simmons
Ignacio Fernandez Gonzalez



International Technical Support Organization

**Deployment Guide Series: IBM Tivoli Configuration
Manager**

September 2005

Archived

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

First Edition (September 2005)

This edition applies to IBM Tivoli Configuration Manager Version 4.2.2.

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

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

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this redbook	xi
Become a published author	xii
Comments welcome	xiii
Chapter 1. Introduction to IBM Tivoli Configuration Manager	1
1.1 Business case for IBM Tivoli Configuration Manager	2
1.2 IBM Tivoli Configuration Manager features	2
1.3 IBM Tivoli Configuration Manager components and services	3
1.3.1 Software Distribution	3
1.3.2 Inventory	3
1.3.3 Activity Planner	4
1.3.4 Change Manager	6
1.3.5 Web Gateway	7
1.3.6 Resource Manager	8
1.3.7 Web Interface	8
1.3.8 Enterprise Directory Query Facility	9
1.3.9 Data Moving	10
1.3.10 Pristine Manager	10
1.4 Understanding the IBM Tivoli Configuration Manager evolution	12
1.4.1 A little bit of IBM Tivoli Configuration Manager history	13
1.4.2 Tivoli Configuration Manager V4.2.x product-based releases	14
1.4.3 Tivoli Configuration Manager releases evolution summary	21
1.4.4 Tivoli Management Framework	21
Chapter 2. Installation planning	25
2.1 Expertise required	27
2.1.1 DB2 administrator	27
2.1.2 Operating systems administrator	27
2.1.3 Network administrator	28
2.1.4 Other products expertise	28
2.2 Creating a deployment plan for Tivoli Configuration Manager	30
2.3 How to deploy Tivoli Configuration Manager components	31
2.3.1 Distributed Tivoli Configuration Manager components	31
2.3.2 Tivoli server configuration	31
2.3.3 Components for managed nodes	32

2.3.4	Components for gateways	32
2.3.5	Components for endpoints	33
2.4	Required roles for installation	33
2.5	RDBMS considerations	34
2.6	RIM considerations	36
2.7	General preinstallation checks, hints, and tips	38
2.7.1	UNIX	39
2.7.2	Microsoft Windows systems on Intel 486 or Pentium systems	40
2.8	Installation methods	40
2.9	Overview of the integrated installation	41
2.10	Server installation	41
2.10.1	Typical server installation	42
2.10.2	Custom server installation	43
2.10.3	Starting the installation programs	43
2.11	Desktop installation	44
2.12	Web Gateway installation	45
2.12.1	Web Gateway prerequisites	46
2.12.2	Starting the installation program	46
2.13	Uninstallation of IBM Tivoli Configuration Manager	46
2.14	Architectural considerations for Tivoli regions	47
2.14.1	Benefits of connecting Tivoli regions	48
2.14.2	Connection types	48
2.14.3	Case study: Hub-spoke architecture	49
2.15	Backup strategies	52
2.15.1	Tivoli backups	53
2.15.2	System-level backups	53
Chapter 3. Installation of IBM Tivoli Configuration Manager in a small/medium enterprise		55
3.1	The XYZ Corporation	56
3.2	Description of the environment	56
3.3	Installing DB2 UDB on a Microsoft Windows server	57
3.4	Installation of Tivoli Management Framework and Configuration Manager components (except Web Gateway, LDAP, and Pristine Manager)	74
3.5	Endpoint installation	99
Chapter 4. Installation of IBM Tivoli Configuration Manager in a large enterprise		111
4.1	The ABBC Company	112
4.2	IBM Tivoli Configuration Manager implementation at ABBC	112
4.3	Installation of Tivoli Configuration Manager V4.2.2 server on AIX 5L	113
4.3.1	General prerequisites for the installation	113
4.3.2	The server installation	114

4.4	Configuring DB2	141
4.4.1	Running the admin scripts	142
4.4.2	Running the schema scripts	144
4.4.3	Running the plug-ins	146
4.5	Setting the Tivoli environment variables	147
4.6	Desktop installation	148
4.6.1	Desktop installation procedures	148
4.6.2	Opening the Tivoli desktop	154
4.7	Installing a gateway on Linux	156
4.7.1	Linux considerations	156
4.7.2	Installing a Linux managed node	158
4.7.3	Defining the managed node as a gateway	168
4.8	Installing Tivoli Configuration Manager components on new gateway	171
4.8.1	Installing the Scalable Collection Service, Version 4.2.2	171
4.8.2	Installing the Inventory Gateway, Version 4.2.2	177
4.8.3	Installing the Pristine Manager Gateway, Version 4.2.2	183
4.8.4	Installing the Resource Manager Gateway, Version 4.2.2	189
4.8.5	Installing the Software Distribution Gateway, Version 4.2.2	195
4.8.6	Checking the installed products	201
4.9	Installing endpoints	203
4.9.1	Installing endpoints on a Windows operating system	203
4.9.2	Installing endpoints on a Linux operating system	210
4.9.3	Installing endpoints on an AIX 5L operating system	212
4.9.4	Installing endpoints on an OS/400 operating system	214
Chapter 5. Extra components		217
5.1	IBM DB2 UDB installation	218
5.2	LDAP server installation	237
5.2.1	System requirements	237
5.2.2	Basic LDAP concepts	237
5.2.3	Installing IBM Tivoli Directory Server V5.2 on a Windows server	238
5.2.4	Configuration steps after IBM Tivoli Directory Server installation	245
5.3	IBM WebSphere Application Server installation	259
5.3.1	Before installing WebSphere Application Server V5.1	259
5.3.2	Installing WebSphere Application Server V5.1	260
5.4	Tivoli Access Manager and Access Manager WebSEAL installation	266
Chapter 6. Enterprise Directory integration		275
6.1	Enterprise Directory integration	276
6.1.1	Exchanging data with a directory server	276
6.1.2	Working with DirectoryContext objects	276
6.2	Directory Query Library and Directory Query	279
Chapter 7. IBM Tivoli Web Gateway for Web User Interface		289

7.1	System requirements	290
7.2	Software requirements	290
7.2.1	Configuring IBM DB2 UDB V8.1 server	290
7.2.2	Configuring IBM WebSphere Application Server	292
7.2.3	Configuring IBM Tivoli Access Manager and IBM Tivoli Access Manager WebSEAL	292
7.3	Installing IBM Tivoli Web Gateway	296
7.4	Preparing to use the Web UI	306
7.4.1	Assigning the WebUI_Admin role	306
7.4.2	Creating Web UI users	307
7.4.3	Publishing profiles to the Web Gateway	307
7.4.4	Starting the Web UI	308
Chapter 8.	Tuning and best practices	311
8.1	Analyzing your Tivoli Configuration Manager environment	312
8.1.1	Applying fix packs or implementing a new version	312
8.1.2	Upgrading: Gradual or everything at the same time	312
8.1.3	Assessing your Tivoli Configuration Manager environment: Assessment tool	313
8.1.4	Tivoli Configuration Manager upgrade: SD_CMSTATUS_VIEW	314
8.1.5	Hub-spoke configuration	314
8.1.6	Tivoli Web Gateway upgrade or migration	314
8.2	Tivoli Management Framework considerations	315
8.2.1	Tuning the operating system: Threads	315
8.2.2	Tuning Tivoli threads: Protecting the oserv	317
8.2.3	Gateway tuning	318
8.2.4	Network tuning parameters	319
8.2.5	Endpoint login storms	320
8.2.6	Protecting the endpoint manager	323
8.2.7	Gateway debug level not verbose	324
8.2.8	Slow links: Method download	324
8.2.9	Monitor disk/file system space and transaction log size	325
8.2.10	Framework V4.1.1: Health check parameters	326
8.2.11	Framework V4.1.1: Endpoint method download control	328
8.2.12	Framework 4.1.1-TMF-0010: TCP backlog	328
8.2.13	Framework 4.1.1-TMF-0010: Updated STATUS DATA in gatelog	328
8.3	Tivoli Configuration Manager Software Distribution	329
8.3.1	Adjusting your Software Distribution configuration	329
8.3.2	Monitoring your Software Distribution report flow	330
8.3.3	Software Distribution high, medium, and low session tuning	331
8.3.4	Notification manager report processing timeout	332
8.3.5	Database: Too many locks on the SD_CM_STATUS table	332
8.3.6	Protect the source host: notify_interval, conn_retry_interval	333

8.3.7	Distribution deadline and retry_ep_cutoff	334
8.3.8	Machine hangs while running a script or during reboot.	334
8.3.9	Force reboot type during commit operations.	334
8.3.10	Software Distribution architecture hints and tips	335
8.3.11	Catalog recovery: SPE disconnected commands	336
8.3.12	Servers versus desktops	336
8.4	Activity Planner	336
8.4.1	Plan Monitor submission tuning: Caching endpoint information . . .	337
8.4.2	Reporting performance tuning: Skipping notification manager . . .	338
8.4.3	Plan performance tuning: Number of activities	338
8.4.4	Activity Planner Monitor: Executer threads	339
8.4.5	Activity Planner Editor: Plan layout and large number of activities .	339
8.4.6	Deleting plans: Cancel pending	340
8.5	Change Manager.	340
8.5.1	Assign priority: Order of activities using Successful Target	340
8.5.2	Adding states to the desired state.	340
8.6	Inventory tuning	341
8.7	Resource Manager	342
8.7.1	Tivoli Web Gateway: Device control	342
8.7.2	From Tivoli Web Gateway to Tivoli Configuration Manager: Results process	343
8.7.3	Jobs in “I” status remain in JOB_RESULT	346
8.7.4	Jobs in “P” status remain in JOB_RESULT	347
8.7.5	Remaining directories in /twg/device/: No jobs in JOB_RESULT . .	348
8.7.6	Web Gateway logging settings	349
8.8	Pristine Manager	349
8.9	Tivoli Configuration Manager V4.2.3.	350
	Appendix A. Microsoft security checklist	351
	\$root_user	352
	Root access	352
	widmap	352
	Access rights	352
	tmersvd.	354
	Windows registry	355
	Principals of the user model	355
	TivoliAP.dll	355
	Tivoli remote access account	356
	Active Directory and TivoliAP.dll	356
	Tivoli roles authority structure	357
	Invoking a method.	357
	Appendix B. Scripts referenced in the book.	359

assess.pl	360
gw_tuning.pl	360
find_rogue	370
protect_epmgr.sh	374
metodos.pl	376
coord.sh	379
del_plan.sh	381
gen_cat.sh	383
Appendix C. Additional material	387
Locating the Web material	388
Using the Web material	388
System requirements for downloading the Web material	388
How to use the Web material	388
Abbreviations and acronyms	389
Related publications	391
IBM Redbooks	391
Other publications	391
Online resources	392
How to get IBM Redbooks	392
Help from IBM	392
Index	393

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX 5L™

AIX®

AS/400®

DB2 Universal Database™


DB2®

IBM®

Informix®

OS/2®

OS/400®

Redbooks (logo) ™

Redbooks™

SecureWay®

Tivoli Enterprise Console®

Tivoli Enterprise™

Tivoli®

WebSphere®

The following terms are trademarks of other companies:

Java, JavaHelp, JavaScript, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, MS-DOS, Windows server, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

Preface

IBM® Tivoli® Configuration Manager controls software distribution and asset management inventory in a multiplatform environment. It is designed for configuration, distribution, change, version, and asset management in a distributed computing environment. Working on top of IBM Tivoli Management Framework, IBM Tivoli Configuration Manager provides an integrated solution for managing complex, distributed enterprise environments.

This IBM Redbook introduces the IBM Tivoli Configuration Manager logical and physical components and covers detailed planning and implementation steps to deploy IBM Tivoli Configuration Manager in small-to-medium and large-sized environments, including IBM AIX® 5L™, Microsoft® Windows®, Linux®, and IBM OS/400® systems.

In addition, we talk about best practices, advanced customization, and tuning topics for IBM Tivoli Configuration Manager.

This IBM Redbook will be useful for IT specialists responsible for implementing IBM Tivoli Configuration Manager V4.2.x in customer environments.

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.

Vasfi Gucer is a Project Leader at the International Technical Support Organization, Austin Center. He worked for IBM Turkey for 10 years and has been with the ITSO since January 1999. He has more than 10 years of experience in the areas of systems management, networking hardware, and software on mainframe and distributed platforms. He has worked on various Tivoli customer projects as a systems architect in Turkey and the U.S. He writes extensively and teaches IBM classes worldwide on Tivoli software. Vasfi is also a IBM Certified Senior IT Specialist.

Luciano Peetz has been with IBM Brazil since he finished college in 1996. Since 2001, he has been working as an IT Specialist in the Tivoli Support Group. In his previous position, he was an AIX 5L and Linux Support Professional. He has worked on Tivoli Configuration Manager deployment and migration from previous versions in several financial institutions in Brazil. He holds the following certifications: AIX V4.3 System Administration Certification, IBM Certified

Deployment Professional - Tivoli Configuration Manager V4.2, IBM Certified Deployment Professional - Tivoli Configuration Manager V4.2.2, and IBM Certified Deployment Professional - Tivoli Workload Scheduler V8.2.

James Simmons has worked in the IT arena for more than 20 years, providing support for various products. He started working for IBM in the manufacture and testing of computer boards. He led a team of professionals that supported the IBM OS/2® operating system. He was also a co-leader of the team that scanned and tested OS/2 for Y2K readiness. Jim joined Tivoli five years ago and has extensive experience with IBM Tivoli Management Framework. Currently, he is working as a Customer Support Engineer in Level 2 Tivoli Management Framework Support.

Ignacio Fernandez Gonzalez has been working for IBM since 1995. Currently, he is working as an IT architect in Tivoli Services within the IBM Software Group. Ignacio has extensive experience in large deployments of the IBM Tivoli suite of products, playing a key role as the Systems Management Team Leader during the Sydney 2000 Olympic Games. He holds a master's degree in Telecommunications Engineering from the Universidad Politecnica de Madrid (UPM). Ignacio is a Member of the Institute of Electrical and Electronic Engineers (IEEE) and holds two IBM Patent Awards. Currently, he is involved in a systems management solution deployment for a financial institution in Madrid, Spain.

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

Sanver Ceylan, Elizabeth Barnes
International Technical Support Organization, Austin Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. OSJB Building 905
11501 Burnet Road
Austin, TX 78758-3493

Archived



Introduction to IBM Tivoli Configuration Manager

IBM Tivoli Configuration Manager (Configuration Manager for short) controls software distribution and asset management inventory in a multiplatform environment. It is designed for configuration, distribution, change, version, and asset management in a distributed computing environment. Working on top of IBM Tivoli Management Framework, Tivoli Configuration Manager provides an integrated solution for managing complex, distributed enterprise environments.

This chapter provides an introduction to Tivoli Configuration Manager and discusses the following topics:

- ▶ Business case for IBM Tivoli Configuration Manager
- ▶ IBM Tivoli Configuration Manager features
- ▶ IBM Tivoli Configuration Manager components and services
- ▶ Understanding the IBM Tivoli Configuration Manager evolution

1.1 Business case for IBM Tivoli Configuration Manager

Distributed enterprises are no longer just using traditional systems for end users. Enterprises are becoming more demanding and require faster and more diverse deployment, change, and asset management in order to achieve their goals. It is not uncommon for end users to have a desktop, laptop, pervasive device, and cellular phone all at once, while e-businesses run at almost chaotic speed to support their business objectives.

Tivoli Configuration Manager is the answer to helping businesses keep up with demanding deployment and asset issues. Using the Tivoli Configuration Manager Software Distribution module, companies can rapidly and efficiently deploy complex mission-critical applications to multiple locations from a central point. After systems have been deployed, the Inventory module enables users to automatically scan for and collect hardware and software configuration information from computer systems across their enterprise.

For example, using Tivoli Configuration Manager, sales information, such as price lists, can be sent to the pervasive devices on demand, mobile users can scan for hardware and software information of their laptops, while at the same time downloading the Tivoli management agent (TMA), and targets for a reference model can include an entire accounting department from an enterprise directory. For speedier distributions, organizations can leverage multicast. A 20 MB distribution to 300 computers at a bandwidth of 500 kilobytes (KB)/second can take, on average, 40 seconds to distribute to all the endpoints. Using unicast, this would take, on average, 3.3 hours.

It is important to understand that all the features and enhancements tie together to enable the environment more flexibility, efficiency, and faster deployment, helping to cut costs and time.

1.2 IBM Tivoli Configuration Manager features

Using Tivoli Configuration Manager, you can:

- ▶ Scan hardware and software to determine which enterprise assets are part of your inventory.
- ▶ Reduce the time and effort in installing and configuring your network population by centralizing and automating the distribution of software across your enterprise.
- ▶ Automate and schedule network operations.
- ▶ Monitor system and configuration changes.

- ▶ Manage the desired state of all elements of your network.
- ▶ Manage your enterprise environment across firewalls.
- ▶ Extend the scope of your managed network to include pervasive devices, such as personal digital assistants (PDAs).

1.3 IBM Tivoli Configuration Manager components and services

Tivoli Configuration Manager is an integrated software distribution and asset management suite that consists of two main components, Software Distribution and Inventory, and various services.

1.3.1 Software Distribution

Using the Software Distribution component, you can install, configure, and update software remotely within your network, eliminating the need to update software manually on numerous systems. You can:

- ▶ Distribute client/server applications and applications for desktops, laptops, and pervasive devices across multiplatform networks.
- ▶ Update existing software with later versions.
- ▶ Synchronize software on distributed systems.

1.3.2 Inventory

Using the Inventory component, you can gather and maintain up-to-date inventory information in a distributed environment quickly, accurately, and easily. This helps system administrators and accounting personnel manage complex, distributed enterprises.

Administrators and accounting personnel can perform the following tasks:

- ▶ Manage all enterprise systems centrally.
- ▶ Determine the installed software base.
- ▶ Confirm a software distribution.
- ▶ Supplement and replace physical inventory function.
- ▶ Assist in procurement planning.
- ▶ Check software requirements.
- ▶ Control assets.

For example, you can combine inventory and software distribution operations to determine if any critical files are missing, and then reestablish the proper

configuration. After creating and deploying management-ready applications, you can continually maintain the desired state of your systems by synchronizing applications and system configurations on an enterprise scale.

1.3.3 Activity Planner

Activity Planner is a deployment service that enables you to:

- ▶ Define a group of activities to be submitted as an activity plan.
- ▶ Submit or schedule the plan for running.
- ▶ Monitor the plan while it runs.

Activities are tasks that can be scheduled to be performed on a set of targets at specified times. Operations can include software distribution, inventory operations, and Tivoli tasks.

Activities contained in a plan can have dependencies associated with them that define circumstances under which the activity should be run. The running of the operation defined in the activity is performed by the application to which the operation belongs. The group of activities forms the activity plan.

Activity Planner is made up of two components, the Activity Plan Editor and the Activity Plan Monitor.

Activity Plan Editor

You can use the Activity Plan Editor to:

- ▶ Manage a group of activities, originating from different applications, as a single activity from a single machine in the network.
- ▶ Schedule the activity plan to run on a specific day and time, to repeat at specific time intervals, or repeat indefinitely.
- ▶ Schedule activities to run at specific time intervals during the week.
- ▶ Set conditions on activities so that the execution of one activity is dependent on the completion result of other activities.
- ▶ Save activity plans in a database to resubmit them at any future time.

Figure 1-1 on page 5 shows the Activity Plan Editor.

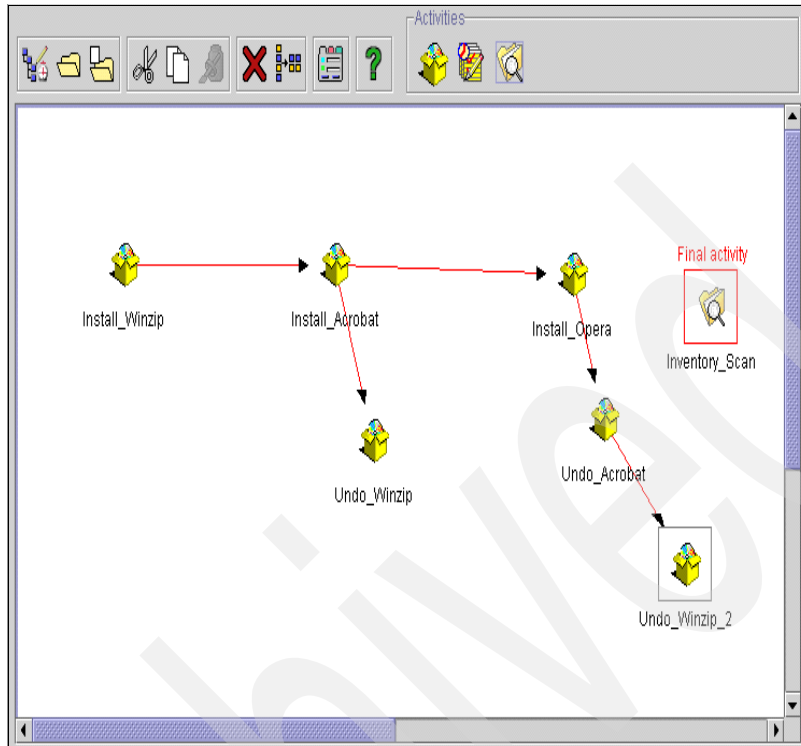


Figure 1-1 Activity Plan Editor

Activity Plan Monitor

You can use the Activity Plan Monitor to:

- ▶ Submit activity plans to be run.
- ▶ View all submitted activity plans along with their status, start time, and completion time.
- ▶ View the list of activities contained in the plan.
- ▶ View a graphical representation of the plan in the Activity Plan Editor window.
- ▶ For each activity, view the targets (gateways, depots) assigned to it.
- ▶ Perform operations such as pause, cancel, and resume.
- ▶ Restart an activity on an endpoint where the operation was unsuccessful.
- ▶ Delete the status information of a plan from the activity plan database.
- ▶ Launch the Distribution Status console to monitor and control software distributions submitted using the Activity Planner.

Figure 1-2 shows the Activity Plan Monitor.

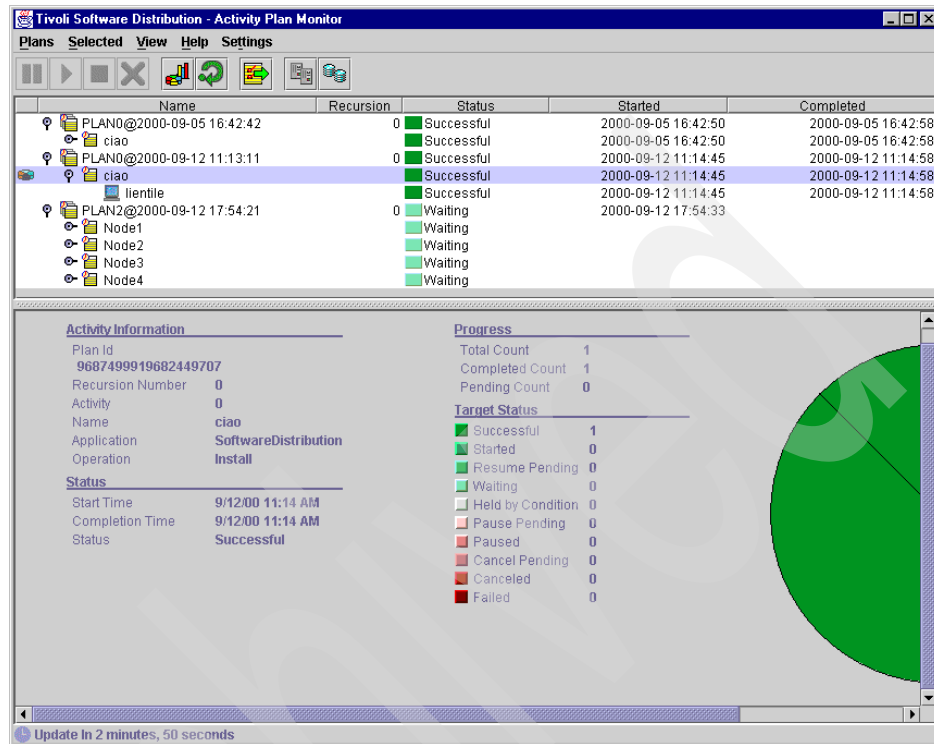


Figure 1-2 Activity Plan Monitor

1.3.4 Change Manager

Change Manager (previously called Change Configuration Manager) is a deployment service that, together with Activity Planner, supports software distribution, inventory, and change management in a large network.

Activity Planner is a prerequisite of Change Manager. Change Manager works with the Activity Plan Monitor to manage specified groups of users, workstations, or devices as single subscribers. Subscribers can be users, groups of users, endpoints, a profile manager, the results of a query, or pervasive devices.

Change Manager uses reference models, which contain an association of configuration elements and subscribers, to simplify the management of your network environment.

Figure 1-3 on page 7 shows the Change Manager.

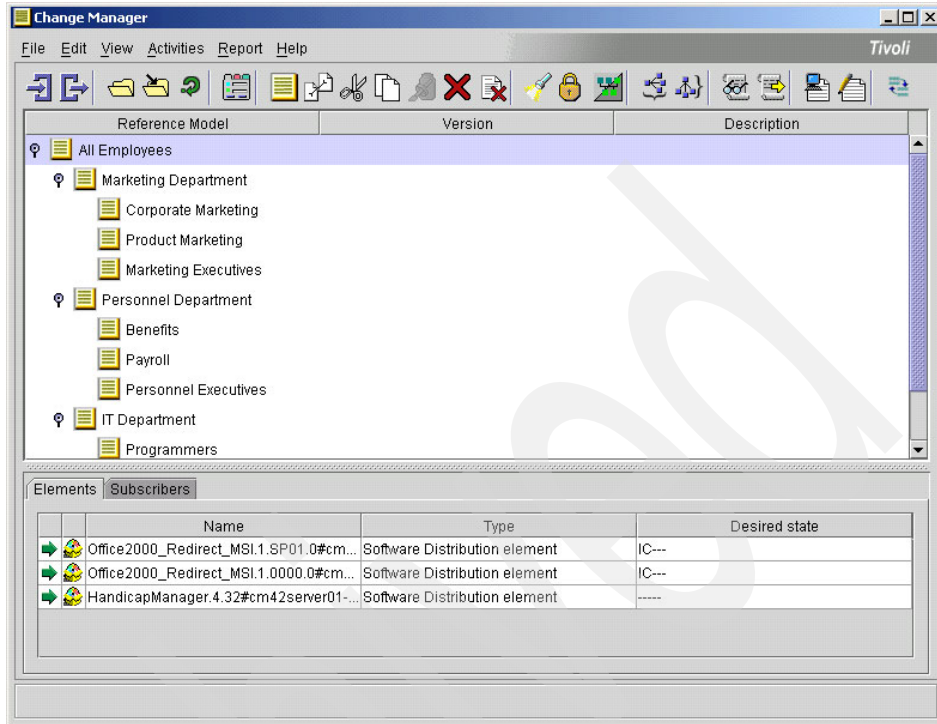


Figure 1-3 Change Manager

1.3.5 Web Gateway

The Web Gateway component supports the Resource Manager deployment service and the Web Interface (Web UI) deployment service.

The Resource Manager deployment service extends the traditional three-tier Tivoli environment to a fourth tier, thus providing software distribution, inventory, and management of pervasive devices such as the Palm Pilot, Pocket PC, and Nokia Communicator. (See 1.3.6, “Resource Manager” on page 8.)

The Web Interface (Web UI) enables software distribution and inventory to be initiated by users. By using the Web Interface, users can access a Web site and install software on their own machine, or generate an inventory scan by themselves. (See 1.3.7, “Web Interface” on page 8).

The Web Gateway component consists of two elements:

- ▶ Web Gateway database
- ▶ Web Gateway server code

These elements are installed on an endpoint machine in the Tivoli environment.

The Web Gateway uses IBM WebSphere® technology and provides improved security by leveraging Access Manager for authentication and the HTTPS protocol for secure communications.

1.3.6 Resource Manager

A Tivoli management region is a three-tier architecture, including servers, gateways, and endpoints, that is created using Tivoli Management Framework. By using the Resource Manager deployment service, you can extend the Tivoli region to a fourth tier, pervasive devices, such as PDAs.

Resource Manager is made up of two subcomponents: the Resource Manager, which is installed on a Tivoli server; and the Resource Manager Gateway, which is installed on a gateway that connects to an endpoint on which the Web Gateway component has been installed.

You can use Resource Manager, together with the Software Distribution, Inventory, and Web Gateway components, to perform the following operations:

- ▶ Add or remove pervasive devices.
- ▶ Provide access to devices for software distribution.
- ▶ Provide access to devices for inventory operations.
- ▶ Customize devices.

1.3.7 Web Interface

The Web Interface deployment service is a browser-based tool that enables remote management operations to be initiated by users on machines that do not have the Tivoli management agent installed. Figure 1-4 on page 9 shows the Web Interface.

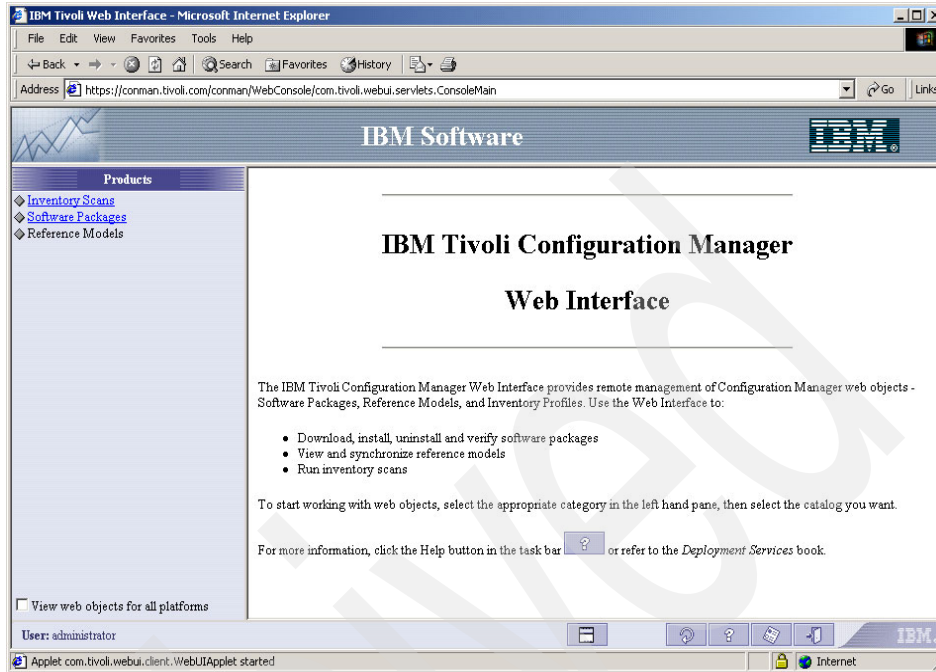


Figure 1-4 Web Interface

1.3.8 Enterprise Directory Query Facility

The Enterprise Directory Query Facility is a deployment service that enables an administrator to use information stored in enterprise directories inside a Tivoli environment. The administrator can select a specific directory object, or container of directory objects, as subscribers for a reference model or an activity plan. The subscribers can then be targets for software distribution or inventory scans.

The Enterprise Directory Query Facility enables you to access the information contained in an enterprise directory server.

The Enterprise Directory Query Facility consists of directory query libraries and directory queries. Directory query libraries reside in policy regions and are created to contain directory queries. Directory queries enable you to find information about the users or the workstations defined in the enterprise directory server.

Enterprise Directory Query Facility supports the following LDAP products:

- ▶ IBM SecureWay® Directory Server

- ▶ Active Directory for Windows 2000
- ▶ Novell Directory Server for NetWare

Figure 1-5 shows the Enterprise Directory Query Facility.

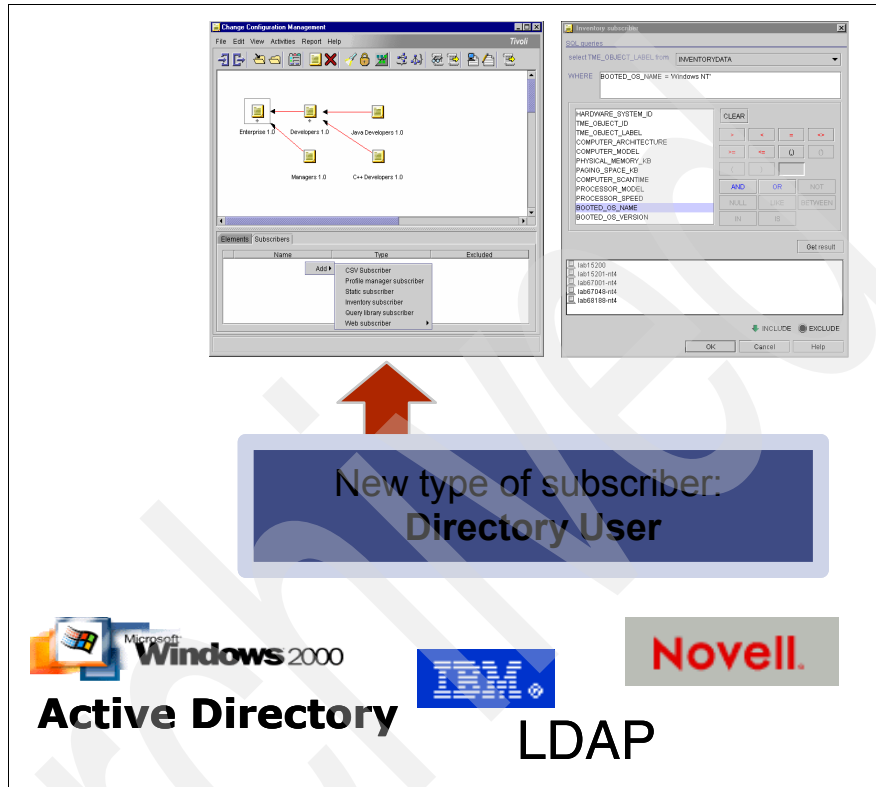


Figure 1-5 Enterprise Directory Query Facility

1.3.9 Data Moving

Data Moving is a Tivoli Configuration Manager component used to send, retrieve, and delete data from endpoint to endpoint or managed node without creating a software package.

1.3.10 Pristine Manager

Pristine Manager is a component of Tivoli Configuration Manager available with Version 4.2.1. Pristine Manager enables Tivoli Configuration Manager to manage machines that have no operating systems installed (bare-metal machines). It does not perform the real pristine set-up; it leverages third-party products.

Figure 1-6 on page 12 shows the relationship between the server, gateway, endpoint, and pristine machines. The sequence of operations are:

1. From the Activity Planner/Change Manager console, define the server and machine databases and create the operating system elements.
2. Create and synchronize the reference model to create the activity plan. The reference model and activity plan are created with information stored on the relational database management system (RDBMS) server. The plan that is generated must be run from the Activity Plan Monitor. The activity plan contains the pristine activity.
3. The Tivoli server distributes the pristine activity to the Remote Installation Service/Automated Deployment Services server on the endpoint for each pristine machine.
4. When a pristine machine boots, the Remote Installation Service/Automated Deployment Services server installs the operating system and a Tivoli management agent on that machine.
5. When the operating system and the Tivoli management agent have been installed on the pristine machine, the pristine machine logs on to the Tivoli gateway to notify the Tivoli server that the Pristine Manager has completed the configuration of the pristine machine.

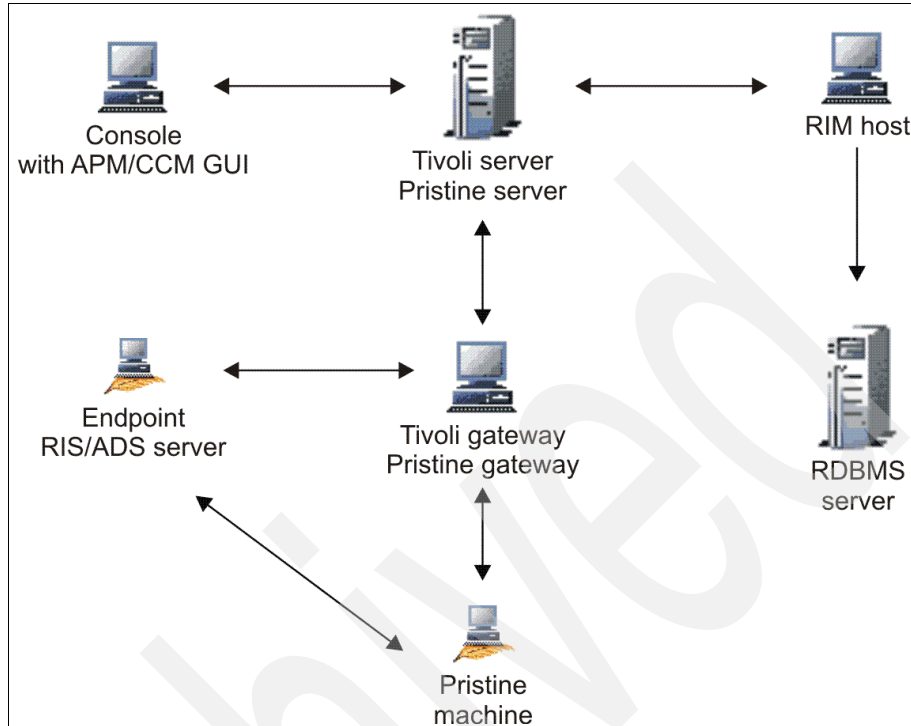


Figure 1-6 Pristine Manager

1.4 Understanding the IBM Tivoli Configuration Manager evolution

This section provides guidelines to understand the evolution of the IBM Tivoli Configuration Manager components. Understanding the product evolution will enable us to validate what patches or enhancements are available in each major release. Although this book's primary focus is IBM Tivoli Configuration Manager V4.2.2, we review generic guidelines, best practices, and hints and tips while using IBM Tivoli Configuration Manager Components V4.2.x as part of your overall IBM Tivoli Configuration Manager deployment process.

As part of your IBM Tivoli Configuration Manager deployment, it is important to understand how the product has evolved to validate what patches, fix packs, or enhancements are available on the installation or upgrade you are about to perform. Version upgrades are often called migrations. However, a move to a new version of Tivoli Configuration Manager does involve new code levels, because new features or enhancements are included. We recommend that you maintain

both a test and a preproduction Tivoli Configuration Manager environment. Your preproduction environment needs to replicate the current production Tivoli Configuration Manager software installed, while your test environment can be upgraded to later versions of Tivoli Configuration Manager.

1.4.1 A little bit of IBM Tivoli Configuration Manager history

In 2000, the introduction of Tivoli Software Distribution V4.0 and Tivoli Inventory V4.0 were based on the multiplexed distribution queuing mechanism. MDist 2 support for Software Distribution and Inventory Distribution control, Software Package Editor, Inventory enhancements (such as schema redesign and endpoint-initiated scans) were introduced with these versions.

In 2001, Tivoli Software Distribution V4.1 was released with the introduction of the Activity Planner, Change and Configuration Manager, which supports reference models (“desired state”), Data Moving Service, roaming/mobile support, pristine tool, and Web User Interface enhancements. In addition, in 2002, both products, Tivoli Software Distribution V4.1 and Tivoli Inventory V4.0, were marketed together as IBM Tivoli Configuration Manager V4.1.

IBM Tivoli Configuration Manager V4.2 was released in 2002. This version provided major enhancement to the Activity Planner to be able to support conditioning by targets and depots, tighter integration for Inventory and Software Distribution components, Change Manager enhancements, such as command line support, Web Gateway component, introduction to support Software Distribution pull mechanism from a browser, pervasive device support (Software Distribution and Inventory), and isolated inventory scans.

In 2003, IBM Tivoli Configuration Manager V4.2.1 released a new component called Pristine Manager to support pristine installations through preexecution boot environment (PXE) protocol plus enhancements in the Inventory and Software Distribution components (Software Package Editor, Data Moving, Activity Planner, and Change Manager).

In 2004, IBM Tivoli Configuration Manager V4.2.2 became available and introduced enhancements in the Tivoli Web Gateway (redesign of reporting results, Nokia 9500 device support), Activity Planner (checkpoint restart), and Data Moving.

IBM Tivoli Configuration Manager V4.2.3 was released in 2005, introducing a new component, patch management solution, and Nokia 9300 pervasive device support.

Figure 1-7 on page 14 shows the historical background of Tivoli Configuration Manager.

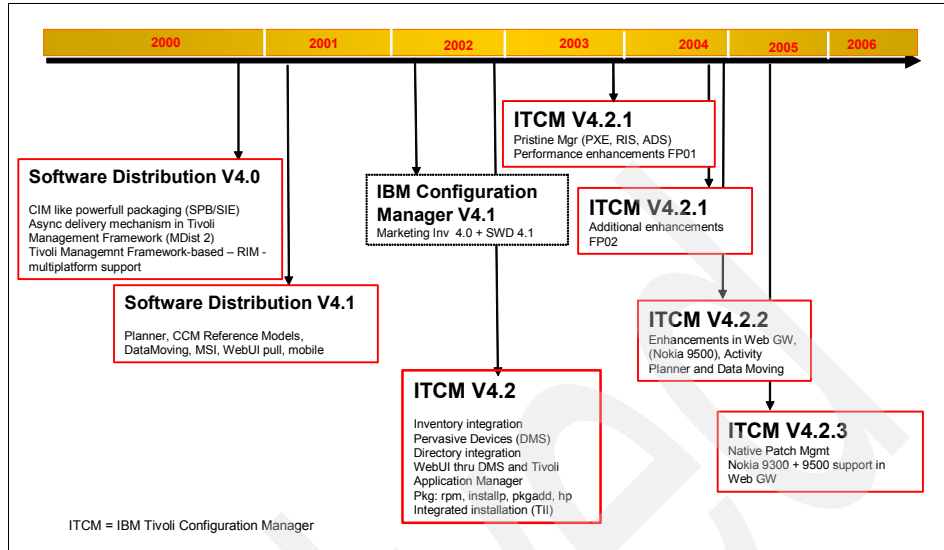


Figure 1-7 Tivoli Configuration Manager historical background

1.4.2 Tivoli Configuration Manager V4.2.x product-based releases

We now summarize the most recent levels of IBM Tivoli Configuration Manager included on each release starting with IBM Tivoli Configuration Manager V4.2. You need to understand what features are provided on each version release and the deployment path to be able to deploy a specific service. The *Release Notes* of each Tivoli Configuration Manager Product version document the new features and enhancements.

IBM Tivoli Configuration Manager V4.2

In this section, we see that IBM Tivoli Configuration Manager V4.2 was built on top of specific product levels of Software Distribution V4.1 and Inventory V4.0 or IBM Tivoli Configuration Manager V4.1.

Important: At the time that this book was written, the latest available fix pack was IBM Tivoli Configuration Manager, Version 4.2 Fix Pack 5. This fix pack contains all fixes from prior IBM Tivoli Configuration Manager V4.2 fix packs plus all fixes released as interim fixes. The list of components of Tivoli Configuration Manager V4.2 Fix Pack 5 is:

- ▶ 4.2-INV-FP05 Inventory, Version 4.2
- ▶ 4.2-INVGW-FP05 Inventory Gateway, Version 4.2
- ▶ 4.2-DistributionSRV-FP05 Software Distribution, Version 4.2
- ▶ 4.2-SWDFR-FP05 Software Distribution, Version 4.2 (fr) French language support
- ▶ 4.2-SWDGW-FP05 Software Distribution Gateway, Version 4.2
- ▶ 4.2-SWDJPS-FP05 Software Package Editor, Version 4.2
- ▶ 4.2-SWDEP-FP05 Software Package Editor for Endpoints, Version 4.2
- ▶ 4.2-APM-FP05 Activity Planner, Version 4.2
- ▶ 4.2-CCM-FP05 Change Manager, Version 4.2
- ▶ 4.2-TRM-FP05 Resource Manager, Version 4.2
- ▶ 4.2-TRMGW-FP05 Resource Manager Gateway, Version 4.2
- ▶ 4.2-WEB-FP05 Web Interface, Version 4.2
- ▶ 4.2-DQY-FP05 Enterprise Directory Query Facility, Version 4.2

Inventory V4.2

The following patches on Tivoli Inventory, Version 4.0, have been incorporated into the Inventory component of IBM Tivoli Configuration Manager, Version 4.2:

- ▶ 4.0-INV-0001 4.0-INV-0017
- ▶ 4.0-INV-0006 4.0-INV-0022
- ▶ 4.0-INV-0010 4.0-INV-0024
- ▶ 4.0-INV-0011 4.0-INV-FP02
- ▶ 4.0-INV-0012 4.0-INV-FP03
- ▶ 4.0-INV-0014

Tip: Tivoli Configuration Manager Inventory V4.2 schema was redesigned so that you need to plan how to migrate your information from the Inventory V4.0 tables to new tables that are created in Inventory V4.2. In particular, the table `sd_cm_status` from Inventory V4.0 changes its name to `sd_inst` in V4.2. This table keeps the information of your Software Distribution catalogs (the Inventory GUID stored in the field `computer_sys_id`, package name, package version, and status for each machine).

Software Distribution V4.2

The following patches on Tivoli Software Distribution, Version 4.1, have been incorporated into the Software Distribution component of IBM Tivoli Configuration Manager, Version 4.2:

- ▶ 4.1-COU-0001 4.1-COU-0032 4.1-COU-0071 4.1-COU-0127
- ▶ 4.1-COU-0002 4.1-COU-0033 4.1-COU-0078 4.1-COU-0128
- ▶ 4.1-COU-0005 4.1-COU-0034 4.1-COU-0079 4.1-COU-0138
- ▶ 4.1-COU-0006 4.1-COU-0035 4.1-COU-0080 4.1-COU-0139
- ▶ 4.1-COU-0007 4.1-COU-0037 4.1-COU-0096 4.1-COU-0151
- ▶ 4.1-COU-0008 4.1-COU-0045 4.1-COU-0097 4.1-COU-0152
- ▶ 4.1-COU-0011 4.1-COU-0046 4.1-COU-0104 4.1-COU-0153
- ▶ 4.1-COU-0016 4.1-COU-0063 4.1-COU-0105 4.1-COU-0154
- ▶ 4.1-COU-0019 4.1-COU-0064 4.1-COU-0107 4.1-COU-0161
- ▶ 4.1-COU-0020 4.1-COU-0065 4.1-COU-0108 4.1-COU-0162
- ▶ 4.1-COU-0023 4.1-COU-0066 4.1-COU-0117 4.1-COU-0163
- ▶ 4.1-COU-0024 4.1-COU-0068 4.1-COU-0118
- ▶ 4.1-COU-0030 4.1-COU-0069 4.1-COU-0119
- ▶ 4.1-COU-0031 4.1-COU-0070 4.1-COU-0120

Activity Planner V4.2

The following patches on the Activity Planner component of Tivoli Software Distribution, Version 4.1, have been incorporated into the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2:

- ▶ 4.1-COU-0003 4.1-COU-0073
- ▶ 4.1-COU-0004 4.1-COU-0083
- ▶ 4.1-COU-0012 4.1-COU-0108
- ▶ 4.1-COU-0013 4.1-COU-0109
- ▶ 4.1-COU-0025 4.1-COU-0121
- ▶ 4.1-COU-0026 4.1-COU-0122
- ▶ 4.1-COU-0041 4.1-COU-0155
- ▶ 4.1-COU-0042 4.1-COU-0156
- ▶ 4.1-COU-0072

Change Manager V4.2

The following patches on the Change Configuration Manager component of Tivoli Software Distribution, Version 4.1, have been incorporated into the Change Manager service of IBM Tivoli Configuration Manager, Version 4.2:

- ▶ 4.1-COU-0014 4.1-COU-076 4.1-COU-0015 4.1-COU-077
- ▶ 4.1-COU-0028 4.1-COU-0123 4.1-COU-0029 4.1-COU-0124
- ▶ 4.1-COU-0043 4.1-COU-0157 4.1-COU-0044 4.1-COU-0158

IBM Tivoli Configuration Manager V4.2.1

Important: While this book was being written, Tivoli Configuration Manager V4.2.1 Fix Pack FP03 was released, superseding all interim fixes and fix packs released previously for Tivoli Configuration Manager V4.2.1 (FP01, FP02, and F2P2). New features are supported with the release of this FP03, such as the Pocket PC Device Agent to enable persistent install after a hard reset or battery dies for the Tivoli Configuration Manager V4.2.1 Web Gateway component, so users avoid performing a Windows logon during critical distributions and defer time management for an end-user notification panel.

The list of components of Tivoli Configuration Manager V4.2.1 Fix Pack FP03 is:

- ▶ 4.2.1-INV-FP03 Inventory, Version 4.2.1
- ▶ 4.2.1-INVGW-FP03 Inventory Gateway, Version 4.2.1
- ▶ 4.2.1-SWDSRV-FP03 Software Distribution, Version 4.2.1
- ▶ 4.2.1-SWDFR-FP03 Software Distribution, Version 4.2.1 (fr) French language support
- ▶ 4.2.1-SWDGW-FP03 Software Distribution Gateway, Version 4.2.1
- ▶ 4.2.1-SWDJPS-FP03 Software Package Editor, Version 4.2.1
- ▶ 4.2.1-SWDEP-FP03 Software Package Editor for Endpoints, Version 4.2.1
- ▶ 4.2.1-APM-FP03 Activity Planner, Version 4.2.1
- ▶ 4.2.1-CCM-FP03 Change Manager, Version 4.2.1
- ▶ 4.2.1-TRM-FP03 Resource Manager, Version 4.2.1
- ▶ 4.2.1-TRMGW-FP03 Resource Manager Gateway, Version 4.2.1
- ▶ 4.2.1-WEB-FP03 Web Interface, Version 4.2.1
- ▶ 4.2.1-PMSRV-FP03 Pristine Manager Server 4.2.1
- ▶ 4.2.1-PMGW-FP03 Pristine Manager Gateway, Version 4.2.1
- ▶ 4.2.1-DQY-FP03 Enterprise Directory Query Facility, Version 4.2.1

We now review the components of IBM Tivoli configuration Manager V4.2.1 that were developed on top of IBM Tivoli Configuration Manager V4.2 FP01 and FP02.

Inventory V4.2.1

The following patch on Tivoli Inventory, Version 4.2, has been incorporated into the Inventory component of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-INV-FP01 4.2-INVGW-FP01

Software Distribution V4.2.1

The following patches on Tivoli Software Distribution, Version 4.2, have been incorporated into the Software Distribution component of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-SWD-FP01 4.2-SWDSRV-FP02
- ▶ 4.2-SWDGW-FP01 4.2-SWDGW-FP02
- ▶ 4.2-SWDJPS-FP01 4.2-SWDJPS-FP02

Activity Planner V4.2.1

The following patch on the Activity Planner component of Tivoli Software Distribution, Version 4.2, has been incorporated into the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-APM-FP01 4.2-APM-FP02

Change Manager V4.2.1

The following patch on the Change Configuration Manager component of Tivoli Software Distribution, Version 4.2, has been incorporated into the Change Manager service of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-CCM-FP01 4.2-CCM-FP02

Resource Manager V4.2.1

The following patches on the Resource Manager component of Tivoli Software Distribution, Version 4.2, have been incorporated into the Resource Manager service of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-TRM-FP01 4.2-TRM-FP02
- ▶ 4.2-TRMGW-FP01 4.2-TRMGW-FP02

Web Interface V4.2.1

The following patch on the Web Interface component of Tivoli Software Distribution, Version 4.2, has been incorporated into the Web Interface service of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-WEB-FP01 4.2-WEB-FP02

Enterprise Query Directory V4.2.1

The following patch on the Enterprise Directory Query component of Tivoli Software Distribution, Version 4.2, has been incorporated into the Enterprise Directory Query service of IBM Tivoli Configuration Manager, Version 4.2.1:

- ▶ 4.2-QDY-FP01 4.2-QDY-FP02

IBM Tivoli Configuration Manager V4.2.2

Important: At the time that this book was written, the latest available Interim Fix was 4.2.2-SWD-0001, which included the following patches:

- ▶ 4.2.2-APM-0001 Activity Planner, Version 4.2.2
- ▶ 4.2.2-SWDSRV-0001 Software Distribution, Version 4.2.2
- ▶ 4.2.2-SWDGW-0001 Software Distribution Gateway, Version 4.2.2
- ▶ 4.2.2-SWDEP-0001 Software Distribution Endpoint, Version 4.2.2

We now review that IBM Tivoli Configuration Manager V4.2.2 was built on top of IBM Tivoli Configuration Manager V4.2.1 Fix Pack FP01.

Inventory

The follow patch on Tivoli Inventory, Version 4.2.1, has been incorporated into the Inventory component of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-INV-FP01 4.2.1-INVGW-FP01

Software Distribution

The following patches on Tivoli Software Distribution, Version 4.2.1, have been incorporated into the Software Distribution component of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-SWD-FP01 4.2.1-SWDSRV-FP01
- ▶ 4.2.1-SWDGW-FP01 4.2.1-SWDJPS-FP01

Activity Planner V4.2.2

The following patch on the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2.1, has been incorporated into the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-APM-FP01

Change Manager V4.2.2

The following patch on the Change Manager service of IBM Tivoli Configuration Manager, Version 4.2.1, has been incorporated into the Change Manager service of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-CCM-FP01

Resource Manager V4.2.2

The following patch on the Resource Manager service of IBM Tivoli Configuration Manager, Version 4.2.1, has been incorporated into the Resource Manager service of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-TRM-FP01

Web Interface V4.2.2

The following patch on the Web Interface service of IBM Tivoli Configuration Manager, Version 4.2.1, has been incorporated into the Web Interface service of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-WEB-FP01

Enterprise Directory Query

The following patch on the Enterprise Directory Query service of IBM Tivoli Configuration Manager, Version 4.2.1, has been incorporated into the Enterprise Directory Query service of IBM Tivoli Configuration Manager, Version 4.2.2:

- ▶ 4.2.1-QDY-FP01

IBM Tivoli Configuration Manager V4.2.3

IBM Tivoli Configuration Manager V4.2.3 was built on top of IBM Tivoli Configuration Manager Version 4.2.2.

Software Distribution

The following patches on Tivoli Software Distribution, Version 4.2.2, have been incorporated into the Software Distribution component of IBM Tivoli Configuration Manager, Version 4.2.3:

- ▶ 4.2.2-SWDSRV-0001 4.2.2-SWDGW-0001
- ▶ 4.2.2-SWDEP-0001

Activity Planner

The following patch on the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2.2, has been incorporated into the Activity Planner service of IBM Tivoli Configuration Manager, Version 4.2.3:

- ▶ 4.2.2-APM-0001

1.4.3 Tivoli Configuration Manager releases evolution summary

Table 1-1 presents a summary of the IBM Tivoli Configuration Manager V4.2.x evolution in terms of fix packs and product releases.

Table 1-1 Tivoli Configuration Manager releases evolution summary

Tivoli Configuration Manager release built on top of	Tivoli Configuration Manager release
4.1 plus specific patches (as described in the previous sections)	4.2
4.2 FP01 + enhancements + new features	4.2.1
4.2.1 FP01 + enhancements	4.2.2
4.2.2 -SWD-001 + new features	4.2.3

In Figure 1-8, we can see how IBM Tivoli Configuration Manager has evolved. Note that Software Distribution V4.0 cannot be upgraded to Tivoli Configuration Manager Software Distribution component V4.2 or later. You need to upgrade first to Tivoli Software Distribution V4.1.

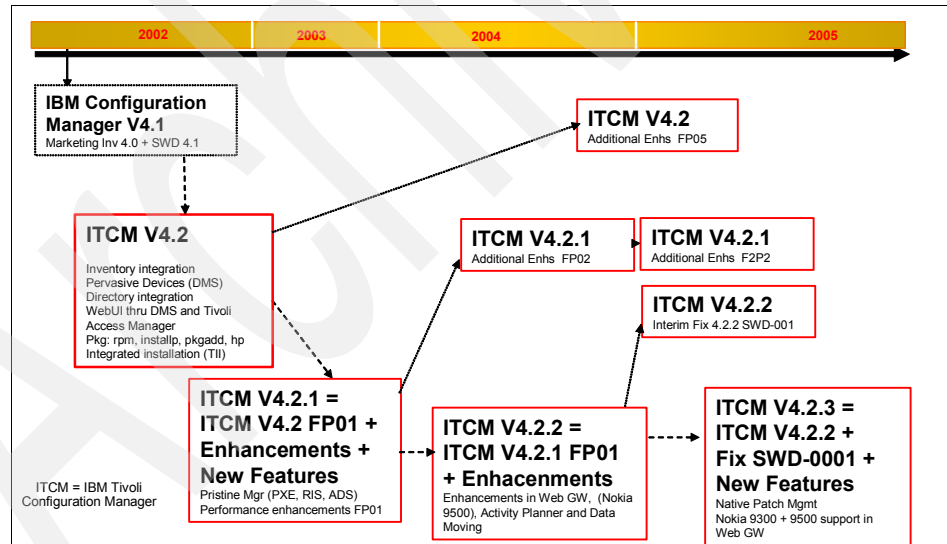


Figure 1-8 IBM Tivoli Configuration Manager releases and fix packs

1.4.4 Tivoli Management Framework

IBM Tivoli Management Framework is the software infrastructure for many Tivoli management products. Tivoli Management Framework provides services that are

used by the installed Tivoli Enterprise™ products such as IBM Tivoli Configuration Manager. Therefore, we also need to understand how to apply Tivoli Management Framework fix packs or patches. We summarize the most recent levels of Tivoli Management Framework required to deploy and install each release of IBM Tivoli Configuration Manager starting with Version 4.2.

Note: We recommend that you carefully read the Field Guide *IBM Tivoli Configuration Manager Upgrade, Version 4.2.1: Guidelines for a Smooth Upgrade*, where more scenarios and interoperability issues about Framework upgrades are described, available at:

http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html

In the following sections, we review upgrades from Tivoli Management Framework V3.7.1 and Tivoli Management Framework V4.1.

Tivoli Management Framework V4.1

In order to install IBM Tivoli Configuration Manager V4.2, you need to either upgrade Tivoli Management Framework V3.7.1, or Tivoli Management Framework rev. B to Tivoli Management Framework V4.1, or perform a fresh installation of Tivoli Management Framework V4.1.

Tivoli Management Framework, Version 4.1, includes all Tivoli Management Framework patches through 3.7.1-TMF-0075.

Important: At the time that this book was written, the latest fix pack available was 4.1-TMF-FP04. Be aware that you also need to upgrade the Tivoli Scalable Collection Service V4.1 to 4.1-CLL-FP01 and 4.1-CLL-0007 for the Inventory Collectors Upgrade of Tivoli Configuration Manager V4.2 Inventory component.

Tivoli Management Framework V4.1.1

Tivoli Management Framework is required for IBM Tivoli Configuration Manager Versions 4.2.1, 4.2.2, and 4.2.3. Tivoli Management Framework, Version 4.1.1, includes all Tivoli server, managed node, and gateway patches through 3.7.1-TMF-0117 and endpoint patches through 3.7.1-LCF-013 and 4.1-LCF-022.

Tivoli Management Framework, Version 3.7.1, can be upgraded to Tivoli Management Framework, Version 4.1.1, using the *Tivoli Management Framework Upgrade Version 3.7 to Version 4.1.1* CD and Version 4.1 can be upgraded to Tivoli Management Framework, Version 4.1.1, using the *Tivoli Management Framework Upgrade Version 4.1 to Version 4.1.1* CD. You can perform new installations of Tivoli Management Framework, Version 4.1.1, using the *Tivoli Management Framework, Version 4.1.1 (1 of 2)* CD.

Important: At the time that this book was written, the latest patch available was 4.1.1-TMF-0038, 4.1.1-TMF-0039, 4.1.1-TMF-0044. Note that you also need to upgrade the Tivoli Scalable Collection Service V4.1.1 to 4.1.1-CLL-FP01 for the Inventory Collectors Upgrade of Tivoli Configuration Manager V4.2.1 Inventory component. Tivoli Configuration Manager V4.2.1 Fix Pack FP03 was tested on top of the 4.1.1-TMF-0039 and 4.1.1-TMF-0044 patches.

Archived

Archived

Installation planning

This chapter describes the planning that needs to be done before implementing IBM Tivoli Configuration Manager across an enterprise. We discuss the expertise required, the requirements of the systems to be installed, some other prerequisites that will be part of the planning phase, and the backup strategies that should go along with the implementation of Tivoli Configuration Manager.

We discuss the following topics in this chapter:

- ▶ Expertise required
- ▶ Creating a deployment plan for Tivoli Configuration Manager
- ▶ How to deploy Tivoli Configuration Manager components
- ▶ Required roles for installation
- ▶ RDBMS considerations
- ▶ RIM considerations
- ▶ General preinstallation checks, hints, and tips
- ▶ Installation methods
- ▶ Overview of the integrated installation
- ▶ Server installation
- ▶ Desktop installation
- ▶ Web Gateway installation

- ▶ Uninstallation of IBM Tivoli Configuration Manager
- ▶ Architectural considerations for Tivoli regions
- ▶ Backup strategies

2.1 Expertise required

Installing Tivoli Configuration Manager requires expertise on several areas. In this section, we describe some of the general expertise required to do the tasks listed. We take each phase of the installation down to the operating system level. In most of the cases, it is not necessary for one person to have all the expertise, but if the person designated as the Tivoli administrator possesses some knowledge of these different products, it will be easier to deploy and maintain the product.

It is not unusual to have different people with expertise in these technologies all working together. The workload of the Tivoli administrator will be significantly more than the workload of, for example, the DB2® administrator in most cases.

2.1.1 DB2 administrator

The DB2 administrator must possess an understanding of how databases work and preferably with the IBM DB2 product. If there is a database administrator in the organization, they need to possess the needed knowledge to perform most of the work to be done with the DB2 database. A thorough knowledge of DB2 is not needed in most instances, but will greatly enhance the Tivoli Configuration Manager experience.

2.1.2 Operating systems administrator

The UNIX administrator should possess advanced knowledge of administering a UNIX server. The UNIX administrator will be called upon at times to upgrade the operating system software and any other software that is resident on the server. The UNIX administrator must also be able to add and delete users and be able to change permissions on directories. There is also a need to be able to do some debugging at the operating system level and a need to know about TCP/IP, networks, Domain Name System (DNS), host files, file systems, cron jobs, ports, adding additional space for a growing processes, and any other assignments that a UNIX administrator would do in the normal course of business.

The Microsoft Windows administrator should possess the advanced knowledge of administering Microsoft Windows computers, both server classes, such as Microsoft Windows 2000 Server servers or Microsoft Windows Server 2003 servers, and workstation classes such as Microsoft Windows XP. This administrator will be called upon to keep the operating system software updated and secure. The administrator must be able to give certain security rights to the Tivoli administrator that will allow the software to operate normally.

See Appendix A, “Microsoft security checklist” on page 351 for requirements associated with users and password and security levels regarding the Windows platform. This appendix also describes the requirements for Microsoft Windows computers.

2.1.3 Network administrator

A good, stable network is essential for the operation of IBM Tivoli Configuration Manager. For this reason, you need to make sure that you have a network administrator on the team implementing Tivoli Configuration Manager.

The use of Domain Name System (DNS) servers or the use of host files is recommended for name resolution of Tivoli servers and managed nodes/gateway communications. You can use Dynamic Host Configuration Protocol (DHCP) for the endpoints. We do not recommend the use of host files for name resolution between Tivoli management regions (TMRs) and managed nodes/gateways, because it becomes a very large task to keep these files updated all the time. We recommend the use of DNS, because there is a central repository where this information is kept and it can be changed easily (the ability to change the host file on every computer is a daunting task at best).

The computers must be able to resolve name and IP addresses in a two-way configuration. Tivoli Configuration Manager can successfully run in environments where network address translation (NAT) protocol is implemented. In addition, Tivoli Firewall Toolkit is available for “tunneling” through firewalls. Tivoli Configuration Manager also employs the Bulk Data Transfer (BDT) protocol, which enables all the Tivoli network traffic to be handled through ports 9401 and 9402.

Being able to quickly recognize a network communication problem will further enhance your ability to administer your Tivoli management region more effectively.

2.1.4 Other products expertise

You can integrate Tivoli Configuration Manager with many other IBM and non-IBM products, but the following products are particularly important in terms of integration (for example, WebSphere and Access Manager are prerequisites for the Web User Interface components). If these products are used in the Tivoli Configuration Manager deployment, the team implementing Tivoli Configuration Manager should have expertise on these as well.

LDAP

Lightweight Directory Access Protocol (LDAP) is used to store relatively static information. “Write once use many times” describes the best use of LDAP. LDAP is structured as a directory that is optimized for lookups. LDAP, when thought of as a tree structure, is very useful in fulfilling the requirements for optimization. Each entry in the LDAP tree has one or more object classes, and every object class has attributes stored in name-value pairs. LDAP attributes, such as `cn=root`, can also specify the type of operations that LDAP supports. An LDAP schema can be relatively simple or moderately complex. In its most simple form, it is just a linear list of users. Any entry can be located in the directory tree by using the relative distinguished name or DN. This guide will use this form of LDAP server.

For an example, if there are employees in the Tivoli deployment department that need access rights, we would set up the department and distinguished names (DN) as shown in Table 2-1.

Table 2-1 LDAP table

Employee name	DN in LDAP
Johnny Smith	<code>cn=Johnny Smith,ou=employee,dc=IBM,dc=com</code>
Sally Jones	<code>cn=Sally Jones,ou=employee,dc=IBM,dc=com</code>
Debra Winslow	<code>cn=Debra Winslow,ou=employee,dc=IBM,dc=com</code>
Sam Rogers	<code>cn=Sam Rogers,ou=employee,dc=IBM,dc=com</code>
Mary Lucinda	<code>cn=Mary Lucinda,ou=employee,dc=IBM,dc=com</code>
Henry Thomas	<code>cn=Henry Thomas,ou=employee,dc=IBM,dc=com</code>

As you can see, the LDAP is nothing more than a lookup server of names, addresses, and other information that might be useful in the normal course of business.

By setting up the LDAP server for use with Tivoli Configuration Manager V4.2.2, the job of authenticating the user is now a little more secure and handled from a central repository, rather than on each individual computer.

WebSphere

Tivoli Management Framework has the ability to be integrated with WebSphere. IBM WebSphere is the leading software platform for on demand businesses. This software family includes more than 150 products that can help businesses become an On Demand Business, regardless of their industries.

For Tivoli Configuration Manager, WebSphere technology is used for the Web Gateway component.

Tivoli Data Warehouse

Tivoli Data Warehouse functionality complements the Tivoli Management Framework. You can use Tivoli Data Warehouse to review historical information using reports provided with the software or you can generate your own reports.

IBM Tivoli Access Manager

IBM Tivoli has three products that integrate with Access Manager: Access Manager for e-business, Access Manager for Operating Systems, and Access Manager for Business Integration. Access Manager started with a product called Identity Manager and matured into the security product portfolio.

The total security solution is found with the integration of Access Manager and Tivoli Configuration Manager.

2.2 Creating a deployment plan for Tivoli Configuration Manager

Creating a deployment plan is essential to creating and installing a configuration management environment. The basic considerations for creating a deployment plan for a Tivoli environment are provided in *Tivoli Management Framework: Planning for Deployment Guide*, GC32-0803. At a minimum, you need to gather the following information before installing any software:

- ▶ Base hardware and software requirements for Tivoli Management Framework and IBM Tivoli Configuration Manager. This information is provided in *Tivoli Management Framework Release Notes*, G111-0890, and *IBM Tivoli Configuration Manager Release Notes*, G111-0926.
- ▶ Whether the computer systems in your distributed network can support this new software, whether these systems can be upgraded to meet your business needs, or whether new systems need to be obtained.
- ▶ Which IBM Tivoli Configuration Manager components to install on which computer systems in your distributed network to support your business needs and whether they have additional third-party software requirements. This information is provided in *IBM Tivoli Configuration Manager Release Notes*, G111-0926, and *Introducing IBM Tivoli Configuration Manager*, GC23-4703.

For each system where you plan to install components of IBM Tivoli Configuration Manager, also provide the following information:

- ▶ Host name

- ▶ Operating system
- ▶ Available memory and available disk space
- ▶ Which components of IBM Tivoli Configuration Manager to install

2.3 How to deploy Tivoli Configuration Manager components

There are many software components included in Configuration Manager. When you plan your deployment, you will decide which components you need and where each component will be used in your Tivoli environment.

2.3.1 Distributed Tivoli Configuration Manager components

Certain Tivoli Configuration Manager components will be installed on either the Tivoli server or managed node; some will be installed on both. Specific components will be installed on gateway systems, while other components will require installation on endpoints.

Of course, because the same system can be a Tivoli server, managed node, gateway, and endpoint, all of the components can be installed on the Tivoli server, with other selected components being installed on various managed nodes and endpoints throughout your Tivoli environment.

2.3.2 Tivoli server configuration

Some components must be installed on the Tivoli server, even if you are not planning to use these components on this system. *IBM Tivoli Configuration Manager Version 4.2.2: Planning and Installation Guide, GC23-4702*, enumerates the components that must be installed on the Tivoli server before any other Tivoli Configuration Manager components can be installed on other systems in the Tivoli environment.

Here are the Tivoli Configuration Manager components that must be installed on the Tivoli server before deploying other components in your Tivoli environment (of course, if you are not going to use the component in your Tivoli environment, you do not need to install it on your Tivoli server):

- ▶ Scalable Collection Service¹ (patch)
- ▶ Inventory
- ▶ Software Distribution
- ▶ Activity Planner

¹ You must install the Scalable Collection Service patch before you install the Inventory component. This patch is not required for the other components.

- ▶ Change Manager
- ▶ Pristine Manager
- ▶ Enterprise Directory Query Facility
- ▶ Resource Manager
- ▶ Web Infrastructure

2.3.3 Components for managed nodes

Here are the Tivoli Configuration Manager components that can be installed on managed nodes.

Install these components on managed nodes if you plan on running an administrative interface or CLI commands from the managed node (and in the case of Software Distribution, on managed nodes that will be acting as source hosts):

- ▶ Scalable Collection Service² (patch)
- ▶ Inventory
- ▶ Software Distribution
- ▶ Activity Planner
- ▶ Change Manager

2.3.4 Components for gateways

There are additional Tivoli Configuration Manager components that must be installed on Tivoli gateways if they are to participate in inventory scans or software distributions.

Install the Scalable Collection Service patch on each gateway that:

- ▶ Connects to endpoints to be scanned by the Inventory component
- ▶ Is a repeater that will act as a collector
- ▶ Is a gateway that connects to the Web Gateway components

Install the Inventory Gateway component on each gateway that will:

- ▶ Distribute Inventory profiles to endpoints
- ▶ Recognize Inventory methods and download these methods to endpoints
- ▶ Run methods to perform requested Inventory actions

Install the Software Distribution Gateway on each gateway that will:

- ▶ Recognize Software Distribution methods and download these methods to endpoints.

² You must install the Scalable Collection Service patch before you install the Inventory component. This patch is not required for the other components.

- ▶ Run methods to perform the requested Software Distribution operation

Install Pristine Manager Gateway component on each gateway that:

- ▶ The Pristine machine will attach
- ▶ The Remote Installation Service or Automatic Deployment Services endpoints are attached

2.3.5 Components for endpoints

These Tivoli Configuration Manager components must be installed on endpoints:

- ▶ Tivoli Desktop for Windows.

This includes interfaces for Activity Planner, Change Manager, Inventory, and Software Distribution.

Note: You must install the Tivoli Desktop for Windows from the Configuration Manager media (not the Framework media) in order to install the additional software for the Configuration Manager component interfaces. Or, if you already installed Desktop for Windows from the Framework CD, you can add the Configuration Manager components by running the Desktop for Windows setup program from the Configuration Manager CD and choosing a customized installation.

- ▶ Software Package Editor.
- ▶ Software Distribution Pristine Tool (Microsoft Windows and OS/2 only).
- ▶ Web Gateway (despite its name, this is not installed on a Tivoli gateway).
- ▶ Web Interface (including Inventory and Software Distribution plug-ins).

2.4 Required roles for installation

Table 2-2 lists the required roles for installing Tivoli Configuration Manager.

Table 2-2 Required roles for installing Tivoli Configuration Manager

Name of the operation	Required role
Install the installation images directly from the CD images, using the InstallShield wizard	For UNIX and Linux: Root access For Windows: A member of the Administrators group
Install from the Tivoli desktop or command line	install_product or super Tivoli Framework roles in a Tivoli region

Name of the operation	Required role
Install using Tivoli Software Installation Service	User role plus one of the following Tivoli administrator roles in a Tivoli region: super, senior, or install_product

2.5 RDBMS considerations

Tivoli Configuration Manager stores data in an external relational database management system (RDBMS). The RDBMS server can be part of your Tivoli environment, but that is not necessary as long as:

- ▶ There is TCP/IP connectivity between the RDBMS server and at least one managed node system in the Tivoli environment.
- ▶ The system or systems in the Tivoli environment that will communicate with the RDBMS system have client software for the RDBMS system installed and configured to connect to the RDBMS system.

During the installation of Tivoli Configuration Manager, you can choose to create five separate databases, or you can decide to create one database, `cm_db`, which will contain tables for each of the Configuration Manager components (Figure 2-1, “Databases created by the installation program” on page 34).

Component	Database
Activity Planner	planner (or <code>cm_db</code>)
Change Manager	ccm_db (or <code>cm_db</code>)
Distribution Status	mdist2 (or <code>cm_db</code>)
Inventory	Invtiv (or <code>cm_db</code>)
Pristine Manager	pristine (or <code>cm_db</code>)
Resource Manager	Invtiv (or <code>cm_db</code>)

Figure 2-1 Databases created by the installation program

These databases, known as repositories, are created in the RDBMS by running SQL admin scripts. Tivoli provides sample SQL scripts in the `FRESH/SQL/admin` directory of the Configuration Manager installation CD. These scripts create the various databases required by Configuration Manager. The scripts might not fit your production environment as-is; make sure to review these scripts and edit them to meet your database requirements.

The values used in the SQL admin scripts are default values. The values can be changed by editing the SQL admin script files.

If you are using the integrated server installation program, you can choose to have the installation program create the configuration repository; in this case, you do not need to explicitly run these SQL scripts.

Depending on the RDBMS product, you might need to do some additional setup outside of these scripts, such as create and catalog the database or create the operating system accounts that the RDBMS will use. This is true whether you run the admin scripts manually, or create the configuration repository using the integrated server installation program. Check the contents of the SQL admin scripts to determine what steps must be done before creating the repository. Example 2-1 shows the `cm_db2_admin.sql` script, which is used if you opt for creating one database (`cm_db`). Note the `PRECONDITION` statements. Also note that tablespaces are created by the script.

Example 2-1 The db2 admin script

```
-- cm_db2_admin.sql
--
-- PRECONDITION: The default 'cm_db' database is created with the statement
-- 'create database cm_db', then catalogued on the client with the
-- statement
-- 'catalog database cm_db at node <server_node_name>'.
-- The users invtiv, planner, tivoli and mdstatus are already created
-- using the operating system user manager utility.
-- This script is then run as the database administrator.
--
-- for single server, cm_db is the primary database with tablespace cm_ts
CREATE REGULAR TABLESPACE cm_ts
  MANAGED BY DATABASE
  USING (FILE 'cm_ts.dat' 1408M)
  EXTENTSIZE 16;

CREATE TEMPORARY TABLESPACE cm_temp_ts
  MANAGED BY DATABASE
  USING (FILE 'cm_temp_ts.dat' 2500)
  EXTENTSIZE 16;

-- logfile size in in 4k pages. Total log space is 176MB
UPDATE DATABASE CONFIGURATION FOR cm_db USING LOGFILSIZ      4096;
UPDATE DATABASE CONFIGURATION FOR cm_db USING LOGPRIMARY     5;
UPDATE DATABASE CONFIGURATION FOR cm_db USING LOGSECOND      5;

GRANT  CREATETAB, CONNECT ON DATABASE TO USER invtiv;
```

```
GRANT USE OF TABLESPACE cm_ts TO USER invtiv;  
  
GRANT CREATETAB, CONNECT ON DATABASE TO USER planner;  
GRANT USE OF TABLESPACE cm_ts TO USER planner;  
  
GRANT CREATETAB, CONNECT ON DATABASE TO USER tivoli;  
GRANT USE OF TABLESPACE cm_ts TO USER tivoli;  
  
GRANT CREATETAB, CONNECT ON DATABASE TO USER mdstatus;  
GRANT USE OF TABLESPACE cm_ts TO USER mdstatus;
```

Note: You should consider tuning your database. This is not a simple task. You need to know what you are doing and why you are doing it before changing any database parameters. Each environment is different and needs special consideration. The first thing you need to do is understand where the bottleneck is. You need to get help from your database administrator when you try to tune your Inventory database (or any other Tivoli database) for best performance. Hints about tuning can be found in the IBM Redbook *All About IBM Tivoli Configuration Manager 4.2*, SG24-6612.

2.6 RIM considerations

A RDBMS Interface Module (RIM) host is a managed node where the RIM object is created. RIM objects are created during installation. When deciding which managed nodes are to be RIM hosts, consider the following requirements:

- ▶ The managed node must be local to the Tivoli region.
- ▶ The managed node must be preconfigured with the RDBMS client or server software.

Notes: Do not install the RDBMS server software on the RIM host unless this machine is designated solely for RDBMS usage and no other Tivoli operations. When you add the number of transactions performed on the RDBMS server to those performed on the RIM host, the number of overall transactions might impact the optimal performance of your Tivoli environment by exceeding network throughput.

RIM is installed as a Framework component. There is no separate installation for RIM.

Figure 2-2 on page 37 shows the RIM objects created for Tivoli Configuration Manager by the installation program, along with their corresponding database and the software components that use it.

Component	Database	RIM object
Activity Planner	planner (or cm_db)	planner
Change Manager	ccm_db (or cm_db)	ccm
Distribution Status	mdist2 (or cm_db)	mdist2
Inventory	Invtiv (or cm_db)	invdh_1 inv_query
Pristine Manager	pristine (or cm_db)	pristine
Resource Manager	Invtiv (or cm_db)	trm (only if inv_query is not in local Tivoli management region)

Figure 2-2 RIM objects created by the installation program

Although RIM objects are created during installation, you can create additional RIM objects using the **wcrtim** command, or by moving a RIM object from one managed node to another using the **wmvrिम** command.

You can also change the database information for a given RIM object using the **wsetrim** command.

The syntax for the **wsetrim** command is:

```
wsetrim [-n name] [-d database] [-u user] [-H db_home] [-s server_id] [-I
instance_home] [-t instance_name] [-a application_label] [-m max_connections]
rim_name
```

Where:

- ▶ **-a application_label** changes the application label for the RIM object.
- ▶ **-d database** changes the name of the database or data source to which the RIM object connects.
- ▶ **-H db_home** changes the path to the database home directory.
- ▶ **-I instance_home** (for DB2 only) changes the path to the DB2 instance for the specified RIM object.
- ▶ **-m max_connections** changes the maximum number of connections between the RIM object and the RDBMS.
- ▶ **-n name** changes the name of the RIM object to name.

- ▶ `-s server_id` changes the server ID for the database.
- ▶ `-t instance_name` (for DB2 only) changes the name of the DB2 instance for the specified RIM object.
- ▶ `-u user` changes the name of the database user that the RIM object uses.

To change the vendor for a RIM object, you must delete the object using the `wdel` command and re-create it using the `wcrtrim` command.

Note: Vendor specification specifies the vendor of the RDBMS you are using when creating the RIM object (one entry for each RDBMS system that is supported by the Tivoli RIM component). Valid entries are:

- ▶ DB2
- ▶ Oracle
- ▶ Sybase
- ▶ Informix®
- ▶ MS_SQL

Note that Microsoft SQL is supported on Windows systems only.

To change the managed node for a RIM object, you can either move the RIM object using the `wmvrim` command or delete and re-create the RIM object. To change the label for a RIM object, you can either use the `wsetrim -n` command or delete and re-create the RIM object. Also, you cannot set the database password for an (RIM) object using the `wsetrim` command. You can use the `wsetrimpw` command for this purpose.

The following example changes the database ID to inventory, the database user to tivoli, the database home directory to /ORACLE, and the database server ID to invdb2 for the inventory RIM object:

```
wsetrim -d inventory -u tivoli -H /ORACLE \ -s invdb2 inventory
```

2.7 General preinstallation checks, hints, and tips

After creating the deployment plan, you must do a number of things before installing IBM Tivoli Configuration Manager V4.2.2:

- ▶ Read the *Tivoli Management Framework Release Notes*, G111-0890, and *IBM Tivoli Configuration Manager Release Notes*, G111-0926.
- ▶ Back up your Tivoli database (as well as perform any normal systems backup).

- ▶ You need to have super or install_product roles for installing Tivoli Configuration Manager V4.2.2 from the Tivoli desktop or the command line.
- ▶ Do not use the C shell for installing on UNIX systems.
- ▶ Do not try to install across TMR boundaries. Always install applications in the local TMR.
- ▶ 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 box opens, the check box is not selected by default. This does not apply to Integrated Server Install.
- ▶ On Linux systems, remote access is often disabled by default. Make sure that you enable it before the install.

2.7.1 UNIX

The install process performs some space checking after the installation starts, 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 might 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 the /var and /usr directories.

There are a number of reasons why you might want to set up specific Tivoli file systems:

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

The 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 or the temporary directory selected during the integrated installation.

2.7.2 Microsoft Windows systems on Intel 486 or Pentium systems

IBM Tivoli Configuration Manager V4.2.2 supports Microsoft Windows NT® 4.0 with Service Pack 5 or later, Windows XP Professional, Windows 2000 Server, and Windows Server 2003.

The drive where you want to install Tivoli must be formatted with NTFS. You can check this from the My Computer window. Right-click the desired drive icon and select **Properties**. The general page includes the file system type. You can also check this from a 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 online help for more information.

Tivoli files for Windows are, by default, stored under the \Tivoli directory on the root of the selected drive. Tivoli also writes install and other log files to %DBDIR%\tmp. You should verify through the Control Panel system applet that you have sufficient swap space defined.

2.8 Installation methods

You can install and upgrade the components of IBM Tivoli Configuration Manager using any of the following different installation mechanisms:

- ▶ Using the integrated installation, which creates a new Tivoli server and installs the appropriate IBM Tivoli Configuration Manager components or upgrades the entire Tivoli management region (Tivoli region) using activity plans.
- ▶ Using Tivoli Software Installation Service, where you select which products to install on which machines.
- ▶ Using the Tivoli desktop, where you select which product and patches to install on which machine.
- ▶ Using the **winstall** and **wpatch** commands provided by Tivoli Management Framework, where you specify which products and patches to install on which machine.
- ▶ Using the software package blocks. Before installing images from software package blocks, the Tivoli environment must be created and Tivoli Configuration Manager must be fully deployed.

We focus on the integrated installation.

2.9 Overview of the integrated installation

InstallShield Multi-Platform (ISMP) is part of the Tivoli Install Imperative and the IBM install strategy to achieve two major goals:

- ▶ Consistent install
- ▶ Simplified maintenance

The first principle helps achieve the goals of Tivoli by providing the customer with a similar installation experience for each Tivoli product. The second principle enables customers to apply maintenance (upgrades) to Tivoli products in a consistent and simplified way.

For IBM Tivoli Configuration Manager V4.2.2, ISMP is being used in the following scenarios:

- ▶ Server installation
- ▶ Desktop installation
- ▶ Web Gateway installation

2.10 Server installation

The server installation scenario starts from CD 5 and should be used if:

- ▶ Tivoli Management Framework is not installed.
- ▶ Tivoli Management Framework exists at the 4.1 level but has a subset of Configuration Manager V4.2.2 applications.

If current installation is not in one of these conditions, the installation is stopped by the installation program.

Note: Because the server install program assumes no previous Tivoli environment, your RIM host must be the Tivoli server (because there are no other systems in the region yet). Move the RIM object from the Tivoli server to another more suitable managed node after you have your Tivoli environment established.

There are two installation types with the server installation:

- ▶ Typical
- ▶ Custom

2.10.1 Typical server installation

The advantage of the typical server install is that you not have to specify as many details during the installation when using this installation option.

When using this installation, the following components are installed, configured, or created:

- ▶ Tivoli Management Framework.
- ▶ To support a single server installation, a Tivoli gateway, called *hostname-gw* is also created on this machine. This gateway is automatically configured as a repeater. The installation of Tivoli Management Framework creates the Tivoli server.
- ▶ Resource Manager and Resource Manager Gateway.
- ▶ Enterprise Directory Query Facility with the default directory context as *directory*.
- ▶ Scalable Collection Service. Scalable Collection Service is considered part of Tivoli Management Framework and is used to collect inventory scan results.
- ▶ Distribution Status console. The Distribution Status console tracks software distributions and other profile distributions. The installation of the Distribution Status console requires the following Java™ components (which are provided by Tivoli Management Framework):
 - Java for Tivoli
 - Java RDBMS Interface Module
 - Java Client Framework for Tivoli

These Java components are used by several of the other IBM Tivoli Configuration Manager components.

The installation of the Distribution Status console creates the *mdist2* RIM object.

- ▶ Activity Planner. This installation creates the Planner RIM object. This RIM object can be on the Tivoli server.
- ▶ Change Manager. This installation creates the *ccm* RIM object.
- ▶ Inventory and Inventory Gateway. The installation of the Inventory component creates the *inv_query* and *invdh_1* RIM objects.
- ▶ Software Distribution and Software Distribution Gateway.
- ▶ Software Package Editor.

This installation program can be used on all platforms supported as a Tivoli server.

2.10.2 Custom server installation

In the custom installation, you have more options (but the installation is more complex). Use the custom server install if you want to:

- ▶ Select components to install.
- ▶ Install additional languages.
- ▶ Choose the type of configuration for creating the RIM objects:
 - None (creates the RIM object, but does not perform any database configuration).
 - Schema scripts only; tablespaces already created.
 - Default tablespaces and run schema scripts.
- ▶ Configure the parameters for each RIM object (the typical install uses the `cm_db` database name and the default user accounts for all RIM objects).
- ▶ Configure Enterprise Directory Query Facility during the installation.

Note: With the typical installation, the Enterprise Directory Query Facility is also installed, but you are not prompted for configuration values, and so must configure this component later. When you choose the custom option, you will be prompted for Enterprise Directory Query Facility parameters.

2.10.3 Starting the installation programs

Before starting the installation program, read the information about the installation you are planning to perform. We describe the general procedures for starting the installation programs in this section.

UNIX

From the `/FRESH` subdirectory of the *IBM Tivoli Configuration Manager Installation* CD 5, enter one of the following commands:

- ▶ If you do not have Java Virtual Machine Version 1.3.1 on the system and you want to download this software to the `/tmp` directory, enter:

```
./file .bin
```

Where *file* is the name of the file that starts the installation program. For each UNIX® operating system, there is a different installation program. For example, for IBM AIX 5L, the installation program file is `setup_aix.bin`.

- ▶ If you want to download the Java Virtual Machine to a directory other than the /tmp directory, enter:

```
./file .bin -is:tempdir directory
```

Where *directory* is the directory to which you download the Java Virtual Machine.

Note: You need at least 50 MB of free space in your tempdir directory.

- ▶ If you have a Java Virtual Machine on the system and do not want to use the Java provided by Tivoli, enter:

```
java -D is.external.home=path -jar setup.jar
```

Where *path* is the path to the setup.jar file that is located on the installation CD under the /FRESH subdirectory.

Note: If the correct version of Java is not installed, the following message appears at the beginning of the install:

```
#java -Dis.external.home=/img/cd5/FRESH -jar setup.jar -jar : illegal  
argument  
Usage: java [-options] class
```

Windows

From the /FRESH subdirectory of the *IBM Tivoli Configuration Manager Installation CD 5*, run the **setup.exe** file.

2.11 Desktop installation

With IBM Tivoli Configuration Manager V4.2.2, you can make a Tivoli endpoint a fully operational Tivoli console on the following Microsoft Windows systems:

- ▶ Windows 2000 Server
- ▶ Windows XP
- ▶ Windows Server 2003

The following components are installed on a Windows PC through the desktop install:

- ▶ Tivoli Desktop for Windows
- ▶ Tivoli Java components
- ▶ Distribution Status console
- ▶ Activity Planner GUI
- ▶ Change Manager GUI

- ▶ Inventory GUI
- ▶ Software Package Editor

During the desktop install, ISMP synchronously runs the following activities behind the scenes:

- ▶ Installs Desktop for Windows.
- ▶ Temporarily unpacks Software Distribution disconnected commands (SPB).
- ▶ Installs a prerequisite SPB for environment setup.
- ▶ Installs all Java mandatory prerequisites.
- ▶ Installs selected applications.
- ▶ Cleans up the environment.

Starting the installation program

In order to start the desktop install, perform the following steps on a Windows system:

1. Insert the *IBM Tivoli Configuration Manager Desktop* CD in the CD-ROM drive.
2. Start the installation by running the **setup.exe** file.

Note: You need 120 MB of free disk space for the desktop installation.

2.12 Web Gateway installation

The Web Gateway installation program uses SPBs to install the Web Gateway components (database and server). Other than SPBs, there is no other method to install Web Gateway components.

The typical installation installs the following components:

- ▶ Tivoli endpoint
- ▶ Web Gateway database
- ▶ Web Gateway server
- ▶ Web Infrastructure
- ▶ Inventory plug-ins for Web Infrastructure
- ▶ Software Distribution plug-ins for Web Infrastructure

The custom installation provides flexibility, because you can install any combination of the listed components.

2.12.1 Web Gateway prerequisites

Although you can install most prerequisite software on any system that is accessible to the Web Gateway server, there are three components that must be installed on the system that will become the Web Gateway server. The prerequisite components that must be installed on the Web Gateway system are:

- ▶ IBM DB2
- ▶ IBM HTTP Server
- ▶ IBM WebSphere Application Server

Optionally:

- ▶ Access Manager and WebSEAL must be installed and configured in the environment to protect Web resources.
- ▶ If Access Manager is installed, configure the Java Runtime Environment provided by Access Manager (PDJRTE).

2.12.2 Starting the installation program

To start the Web Gateway installation program, from the *IBM Tivoli Configuration Manager CD 4 for Web Gateway*:

- ▶ UNIX

Run `./file.bin` for UNIX, where *file* is the name of the file that starts the installation program. For each UNIX operating system, there is a different installation program. For example, for IBM AIX 5L, the installation program file will be `setup_aix.bin`.

- ▶ Windows

Run the `Setup.exe` file.

2.13 Uninstallation of IBM Tivoli Configuration Manager

To uninstall IBM Tivoli Configuration Manager, use the `wuninst` command. The `wuninst` command is not specific to IBM Tivoli Configuration Manager. It is used for all Tivoli Enterprise products. It is a wrapper script that invokes product-specific uninstall scripts. This command removes the component for the specified machines in your Tivoli environment or from the entire local Tivoli region. To uninstall the component using the `wuninst` command, you need to specify the component tag for that component.

The syntax for this command is:

```
wuninst tag hostname... [-rmfiles]
```

Where:

- ▶ *tag* specifies the registered product tag for the Tivoli Enterprise product to remove.

Tip: For the list of these tags, see *IBM Tivoli Configuration Manager Planning and Installation Guide*, GC23-4702. You can also list the registered product tags by running the **wuninst -list** command.

- ▶ *hostname* specifies the name of the managed node from which to remove the product. If you specify the Tivoli server, the product is removed from all managed nodes in the Tivoli region.
- ▶ `-rmfiles` indicates that all product files are to be removed from specified managed nodes. When you do not specify this option, the command removes the database entries only. When you issue this command from the Tivoli server and specify this option, all entries for each managed node in the Tivoli region are removed.

2.14 Architectural considerations for Tivoli regions

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

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

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

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

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

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

2.14.1 Benefits of connecting Tivoli regions

Certain situations in a Tivoli environment can make the connection of two or more Tivoli regions necessary or desirable. These situations include:

- ▶ *Decrease server load:* The load on a single Tivoli server caused by network activity, memory demands, or the number of clients can be lessened by multiple Tivoli regions.
- ▶ *Localized system administration:* Multiple Tivoli regions enable local control with more independence at different operational sites.
- ▶ *Enhance security:* An additional Tivoli region can be used to restrict local administrators' access to certain machines within the enterprise. Also, an additional Tivoli region enables differing encryption levels within a Tivoli environment.

With the *super authorization role* and the Tivoli region password (if it exists), a Tivoli administrator can connect or disconnect two or more Tivoli regions.

2.14.2 Connection types

There are two types of connections.

One-way region connections

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

Two-way region connections

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

2.14.3 Case study: Hub-spoke architecture

The Tivoli physical topology is primarily determined by the underlying network topology and management system performance goals. In large network environments, we recommend deploying your Tivoli environment using a hub-spoke architecture. In a hub-spoke architecture, the Tivoli environment is segmented into several TMRs. Each TMR can be responsible for directly managing a different physical segment of the enterprise, serve a specific business unit, or be organized by security access levels. The central TMR that manages the other TMRs is called the *hub*, and the TMRs that it manages are called *spokes*.

Whether using a one-way or two-way connection between the hub and spoke TMRs, the hub Tivoli server forms the central administration point from which all managed functions are performed within a Tivoli environment. It is dedicated primarily to high-level management functions, such as creating administrator desktops and Tivoli Enterprise Consoles; creating, configuring, and distributing sentry profiles to spoke servers; and other hub-wide management activities.

Spoke TMRs provide the direct control function for all endpoints in the Tivoli environment. Spoke regions can be used to group managed nodes by physical location in the network and to localize functions in order to improve network and system performance. Generally, spoke TMRs are not used as entry points for administrators. Tivoli administrators can use either the hub TMR or any managed node strategically placed in the design as an entry point into the Tivoli management environment.

Figure 2-3 on page 50 illustrates a sample hub-spoke architecture.

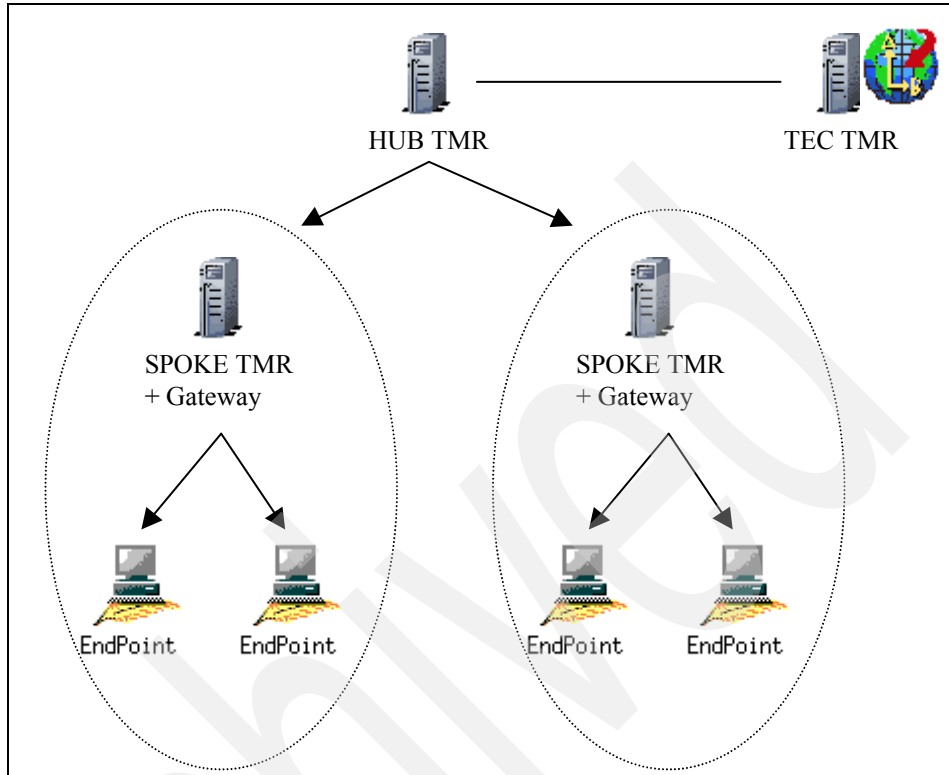


Figure 2-3 Hub-spoke architecture

All managed systems (managed nodes, gateways, and endpoints) are spread out into the Tivoli environment beneath spoke TMRs, as determined by function, server load, or physical network location.

Managed nodes are still required in this environment. For example, managed nodes can be used to support remote Tivoli administrators' desktops or to serve for profile staging.

Endpoint gateways are installed throughout the Tivoli environment to host endpoints. In this Tivoli environment hierarchy, all endpoint gateways are assigned to spoke TMRs only.

Considerations when deciding on the design of the hub-spoke architecture

Now, we review considerations when deciding on the design of the hub-spoke for Tivoli region architecture.

Tivoli region architecture

One of the most important factors for designing the hub-spoke Tivoli region architecture is the scalability limitations of the Tivoli environment:

- ▶ Number of endpoints

The number of endpoints managed by a single region has been increased to tens of thousands. It has been shown in production environments that 20,000 endpoints can be managed by a single region. For organizations requiring more than 20,000 endpoints to be managed, multiple regions are *required*. The limit of 20,000 endpoints represents a threshold beyond which special performance and tuning requirements might be needed. Therefore, use multiple connected regions.

- ▶ Number of managed nodes

Generally, a single Tivoli server can support a maximum of 200 managed nodes. However, use endpoints instead of managed nodes in most cases. Endpoints are the preferred mechanism for managing your environment. The introduction of endpoints greatly reduces the number of managed nodes in a single region. A gateway installed on a managed node can perform all communication and operations with thousands of endpoints. Endpoints, therefore, have no direct communication with a Tivoli server.

In addition, the ability to perform maintenance functions such as database checks are greatly inhibited by large numbers of managed nodes. If the network contains more than 200 managed nodes, create multiple regions and connect them.

Note: For performance reasons, in a multi-Tivoli region environment, it is important to make sure that endpoints get connected to the designated Tivoli region. The best way to ensure that is to configure the endpoints to log in to specific gateways and disable broadcasting.

Recommendations for creating policy regions in a hub-spoke architecture

Now we review recommendations on creating policy regions in a Hub-Spoke for TMR architecture.

Tivoli region architecture

It is a good practice to create policy regions based on a Tivoli application (Tivoli Configuration Manager, Tivoli Monitoring, and so on) only on the hub Tivoli server. The subscriber policy regions then reside on the spoke Tivoli servers. The subscribed policy regions contain the profile managers used for distributing to endpoints. Organizing your policy regions in this manner enables the hub

server to be the central point of operations for each application and associated functions.

This also avoids subscribing endpoints across Tivoli region boundaries. If an endpoint is subscribed across Tivoli region boundaries, a new object is created in the object database and the `wchkdb` command must track the object directly, causing unnecessary transactions across Tivoli region boundaries and server load. Instead, if endpoints stay subscribed to their local Tivoli region, the hub and spoke Tivoli regions only need to exchange resources, causing only an entry in the Tivoli Name Registry on the hub Tivoli region.

See Figure 2-4 for more details.

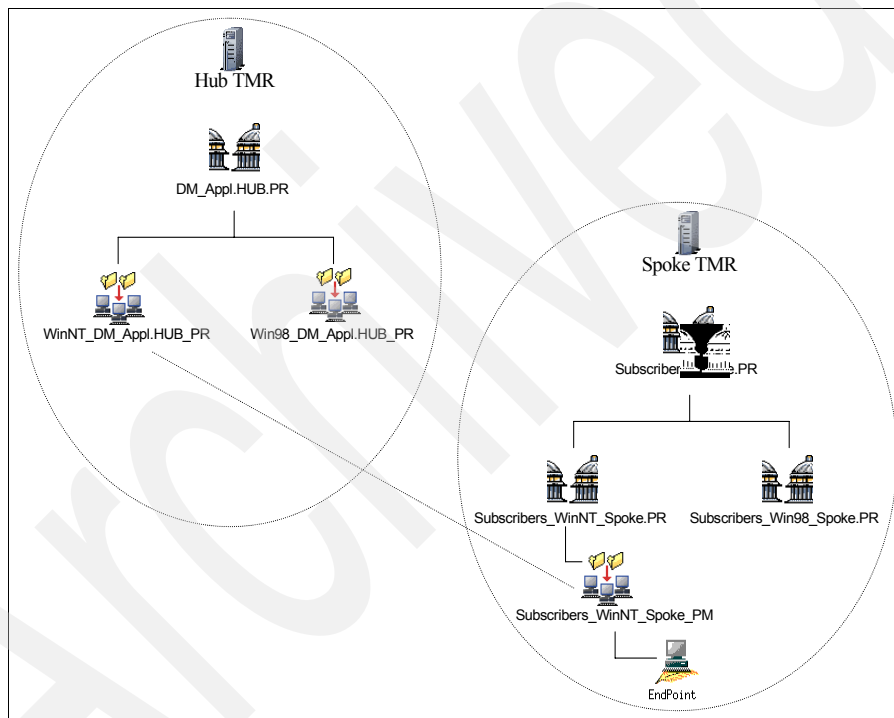


Figure 2-4 Subscription example in a hub-spoke model

2.15 Backup strategies

This section describes several of the backup strategies that should be deployed when using Tivoli Configuration Manager V4.2.2. Without a good backup strategy, the enterprise can be vulnerable to outages of undeterminable lengths of time.

2.15.1 Tivoli backups

During the installation process, the only logging files that are created are `tivoli.sinstall` and the `tivoli.cinstall`. The `tivoli.sinstall` file gets created with the Tivoli Framework server installation. The `tivoli.cinstall` file gets overwritten with each product that gets installed on the Tivoli server.

There are two ways to back up the Tivoli server and managed nodes.

The `wbkupdb` command is available to back up the pertinent files in the `$DBDIR` directory. This will not back up any custom scripts that you might have or anything outside of `$DBDIR`. But, this is sufficient to restore a Tivoli server back to running state if there is some corruption in the database.

2.15.2 System-level backups

The other method is to do a system level backup, or back up everything under the `../Tivoli` directory. This will capture all the scripts that were built for tasks and any other custom scripts that might be in the environment.

With both of these methods, it is best to be run outside of the Tivoli environment, especially if you are running these on a schedule. On UNIX, run a cron job, and on Windows, run it using Windows Scheduler.

It is also important to make sure you have a “clean” system before backing up. Use the `wchkdb` command with the appropriate parameters of either `-ux` for interconnect Tivoli regions, `-u` for all managed nodes, and `-ut` for just the Tivoli server. A “clean” `wchkdb` command result will allow for better backup and restore capabilities.

One of the steps that is almost always forgotten is to check the backups for validity. Too often there are backups that are not validated, and when necessary, they might not be good candidates for restore purposes. A good plan is to have a machine that you can use to restore the backup. Then, use a set of tests to make sure that the backup is valid. If it is not, debug the problem to make sure that you can back up successfully. One debugging tip is to back up individual managed nodes and not the whole Tivoli region at a time. If there is a failure, you can see which managed node has the failure and further debug just that managed node.

Archived

Installation of IBM Tivoli Configuration Manager in a small/medium enterprise

In this chapter, we go through a typical installation scenario that would be comparable to what would happen in a small-to-medium enterprise. We introduce the XYZ Corporation, a fictitious company and deploy IBM Tivoli Configuration Manager V4.2.2 on a Microsoft Windows server™. We also go through some troubleshooting examples as we go along and get some common errors for the installation of Tivoli Configuration Manager V4.2.2.

Note that this deployment would be an ideal scenario for a proof of concept or demo installation. will cover a more complex installation in Chapter 4, “Installation of IBM Tivoli Configuration Manager in a large enterprise” on page 111.

We discuss the following topics in this chapter:

- ▶ The XYZ Corporation
- ▶ Description of the environment
- ▶ Installing DB2 UDB on a Microsoft Windows server
- ▶ Installation of Tivoli Management Framework and Configuration Manager components (except Web Gateway, LDAP, and Pristine Manager)

3.1 The XYZ Corporation

The XYZ Corporation is a small-to-medium sized business that employs 250 people in its operation. The XYZ Corporation has an IT department that manages the operating systems of each of the 250 computers that are currently being used by the employees for data entry. The XYZ Corporation has offices in the United States, Canada, and Europe.

Note: All names and references to company and other business institutions used in this chapter are purely fictional. Any match with a real company or institution is coincidental.

3.2 Description of the environment

Of the 250 computers that XYZ Corporation has, approximately 10% are Red Hat Linux machines and the other 90% are differing flavors of Microsoft Windows. Of the Microsoft Windows computers, approximately 50% are Microsoft XP Professional, 30% are Microsoft 2000 Server, 10% are Microsoft Server 2003, and 10% are Microsoft Windows 98.

The network is a fast third generation (3G) network employing Cisco routers. The XYZ Corporation has several of the Linux computers hosting the companies Web site. The remaining computers are all behind a firewall within its own network.

We only deploy to four endpoints in this scenario, but deploying to any number of endpoints is essentially the same. It is also important here to mention that the gateway can handle approximately 2000 endpoints, unless you are also using Software Distribution, Inventory, IBM Tivoli Monitoring, Distributed Monitoring, and IBM Tivoli Enterprise Console®. These products will lessen the number of endpoints that can effectively be maintained by the gateway.

Figure 3-1 on page 57 shows the test environment of XYZ Corporation that we used to deploy Tivoli Configuration Manager.

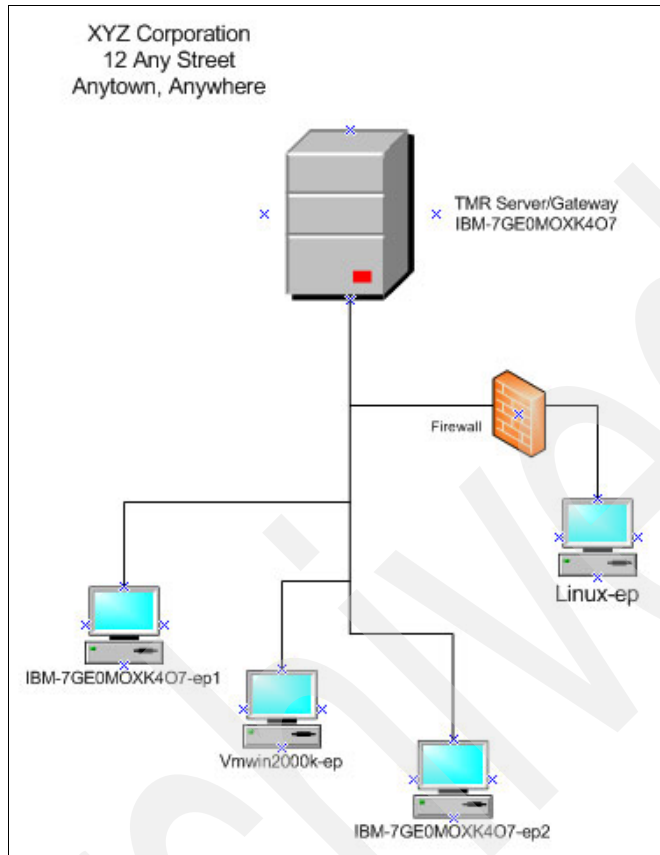


Figure 3-1 Example topology

It is always a good idea to have some kind of map of how you are going to deploy the enterprise. This can allow for more questions such as what routers do I have to contend with, where are the firewalls, and where are my slow links.

3.3 Installing DB2 UDB on a Microsoft Windows server

Before installing Tivoli Configuration Manager, we need install IBM DB2 Universal Database™ (UDB) in our environment. To install DB2 UDB on a Windows system, you need a drive with enough space to hold the binaries of the database program and enough space for the database. To learn the latest hardware and software requirements, check the *DB2 Universal Database Release Notes Version 8*, which comes online with the product.

Perform the following steps to install DB2 UDB on a Windows server. Note that in this environment, we install DB2 UDB on the same machine as the Tivoli server.

1. On the DB2 CD-ROM for Windows is the setup program needed to install DB2 UDB database. Figure 3-2 shows the first window you will see when starting the **setup.exe** program.

From this Welcome window, you can view the Installation Prerequisites, the Release Notes, take a Quick Tour, Install Products, and Exit.



Figure 3-2 Welcome window

It is always a good idea to check the *Release Notes* for any new developments or corrections that might have taken place with the product before installing the product. There might also be some last minute installation requirements or last minute errors that need to be addressed by the installer.

2. Select **Install Products** from the menu bar. The window shown in Figure 3-3 opens.



Figure 3-3 How to set up a connection

3. Here, select how you want to set up the connection. You can either use a pre-existing profile, search the network for databases, or manually configure a connection to the database. We select **Search the network** to look for a database in the network to see if there might be one there that we can use. A good idea is to talk to the database administrator to see what your database options are.

4. Figure 3-4 shows the output of the network search option from the previous window. We select the **Known systems** option here if we know what they are. We assume here that we do not know of any database that we can use and we proceed to install a new database on the Tivoli server. This might be a good idea in such a small network. As the network gets larger, we might want to look at moving the database off of the Tivoli server to a managed node. Here, our concern in disk space and speed for the optimum performance in the query scans and other reporting that we might need.

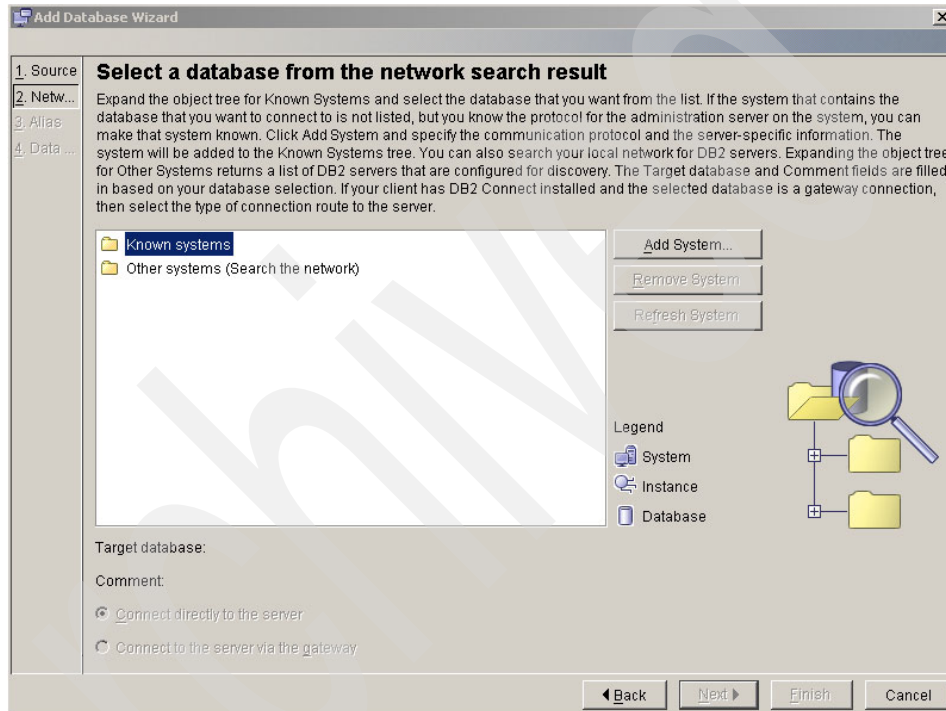


Figure 3-4 Network search

5. In the window in Figure 3-5, select the method that you want to use for configuring the connection. We now select **Manually configure a connection to a database**. This assumes that we do not currently have a database available for our use.

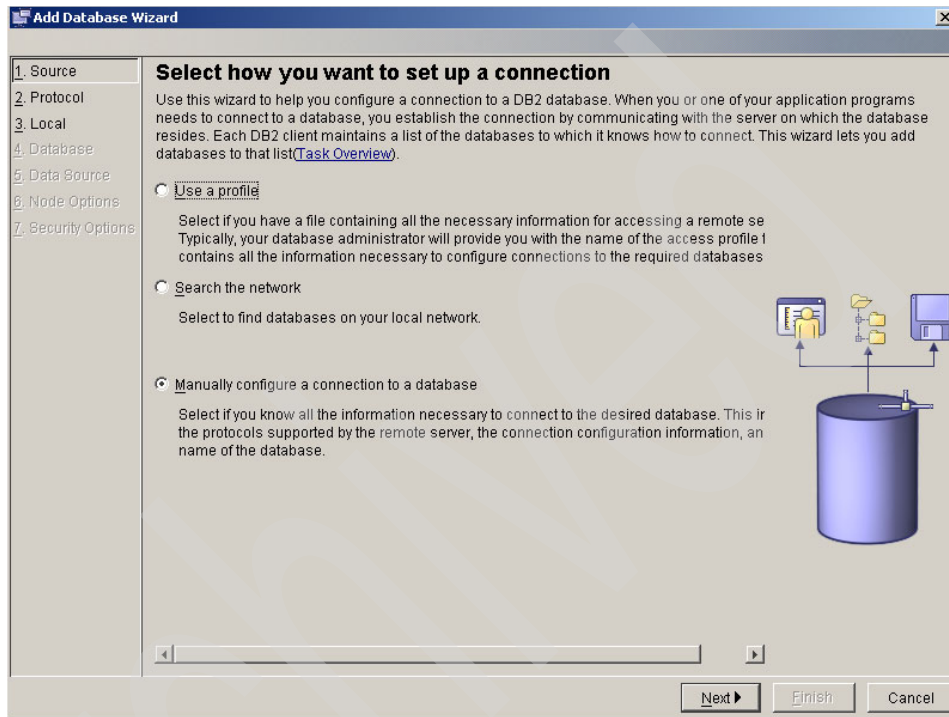


Figure 3-5 Set up a connection

6. The next window (Figure 3-6) asks for the communication protocol. We select the **TCP/IP** communications protocol.

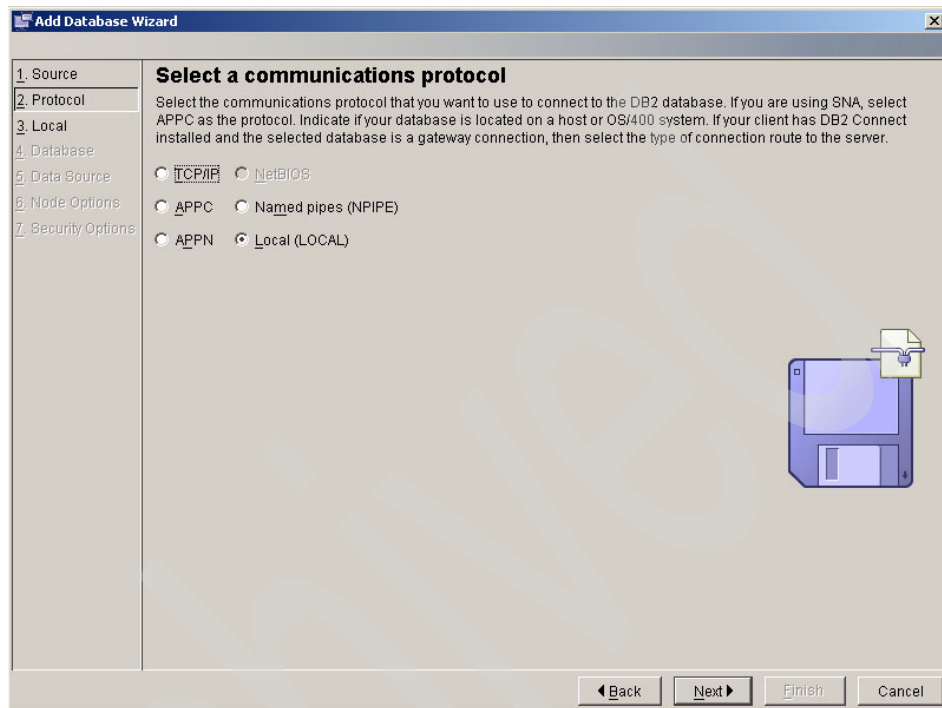


Figure 3-6 Communications protocol

7. In Figure 3-7, we specify information for a database on this system by selecting the **Database in another instance**.

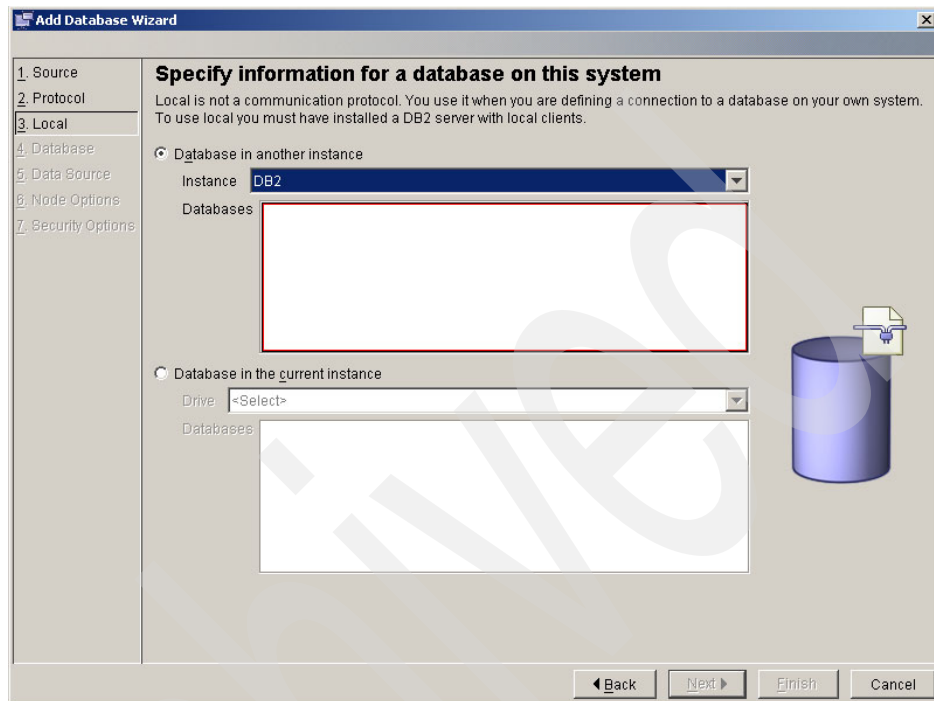


Figure 3-7 Specify information for a database

As you can see from Figure 3-8, there are no databases here or any aliases available.

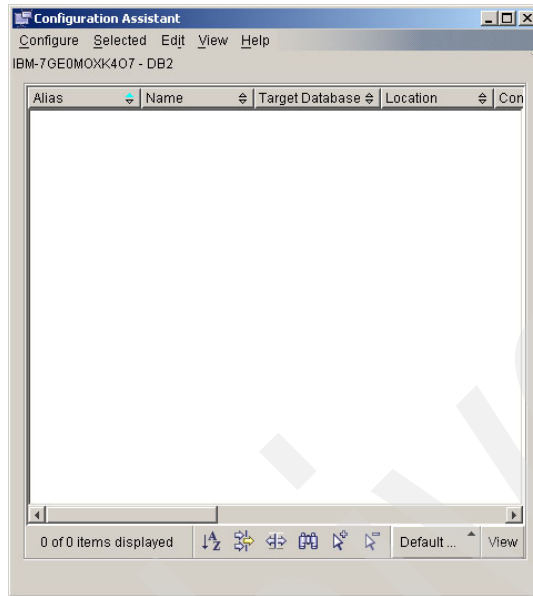


Figure 3-8 Configuration Assistant

8. In Figure 3-9, we select **Add System** to add another system.

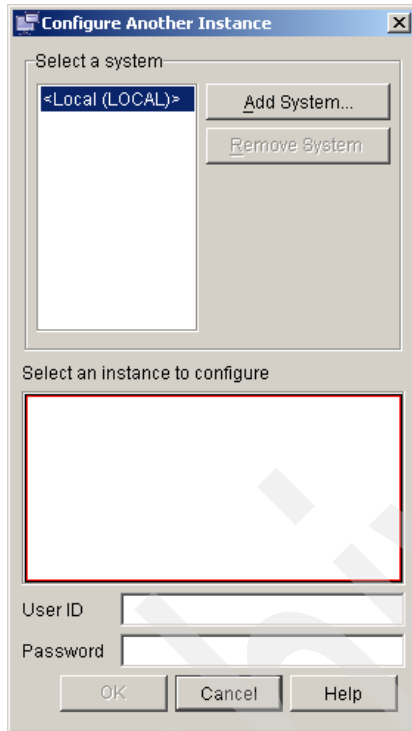


Figure 3-9 *Configure Another Instance*

9. In Figure 3-10, we enter the system name, host name, node name, and what operating system that is employed. You can use the **Discover** button and the **View Details** button for their respective variables. When you are finished, click the **OK** button. The database will be set up at this point and will take several minutes to complete.

The screenshot shows a dialog box titled "Add System" with the following fields and buttons:

- System name:** Text box containing "IBM-7GE0MOXK407" and a "Discover" button.
- Host name:** Text box containing "IBM-7GE0MOXK407" and a "View Details..." button.
- Node name:** Text box containing "IBM-7GE0".
- Operating system:** Dropdown menu with "Windows" selected.
- Comment:** Empty text box.
- Buttons:** "OK", "Cancel", "Apply", "Reset", "Show Command", and "Help".

Figure 3-10 Add System

After the database is created, the Control Center lists the host system, as shown in Figure 3-11.

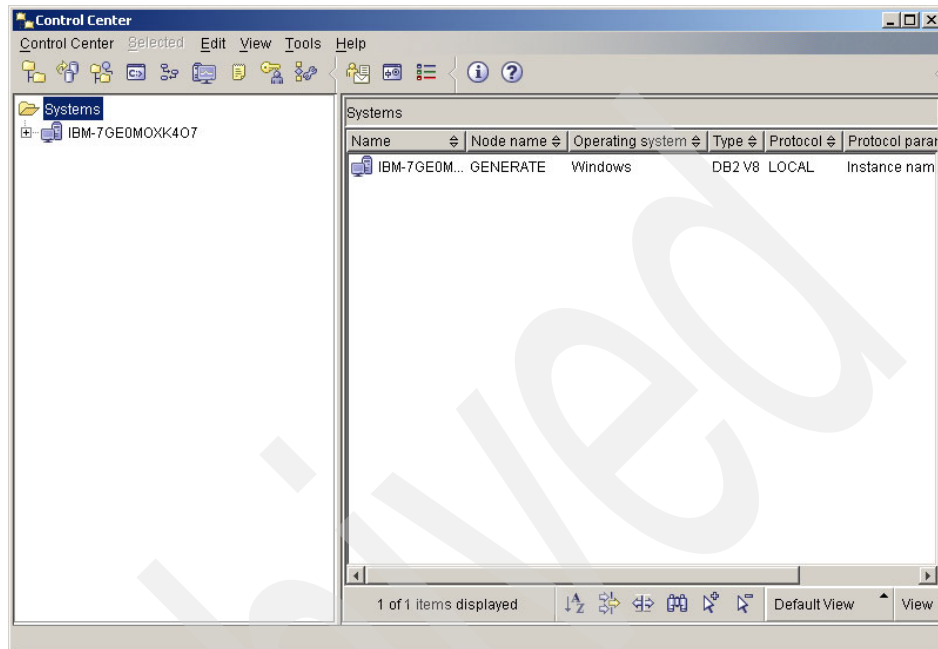


Figure 3-11 Control Center

As shown in Figure 3-12, a databases folder is also created.

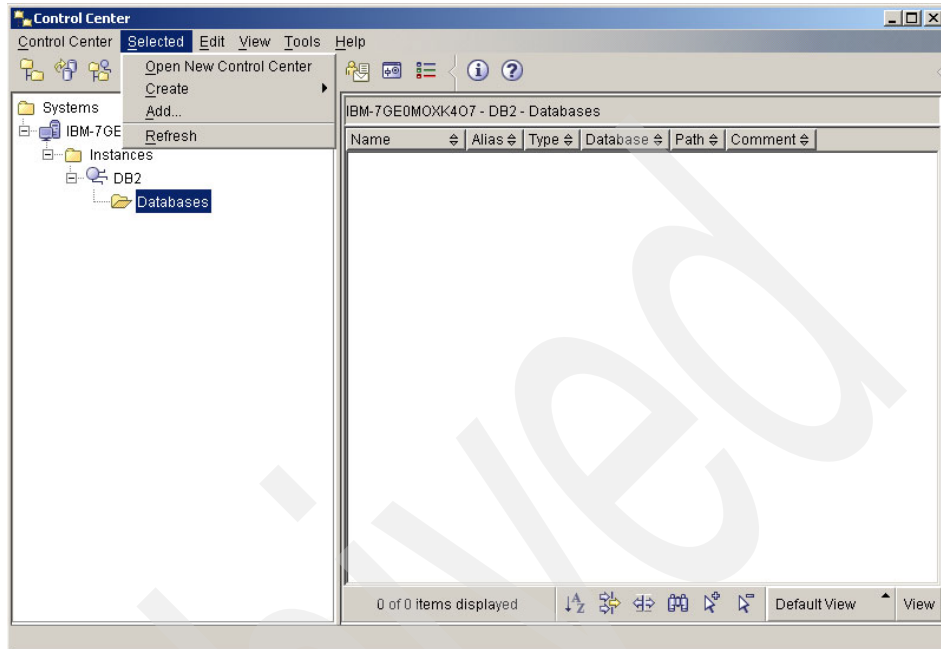


Figure 3-12 Control Center

10. We now attempt to start the database. Figure 3-13 shows the error pop-up window.

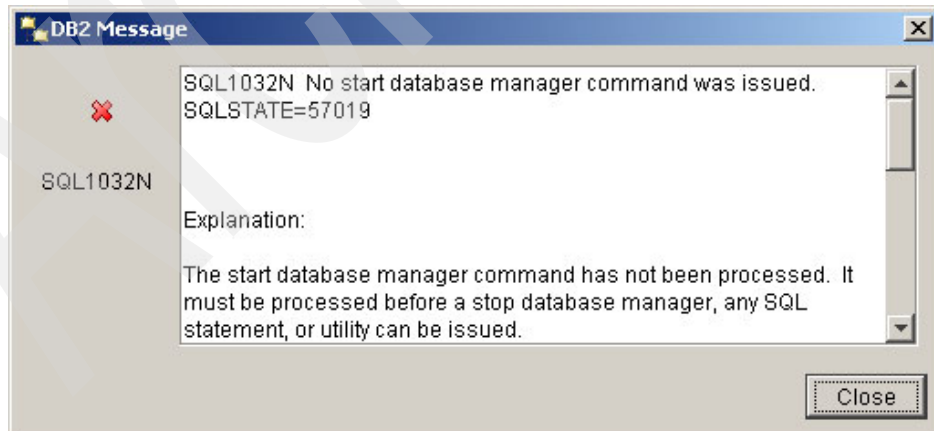


Figure 3-13 Error message

This error tells us that we have not started the database yet. In fact, we do not have a database installed yet, just the DB2 program files. Next, we install a database.

11. Installing a sample database is always a good idea. It does not take up much space and you can use it for a baseline to check to see if the database is functional. From this window (Figure 3-14), click **Create Sample Database**.

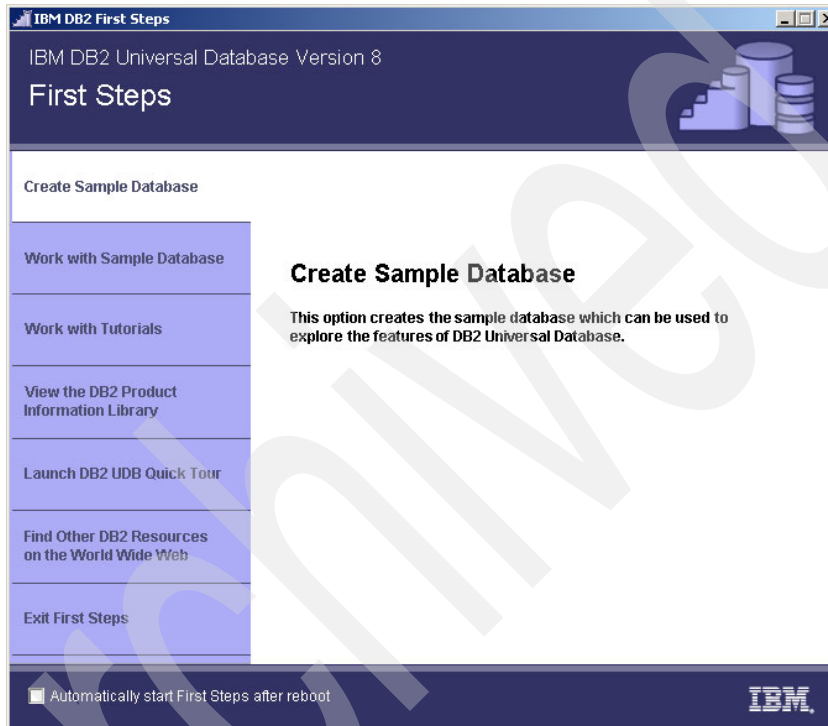


Figure 3-14 Create Sample Database

12. Figure 3-15 shows the default description of the sample database. Click **OK** to continue.

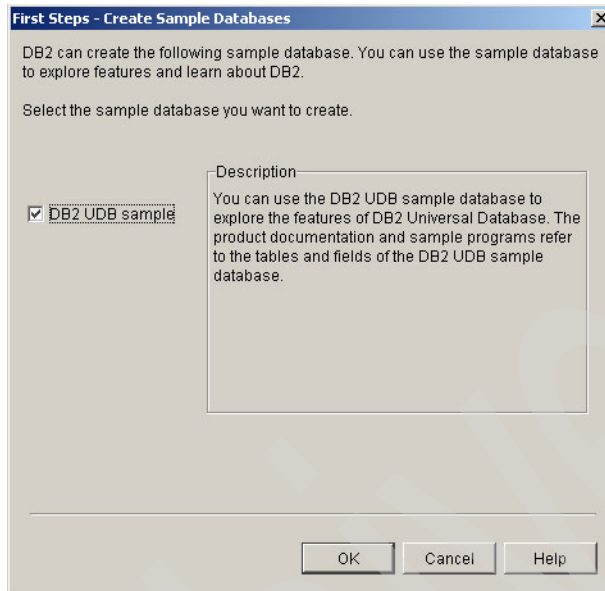


Figure 3-15 Create Sample Databases

13. in the database creation cycle, first an MS-DOS® window opens. No action is required here.

14. Figure 3-16 shows us that the database is being created and how much time has elapsed.

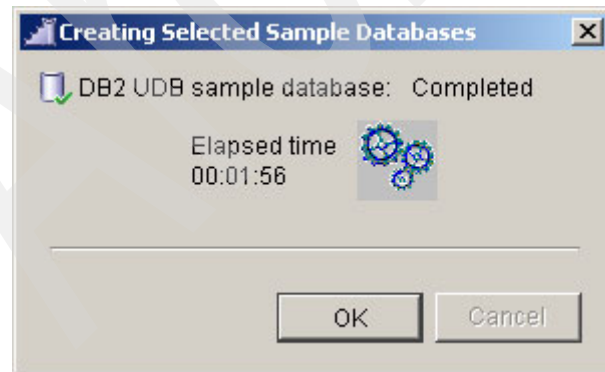
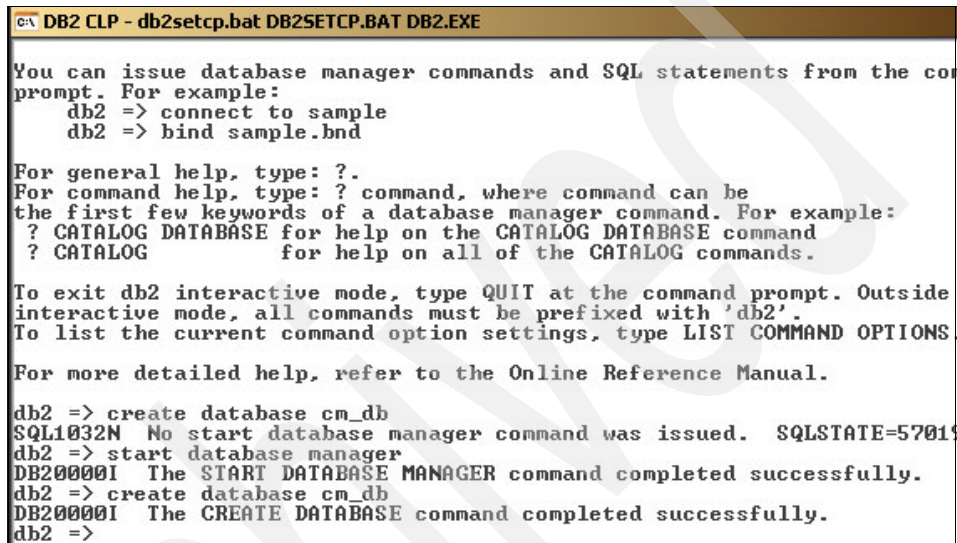


Figure 3-16 Creating Selected Sample Databases

15. If you log in to the computer with the DB2 administrator ID and password (from the command line, issue a `su - db2admin`), you can connect to the sample database. Also, as shown Figure 3-17, we have issue the command `create database cm_db`. You can see that it was successful only after we issued the start database manager command. It is always important to read all the errors, because they will provide clues as to why the error was spawned.



```
DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE

You can issue database manager commands and SQL statements from the command
prompt. For example:
  db2 => connect to sample
  db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG           for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => create database cm_db
SQL1032N No start database manager command was issued.  SQLSTATE=57015
db2 => start database manager
DB20000I The START DATABASE MANAGER command completed successfully.
db2 => create database cm_db
DB20000I The CREATE DATABASE command completed successfully.
db2 =>
```

Figure 3-17 After the database has been created

16. We can then connect (Figure 3-18) to the cm_db successfully. It was created correctly and is now usable.

```
DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE
the first few keywords of a database manager command. For example:
? CATALOG DATABASE for help on the CATALOG DATABASE command
? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS

For more detailed help, refer to the Online Reference Manual.

db2 => create database cm_db
SQL1032N No start database manager command was issued.  SQLSTATE=5701
db2 => start database manager
DB20000I The START DATABASE MANAGER command completed successfully.
db2 => create database cm_db
DB20000I The CREATE DATABASE command completed successfully.
db2 => connect to cm_db

      Database Connection Information

Database server      = DB2/NT 8.1.0
SQL authorization ID = JSIMMONS
Local database alias = CM_DB

db2 =>
```

Figure 3-18 Database information

17. Issue a list active databases, as shown in Figure 3-19, and see that cm_db is an active database.

```
DB2 CLP - db2setcp.bat DB2SETCP.BAT DB2.EXE
For more detailed help, refer to the Online Reference Manual.

db2 => create database cm_db
SQL1032N No start database manager command was issued.  SQLSTATE=57019
db2 => start database manager
DB20000I The START DATABASE MANAGER command completed successfully.
db2 => create database cm_db
DB20000I The CREATE DATABASE command completed successfully.
db2 => connect to cm_db

      Database Connection Information

Database server      = DB2/NT 8.1.0
SQL authorization ID = JSIMMONS
Local database alias = CM_DB

db2 => list active databases

                        Active Databases

Database name          = CM_DB
Applications connected currently = 1
Database path          = C:\DB2\NODE0000\SQL00002\

db2 =>
```

Figure 3-19 List active databases

18. In Figure 3-20, you will see the directories and the files that were created when creating the database.

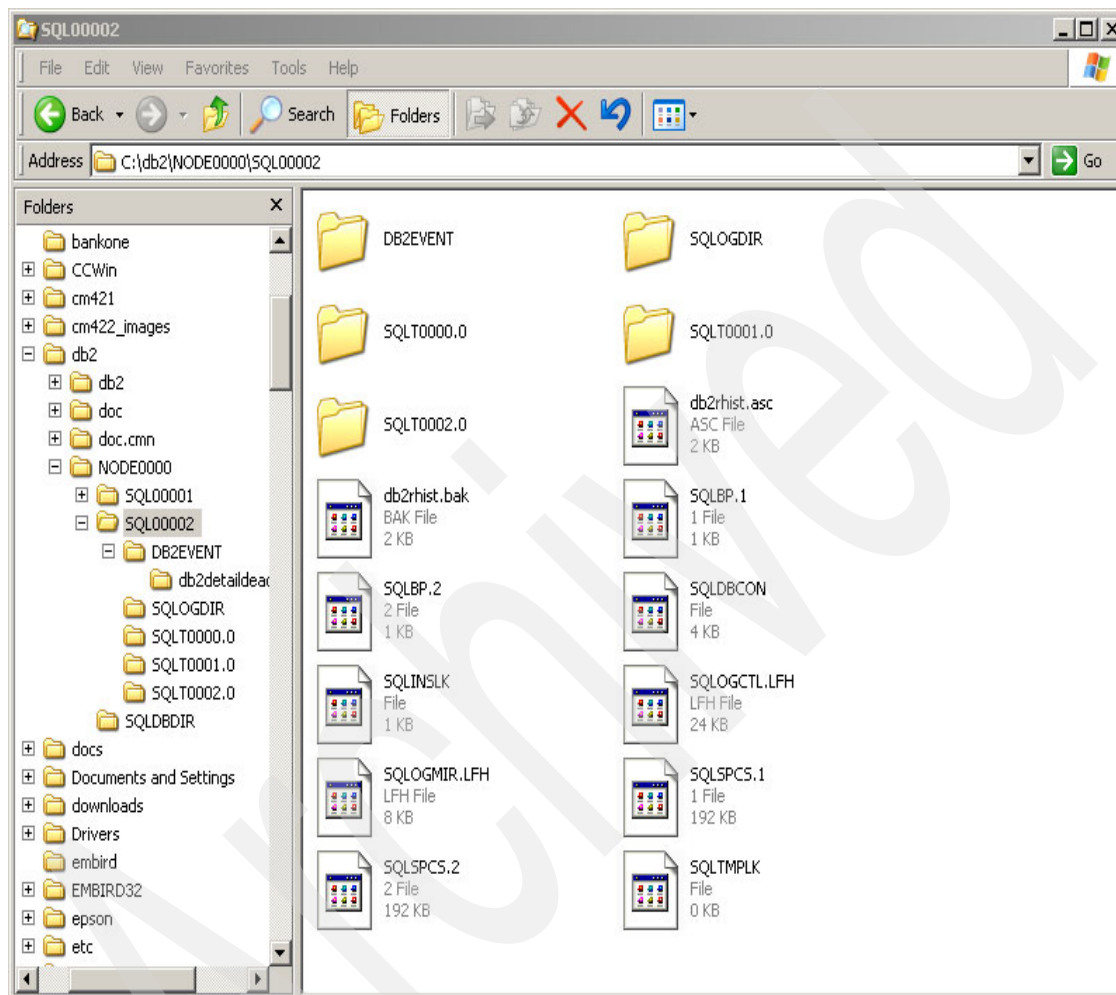


Figure 3-20 Directory listing of the database

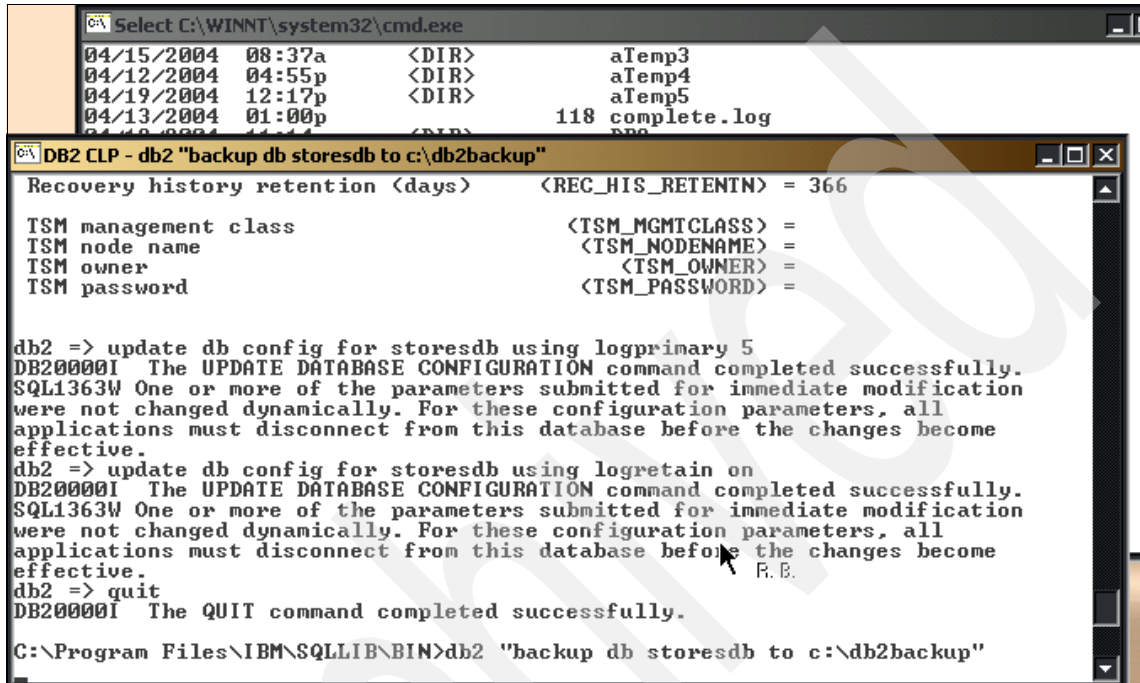
19. We recommend that you back up the database at this point and from time to time so that if something should happen and data is lost, a simple restore can bring the data back and valuable data not lost.

For example, issue the following simple command from the DB2 command prompt:

```
db2 "backup db storesdb to c:\db2backup
```

Where the db2backup directory was previously created and the backups will reside.

Figure 3-21 shows the backup operation.



The screenshot shows two overlapping windows. The top window is a standard Windows Explorer-style window titled 'Select C:\WINNT\system32\cmd.exe'. It displays a directory listing with columns for date, time, and file name. The files listed are 'aTemp3', 'aTemp4', 'aTemp5', and '118 complete.log'. The bottom window is a terminal window titled 'DB2 CLP - db2 "backup db storesdb to c:\db2backup"'. It shows the output of a DB2 backup command. The output includes configuration parameters for recovery history retention, TSM management class, node name, owner, and password. It then shows two successful 'UPDATE DATABASE CONFIGURATION' commands for 'logprimary 5' and 'logretain on', followed by a 'quit' command. The prompt at the bottom of the terminal window is 'C:\Program Files\IBM\SQLLIB\BIN>db2 "backup db storesdb to c:\db2backup"'. A large, semi-transparent watermark 'IBM' is visible across the terminal window.

```
C:\WINNT\system32\cmd.exe
04/15/2004 08:37a <DIR> aTemp3
04/12/2004 04:55p <DIR> aTemp4
04/19/2004 12:17p <DIR> aTemp5
04/13/2004 01:00p 118 complete.log
04/10/2004 11:14 <DIR> ...

DB2 CLP - db2 "backup db storesdb to c:\db2backup"
Recovery history retention <days> <REC_HIS_RETENTN> = 366
TSM management class <TSM_MGMTCLASS> =
TSM node name <TSM_NODENAME> =
TSM owner <TSM_OWNER> =
TSM password <TSM_PASSWORD> =

db2 => update db config for storesdb using logprimary 5
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, all
applications must disconnect from this database before the changes become
effective.
db2 => update db config for storesdb using logretain on
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, all
applications must disconnect from this database before the changes become
effective.
db2 => quit
DB20000I The QUIT command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2 "backup db storesdb to c:\db2backup"
```

Figure 3-21 Time to back up

Now, we are ready to install Tivoli Configuration Manager V4.2.2.

3.4 Installation of Tivoli Management Framework and Configuration Manager components (except Web Gateway, LDAP, and Pristine Manager)

In this section, we go through an installation of the Tivoli Configuration Manager components on our Microsoft Windows Tivoli server and show the windows and options that are available to the installer. We do a server installation and use only one database, cm_db, for all RIM objects.

Perform the following steps:

1. Run the setup program from the installation CD to start the Tivoli Configuration Manager V4.2.2 installation on a Windows platform. The program will prompt you to select the language to be used for the wizard (Figure 3-22). We can install in various languages here. English is the default language. Click **OK** to proceed.

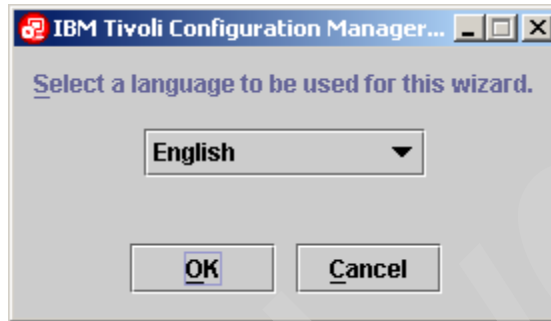


Figure 3-22 Select a language

2. The next window (Figure 3-23) contains information as to what you are about to do. As you can see from this window, we can either install or upgrade the server component of Tivoli Configuration Manager V4.2.2. For this installation, we install the product. Click **Next** to proceed.

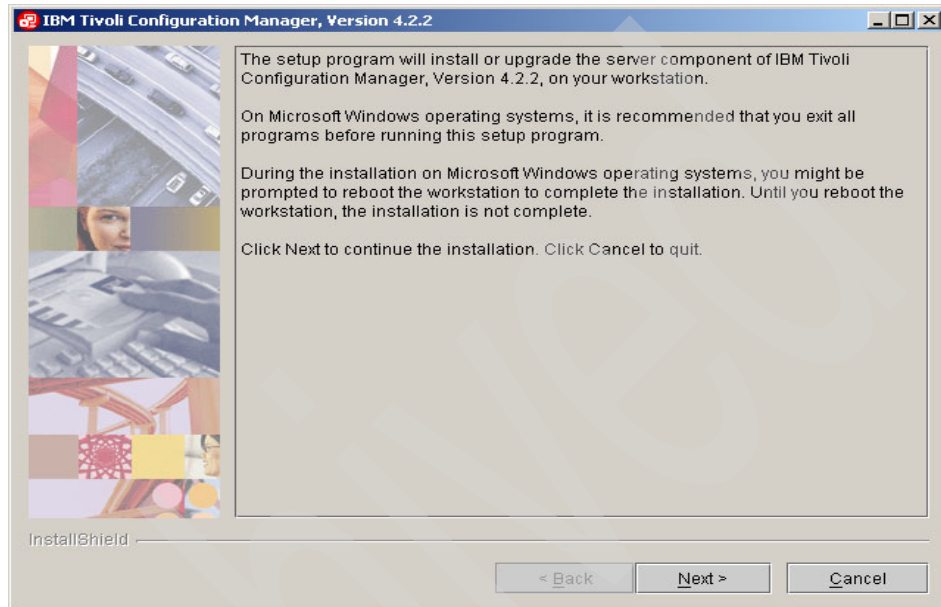


Figure 3-23 Installation information window

3. In the License Agreement window (Figure 3-24), select **Accept** and then click **Next**.

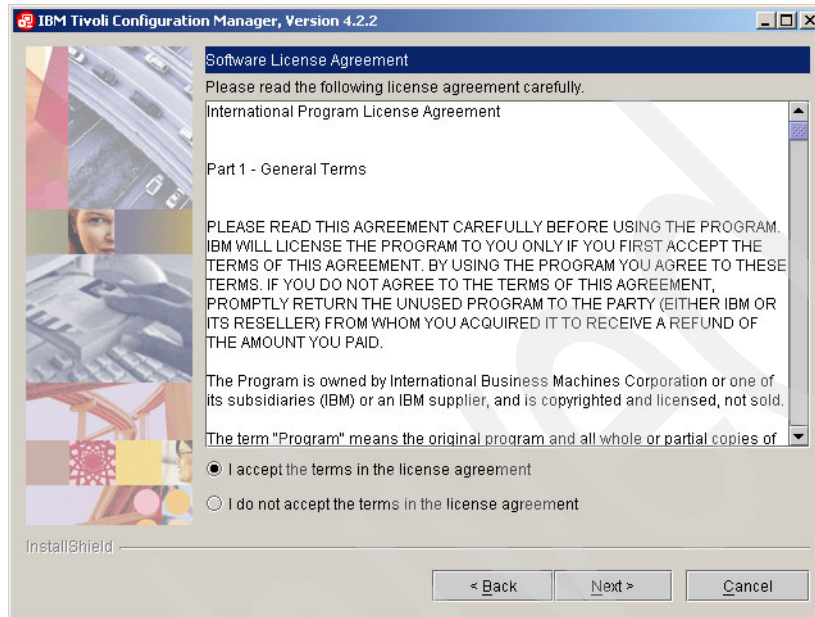


Figure 3-24 License Agreement

4. You will see the window in Figure 3-25 only if there is a steepens script on the computer, but oserv is not running. Note that the setup_env script is installed as part of a Tivoli Management Framework installation. If you get this message and continue to install, the installation wizard will reinstall the Framework binaries and the existing instance of Framework will be overwritten.

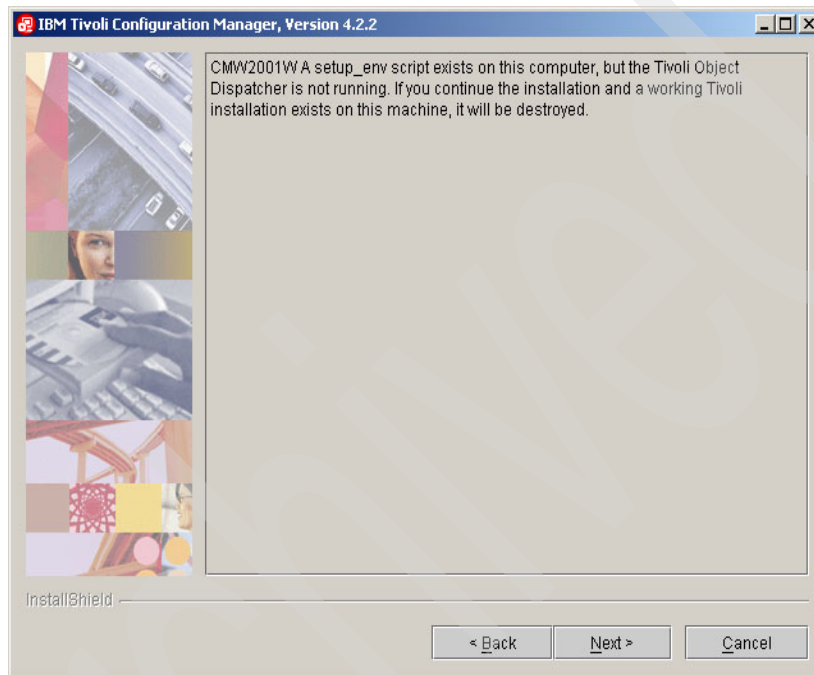


Figure 3-25 Warning window

5. In the next window (Figure 3-26), select the Destination directory for the installation.

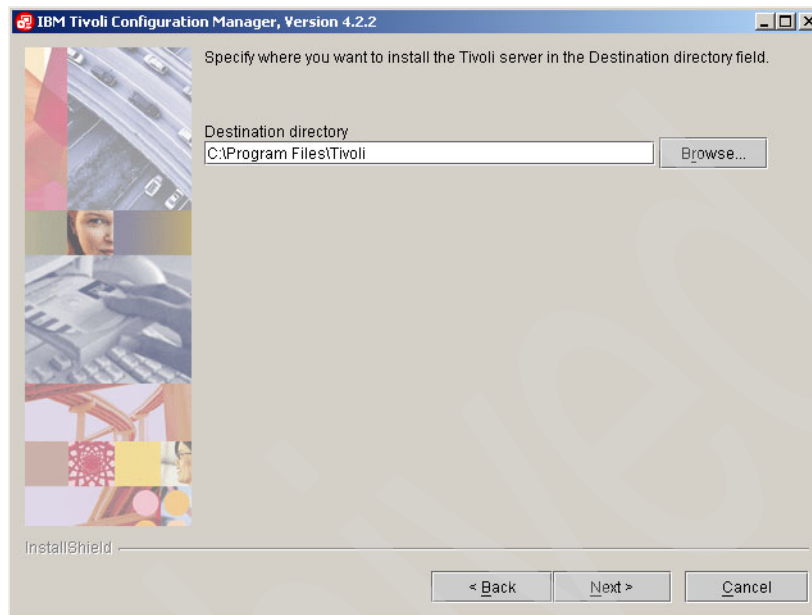


Figure 3-26 Destination directory

6. In Figure 3-27, we select **Custom**, because we do not want to install everything. Note that Typical is the default option.

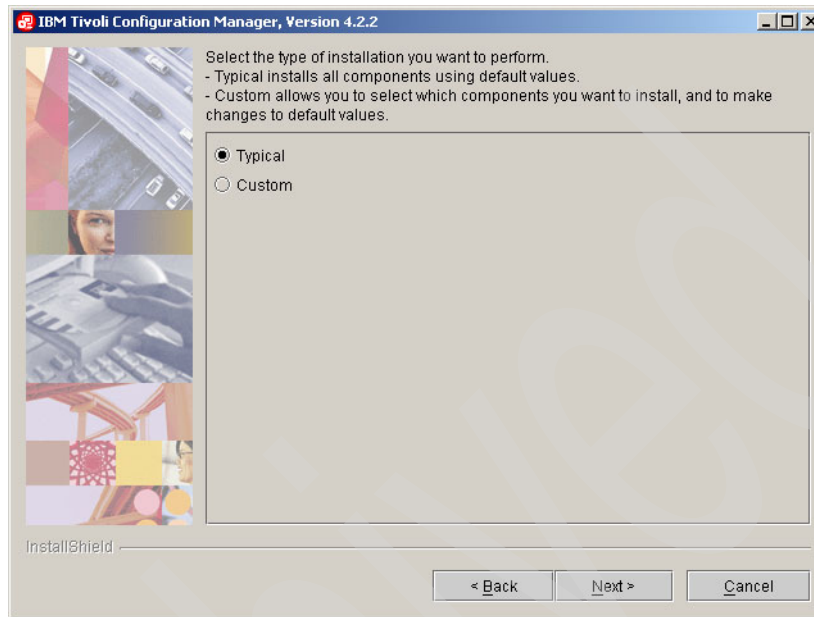


Figure 3-27 Type of installation

7. In the next window (Figure 3-28), you can select other installation languages.

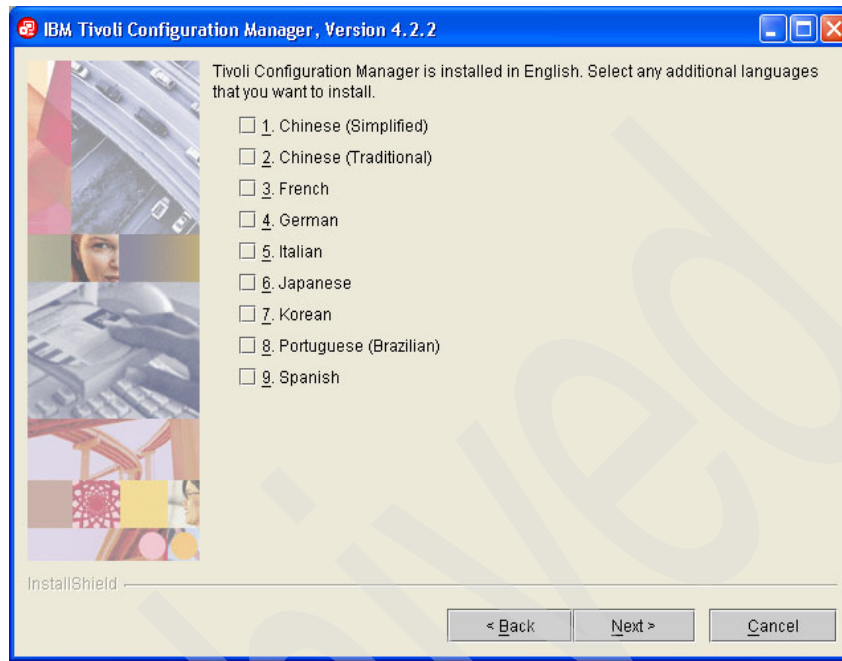


Figure 3-28 Additional languages

8. In the next window (Figure 3-29), we choose the repository configuration. This step will save us from having to do this manually with all the products.

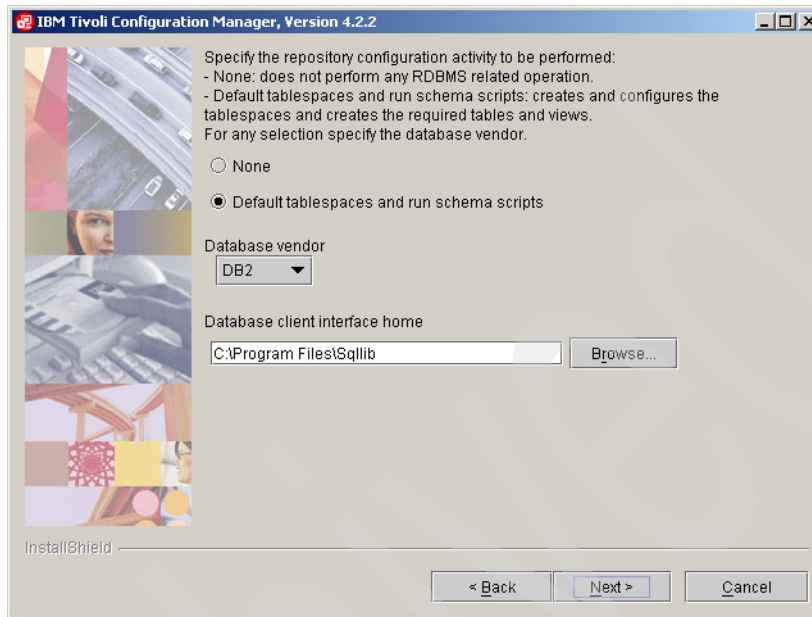


Figure 3-29 Repository configuration

9. In Figure 3-29, make sure that you change the interface home. It is C:\Program Files\IBM\SQLLIB. If you are not sure, click **Browse** to find the database home, as shown in Figure 3-30 on page 83. We verify that it is under the c:\Program Files\IBM\SQLLIB directory.

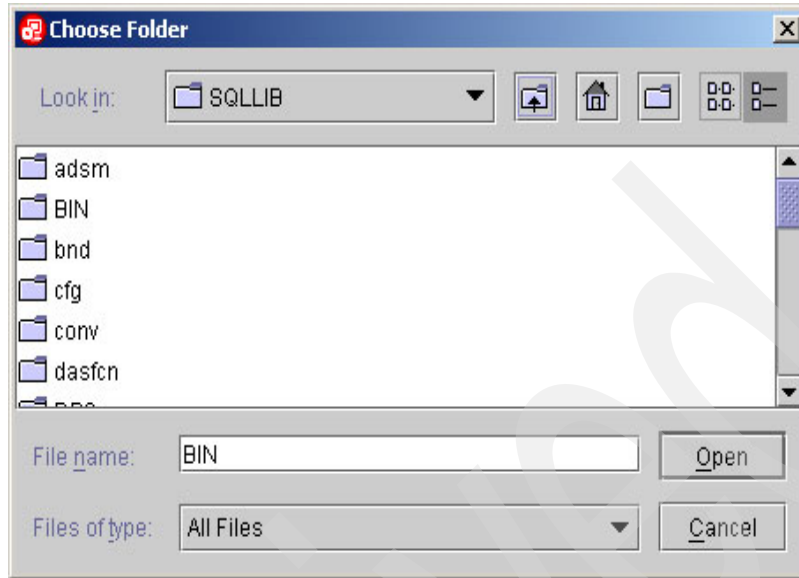


Figure 3-30 Browse window

10. From a CMD window or MS-DOS prompt, as shown in Figure 3-31, we can run some DB2 commands. This simple DB2 command screen shows us that the active databases are working correctly and we can connect to the database cm_db.

```

DB2 CLP

For more detailed help, refer to the Online Reference Manual.

db2 => create database cm_db
SQL1032N No start database manager command was issued.  SQLSTATE=57019
db2 => start database manager
DB20000I The START DATABASE MANAGER command completed successfully.
db2 => create database cm_db
DB20000I The CREATE DATABASE command completed successfully.
db2 => connect to cm_db

      Database Connection Information
Database server      = DB2/NT 8.1.0
SQL authorization ID = JSIMMONS
Local database alias = CM_DB

db2 => list active databases

              Active Databases
Database name           = CM_DB
Applications connected currently = 1
Database path           = C:\DB2\NODE0000\SQL00002\

```

Figure 3-31 Command line

11. Next, the window shown in Figure 3-32 opens again.

Here, we select the **Default tablespaces and run schema scripts** (the default is None) so that we will not have to do it at a later time.

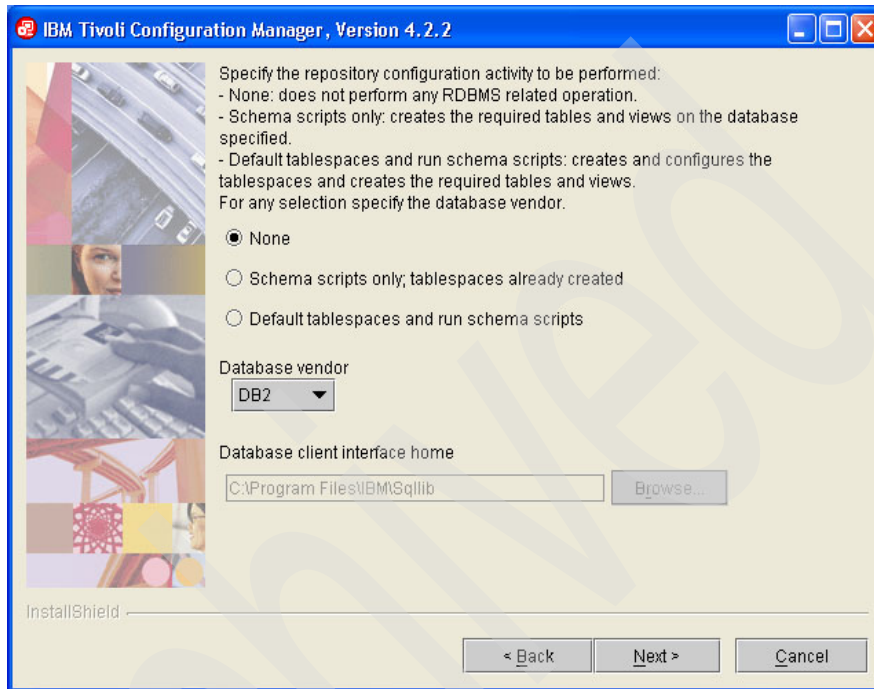
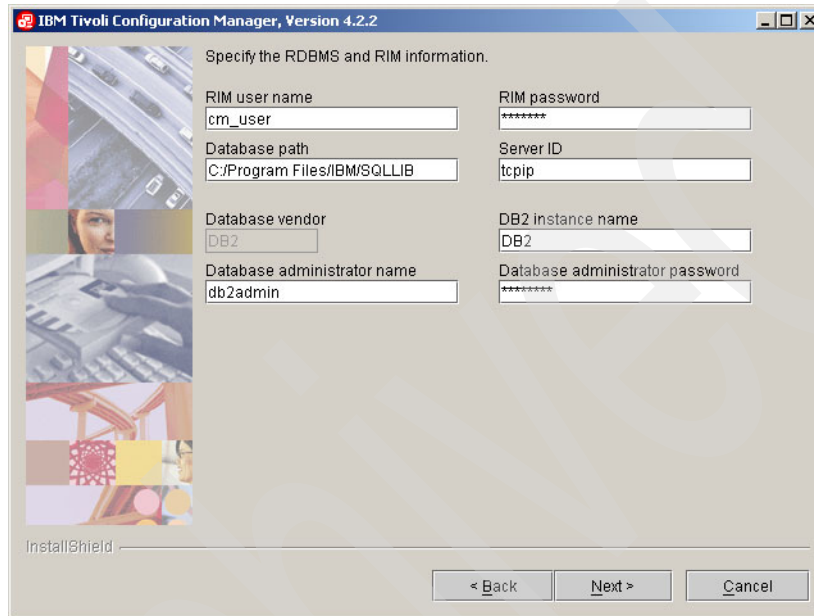


Figure 3-32 Select Default tablespaces and run schema scripts

12. Figure 3-33 is where we begin to create the RIM objects so that the RDBMS database can be accessed by Inventory, MDist 2, and Software Distribution. Make sure that you enter your RIM password, Server ID, and Database administrator password correctly. There is a total of five RIM object windows to set up all the RIM objects.



The screenshot shows a dialog box titled "IBM Tivoli Configuration Manager, Version 4.2.2". The main text reads "Specify the RDBMS and RIM information." The dialog contains several input fields:

RIM user name	cm_user	RIM password	*****
Database path	C:/Program Files/IBM/SQLLIB	Server ID	tcipip
Database vendor	DB2	DB2 instance name	DB2
Database administrator name	db2admin	Database administrator password	*****

At the bottom left, there is a progress indicator for "InstallShield". At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Figure 3-33 RIM configuration

13. If one of the database parameters is wrong, you might receive an error message, as shown in Figure 3-34.

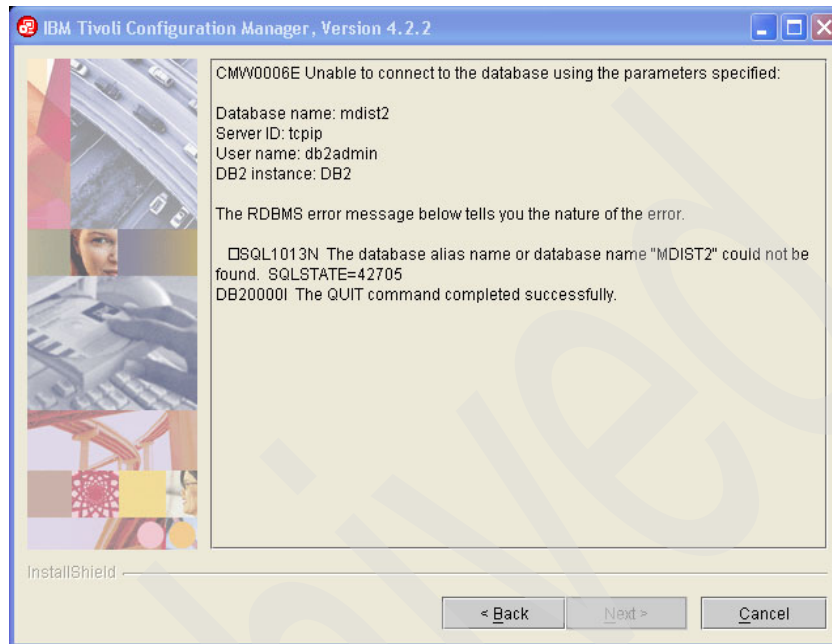


Figure 3-34 Database connection error

We get an error here, because the database name does not conform to the name we gave the database. The correct name of the database is `cm_db`, not `mdist2`. Note that we choose to use one database, `cm_db`, instead of a separate database for each RIM object. We click **Back** to go back to correct the error.

14. The next window prompts for the Activity Planner User name and Password (Figure 3-35). Use a user that is on this system, or you can use the DB2 user ID and password. Here, we use an administrator's ID and password on this computer.

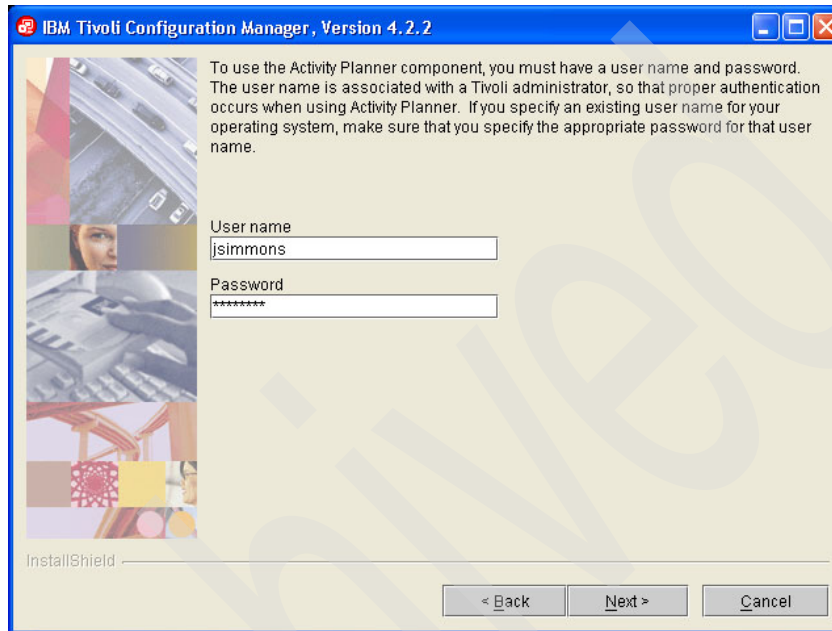


Figure 3-35 Activity Planner User name window

15. In the next window, we set up the LDAP services to be used with Tivoli Configuration Manager (Figure 3-36). We do not use LDAP at XYZ, so we bypass this window.

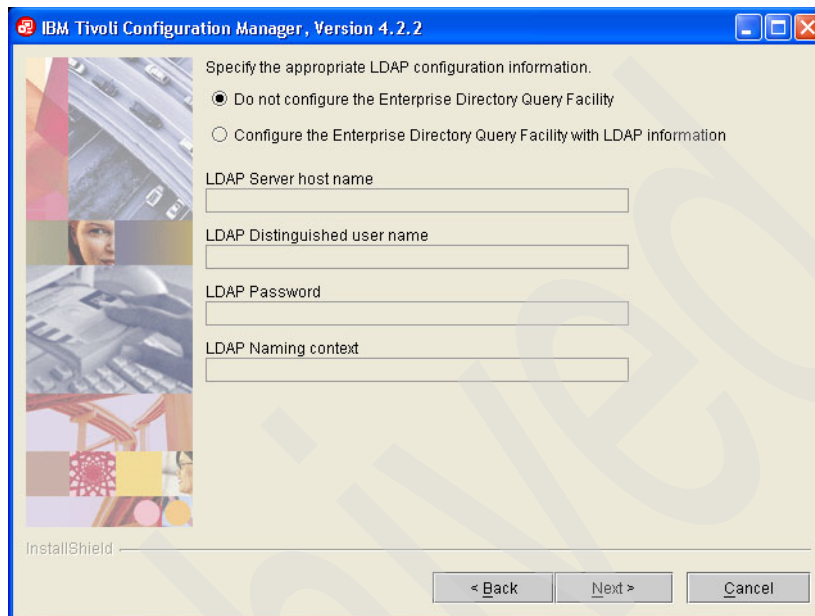


Figure 3-36 LDAP configuration window

16. Figure 3-37 shows the components to install. Because we are performing the custom installation, we can select the components to install here (all components are selected by default). In the XYZ Corporation environment, we do not use the Web User Interface and the Pristine Manager, so clear those options.

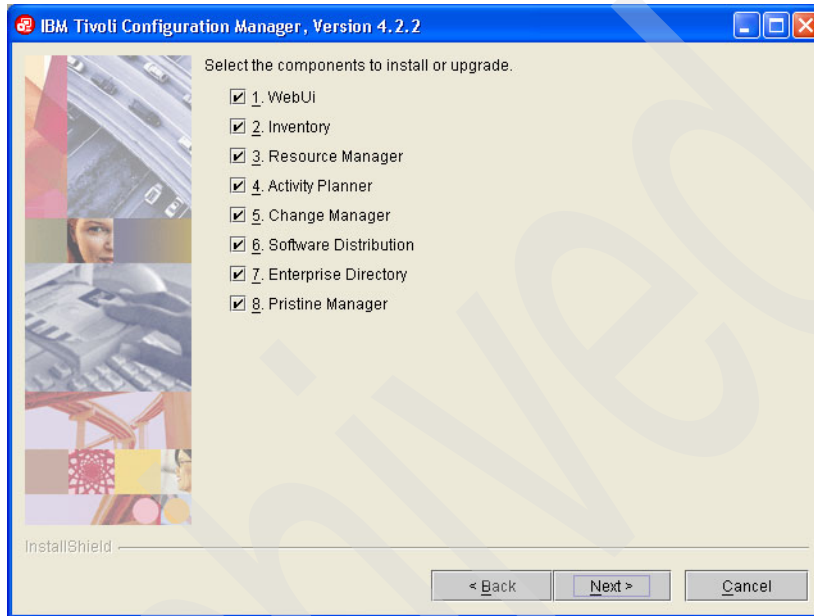


Figure 3-37 Component selection window

17. Figure 3-38 shows the operations that will be performed. (You will get a number of windows like this. Click **Next** to see the other windows.) Take a couple of minutes to go through these windows to make sure that you have entered all the data correctly.

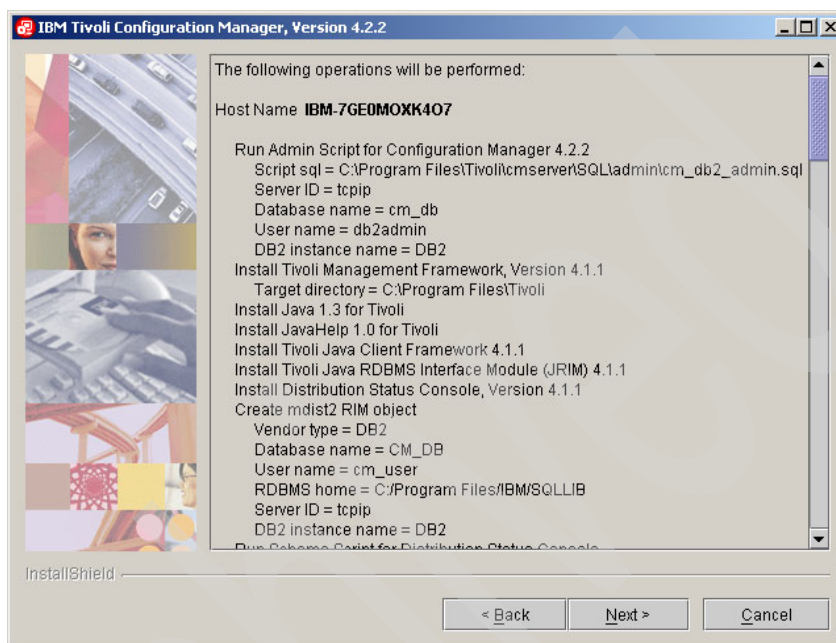


Figure 3-38 Operations that will be performed

18. The next window (Figure 3-39) shows you the installer step list.

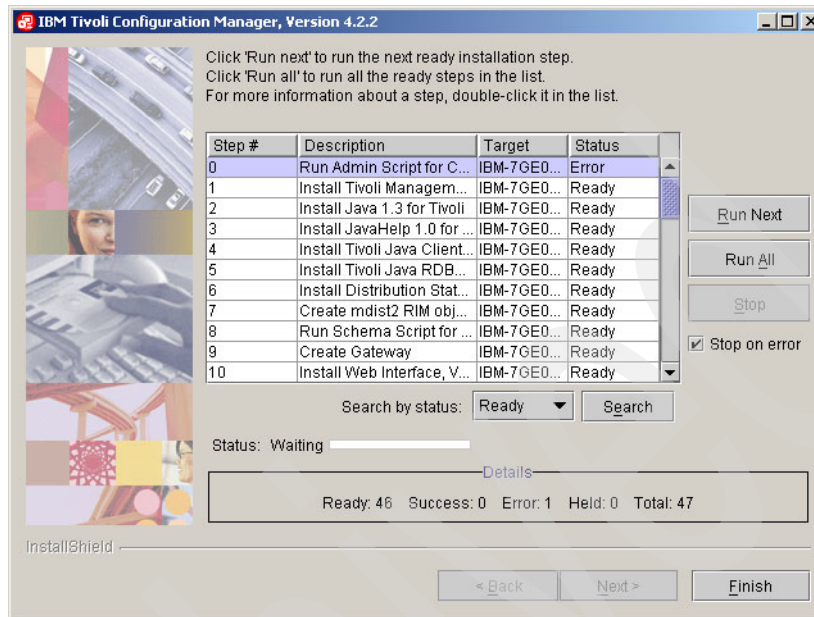


Figure 3-39 Installer step list

As you can see here, there will be 46 different steps to this installation. Usually, you will select **Run All** unless there are some particular problems that you are experiencing. We recommend that you select the **Stop on error** option. A lot of the steps are done in a logical order, and if one step fails, it will be followed by several others. It is best to handle the errors as they occur to prevent other errors further down the step list. In our case, when we clicked **Run All**, we received an error in the first step.

19. Right-click the error line and select **Diagnose error**, and the window shown in Figure 3-40 opens.

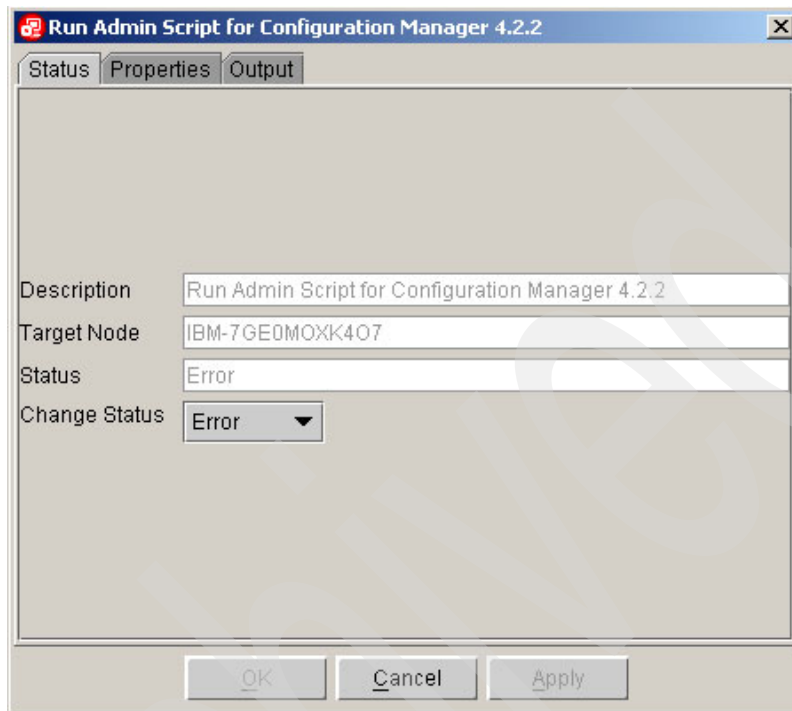


Figure 3-40 Diagnose error window

The first tab here shows us the status. We want to change the status to **Ready** before we rerun this part of the step list. The Properties tab shows what variables are being used and what script is being run. The Output tab shows the log file and the error. From these three tabs, we can usually tell what the error is and correct any issues that might be present.

We can see here from the output that the “validation of admin script failed,” as shown in Figure 3-41.

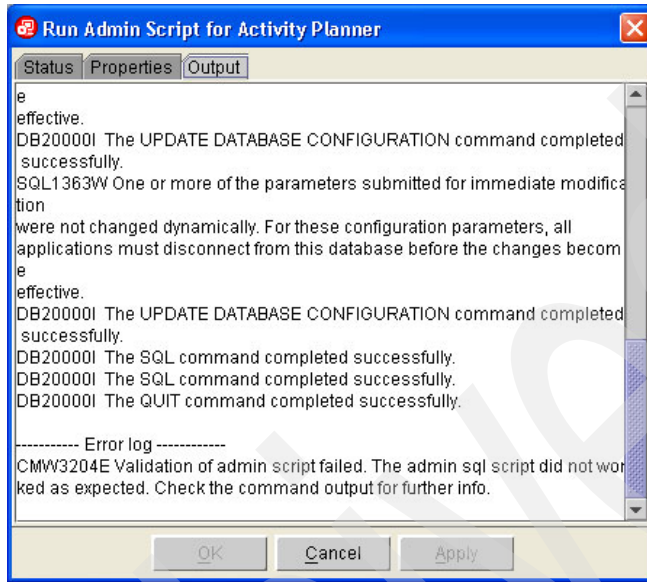


Figure 3-41 Output tab

Looking at Figure 3-41, we can see that there are differences in the password and the password to validate. We change them to match.

20. We go to the Properties tab. Everything else in here looks OK. After changing the passwords to conform to the password of the db2admin user, click **Apply**. See Figure 3-42.

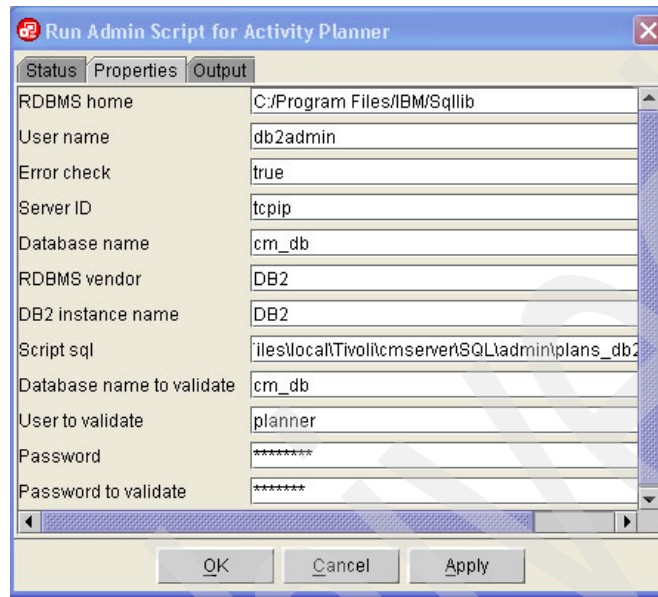


Figure 3-42 Properties tab

21. Then, go to the Status tab and change the status from Error to **Ready**. Then, click out of the box back to the step list window. Go to the Run All tab. The install will restart the first step to completion.

22. After successfully making it past the first step, we are required to reboot the computer (Figure 3-43). This is to pick up any dynamic link libraries (DLLs) that were installed and to set up some users that are required by Tivoli to run the product.

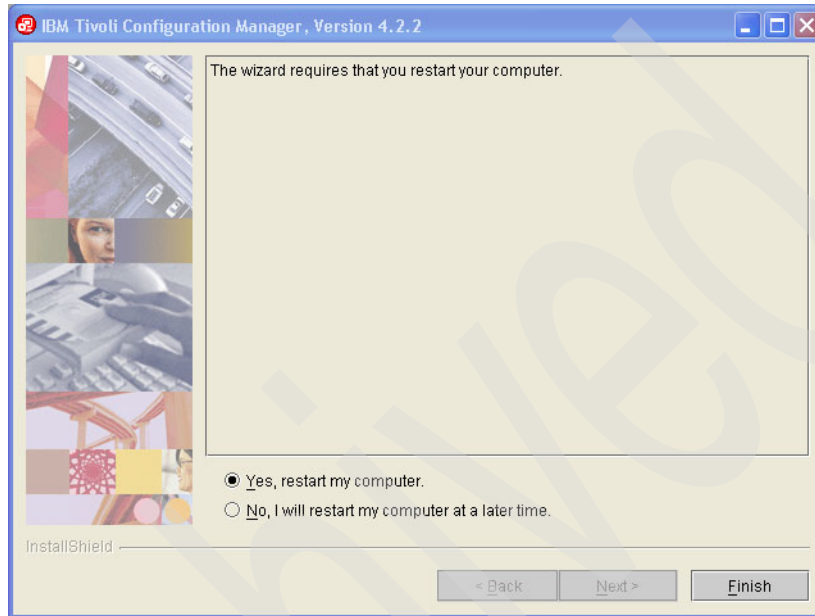


Figure 3-43 Successful Framework install requires reboot

After rebooting, we again get the step list and proceed through the rest of the installation.

23. Figure 3-44 shows another error window that you might encounter.

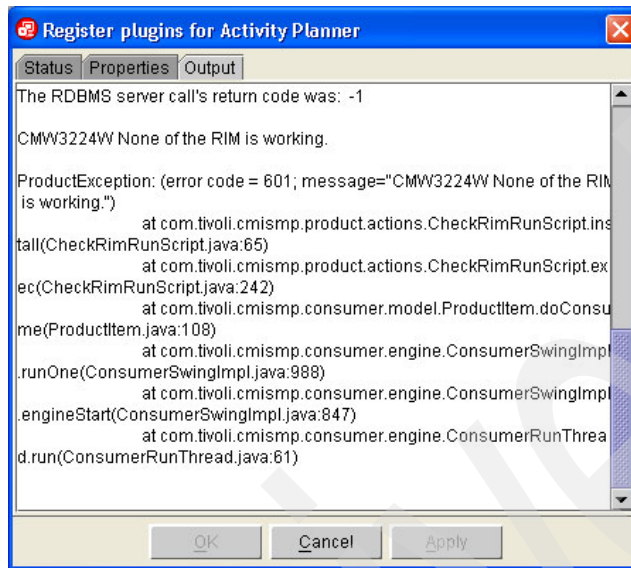
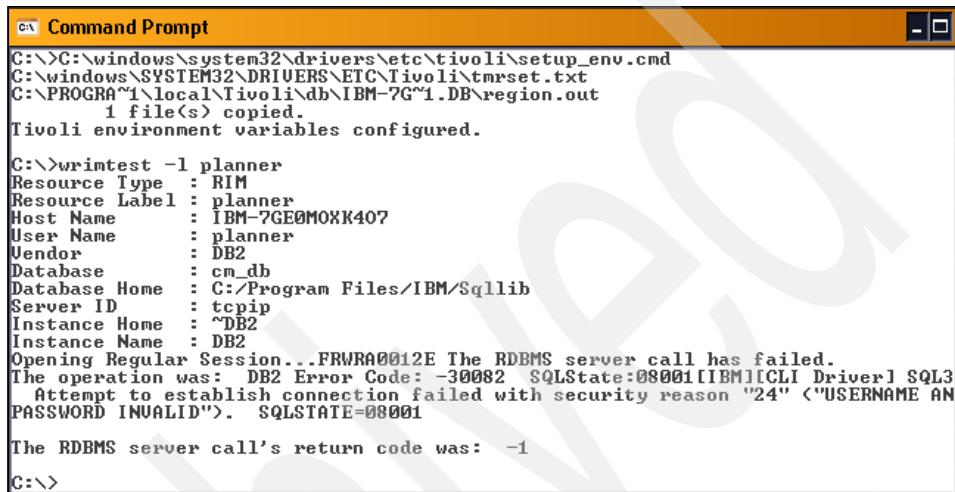


Figure 3-44 RDBMS server call error

As you can see from this window, the Activity Planner is having a problem with the RIM object and the database.

24. By using the command line interface, we can test the RIM object to see if it is functional. We see here that it is not, as shown in Figure 3-45. We also can see the error, "Username and password invalid." By looking here, we see that the user name is planner. We use the `wsetrim -u db2admin` command to set the user to the db2admin user and the `wsetrimpw planner db2admin planner` command to reset the password to the password of the db2admin.



```
C:\>C:\windows\system32\drivers\etc\tivoli\setup_env.cmd
C:\windows\SYSTEM32\DRIVERS\ETC\tivoli\triset.txt
C:\PROGRAM~1\local\tivoli\db\IBM-7G~1.DB\region.out
1 file(s) copied.
Tivoli environment variables configured.

C:\>wrintest -l planner
Resource Type : RIM
Resource Label : planner
Host Name : IBM-7GE0M0XK407
User Name : planner
Vendor : DB2
Database : cm_db
Database Home : C:/Program Files/IBM/SqlLib
Server ID : tcpip
Instance Home : ~DB2
Instance Name : DB2
Opening Regular Session...FRWR00012E The RDBMS server call has failed.
The operation was: DB2 Error Code: -30082 SQLState:08001[IBM][CLI Driver] SQL3
Attempt to establish connection failed with security reason "24" ("USERNAME AN
PASSWORD INVALID"). SQLSTATE=08001

The RDBMS server call's return code was: -1

C:\>
```

Figure 3-45 User name and password invalid error

25. Then, return to the step list and restart that part of the list. The error is now gone.

26. After successfully going through the step list, we can open the administrator GUI using the Tivoli Desktop for Windows that was installed, as shown in Figure 3-46.

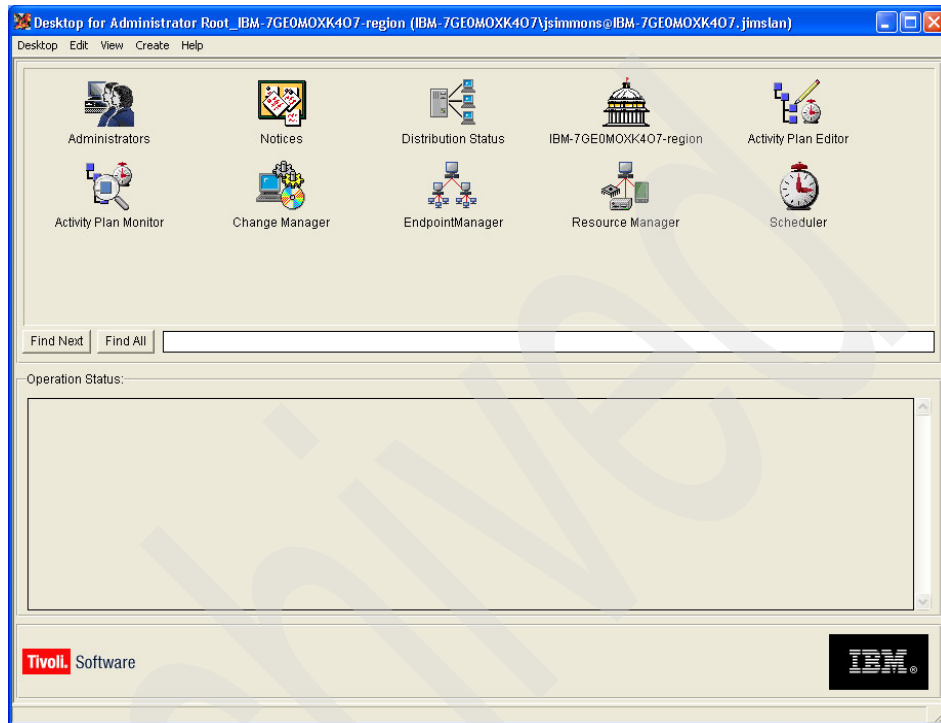


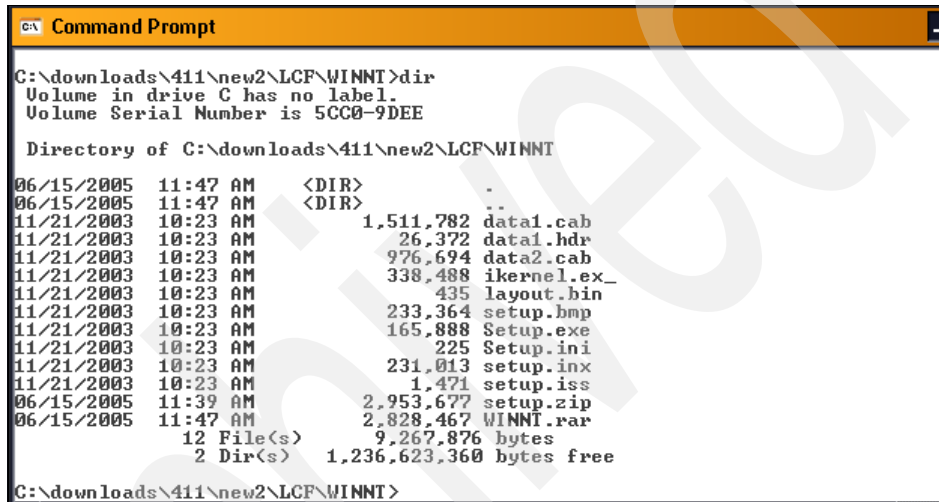
Figure 3-46 Administrator GUI

27. You can see here that we have successfully installed all the products we selected during the installation sequence. Our final operation before installing endpoints is to create a gateway. Right-click the **Endpoint Manager** icon and select **Create Gateway** to create a gateway on the Tivoli server.
28. Enter in the name of the Tivoli server (because we are creating this gateway on the Tivoli server) in the Managed Node Proxy field, enter the name of the gateway in the Name field, and click the **Create & Close** button. By default, the name of the gateway is *managednode_name-gw*, but you can give it any name that you want. At this point, we are ready to install endpoints.

3.5 Endpoint installation

In this section, we explain how to install an endpoint. We install two endpoints on one computer and one on a Microsoft Windows 2000 Server and one on a Linux server. Perform the following steps:

1. Figure 3-47 shows the directory from which we will install the Windows endpoint. The Microsoft Windows implementation of the endpoint is installed here using the **Setup.exe** file and employs the InstallShield technology to install the endpoint.



```
C:\ Command Prompt
C:\downloads\411\new2\LCF\WINNT>dir
Volume in drive C has no label.
Volume Serial Number is 5CC0-9DEE

Directory of C:\downloads\411\new2\LCF\WINNT

06/15/2005  11:47 AM  <DIR>          .
06/15/2005  11:47 AM  <DIR>          ..
11/21/2003  10:23 AM           1,511,782 data1.cab
11/21/2003  10:23 AM             26,372 data1.hdr
11/21/2003  10:23 AM           976,694 data2.cab
11/21/2003  10:23 AM           338,488 ikernel.ex_
11/21/2003  10:23 AM             435 layout.bin
11/21/2003  10:23 AM           233,364 setup.bmp
11/21/2003  10:23 AM           165,888 Setup.exe
11/21/2003  10:23 AM             225 Setup.ini
11/21/2003  10:23 AM           231,013 setup.inx
11/21/2003  10:23 AM             1,471 setup.iss
06/15/2005  11:39 AM       2,953,677 setup.zip
06/15/2005  11:47 AM       2,828,467 WINNT.rar
              12 File(s)          9,267,876 bytes
              2 Dir(s)     1,236,623,360 bytes free

C:\downloads\411\new2\LCF\WINNT>
```

Figure 3-47 Endpoint installation file directory

Note: The first endpoint on a network segment *must* be installed using the Setup.exe file. After that, you can use the **winstlcf** command.

2. Figure 3-48 shows the InstallShield interface opening to direct us through the rest of the installation. Click **Next**.

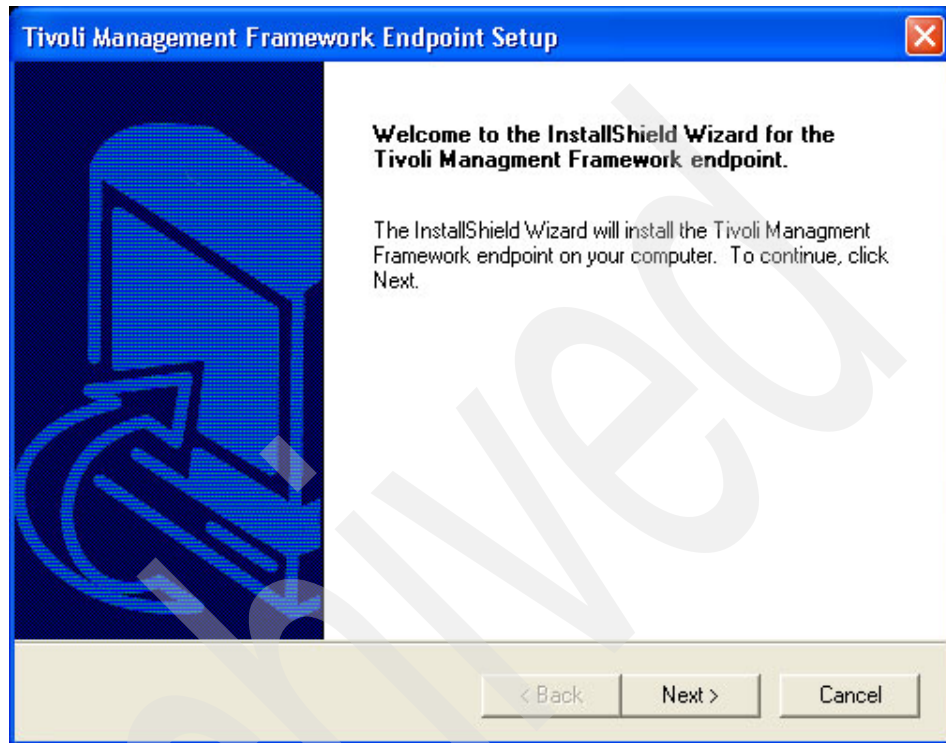


Figure 3-48 InstallShield Wizard

3. In the License Agreement window (Figure 3-49), select **Yes** if you agree to the terms (if you select the **No** button here, the installation of the endpoint will terminate).

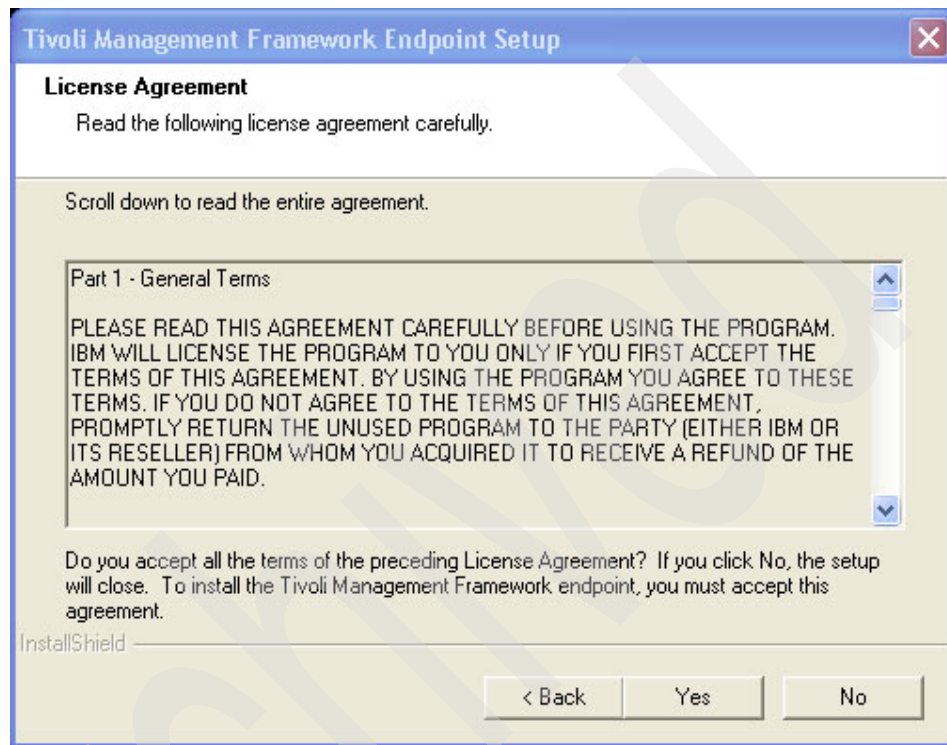


Figure 3-49 License Agreement

4. The next window (Figure 3-50) shows what permissions are required to be in place on the endpoint prior to installation. If these are not adhered to, tasks, scans, and anything else that you might want to do with this endpoint will not be available.

Click **Next**.

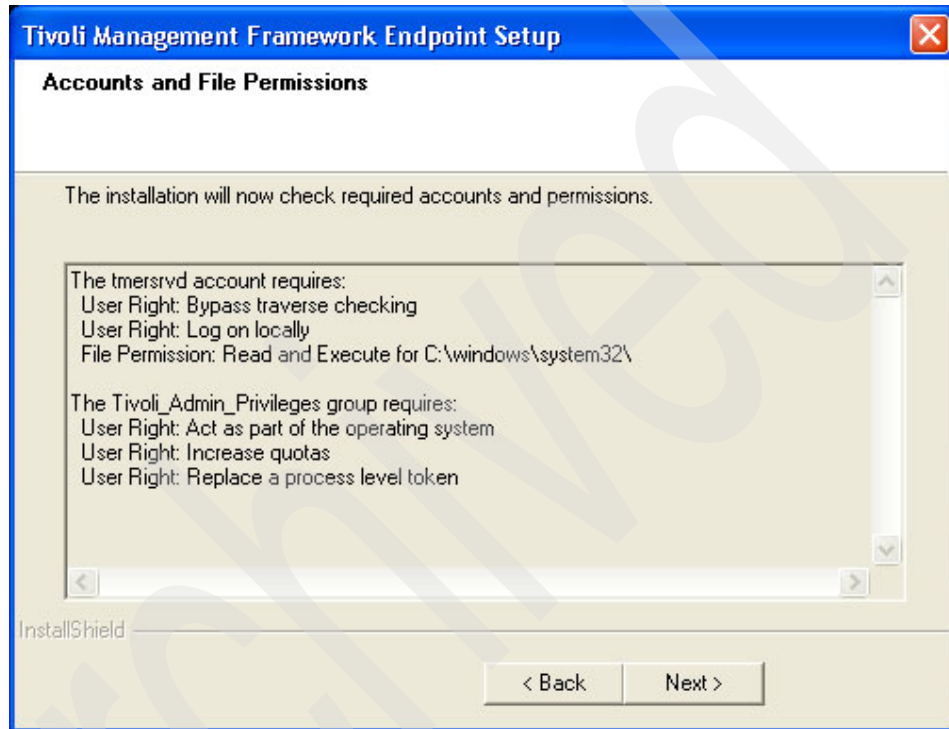


Figure 3-50 Accounts and File Permissions

- In the window shown in Figure 3-51, you can change where the endpoint files will be stored. From time to time, we will use these folders to hold Inventory scan information, tools that we use for other tasks, and so on. As you can see here, we designated a folder that also reflects the port number of the endpoint. Installing more than one endpoint on the same computer requires that you use different ports for each instance of the endpoint. Also note here that the space required is only 2008 kilobytes (KB). As we mentioned in 3.2, “Description of the environment” on page 56, this number will grow depending on what other Tivoli products you install and implement in your Tivoli environment. Products such as IBM Tivoli Monitoring and Tivoli Enterprise Console will increase this space, though marginally.

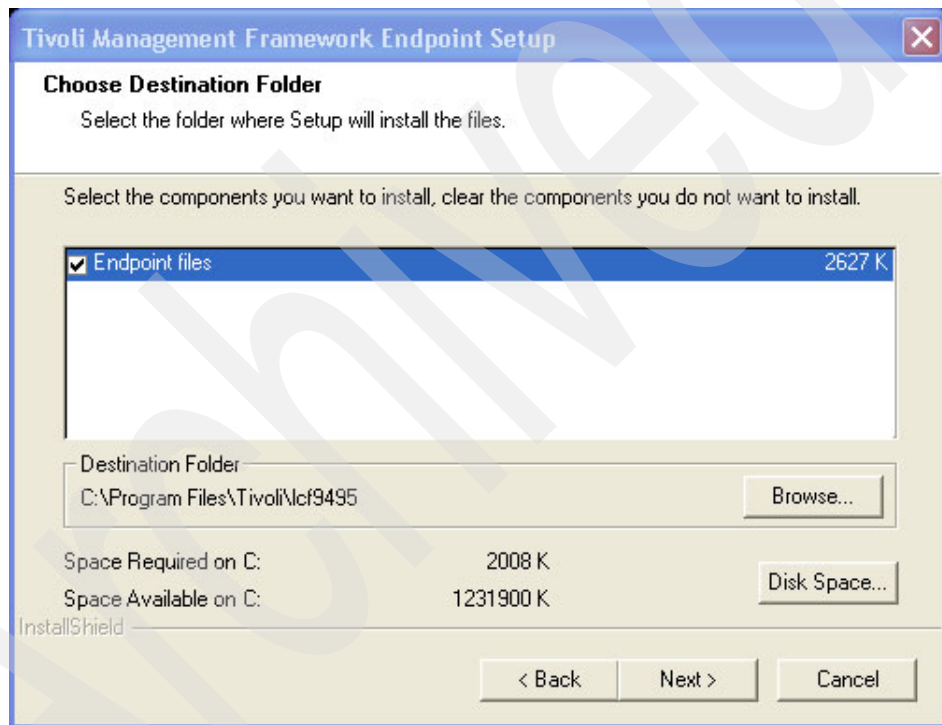


Figure 3-51 Destination Folder

6. If your environment requires that your endpoints have access to another system within the network and it is a shared resource, you need to fill out the appropriate information, as shown in Figure 3-52. If you currently do not have a need for this, leave it blank. This is commonly referred to as the TRAA account (Tivoli remote access account). You can use this process later by issuing commands from the command line interface (CLI).

Tivoli Remote Access Account

Provide Remote File Access Information

Enter a user name and password for Tivoli remote access account.

NOTE: This information will be used when accessing shared resources on other networked systems. Refer to the wlictap documentation for more information.

User

Password

InstallShield

< Back Next > Cancel

Figure 3-52 Tivoli Remote File Access

7. Figure 3-53 shows the Advanced Settings window. We know the gateway port is 9494, and on this endpoint, we designated 9495 as the port the endpoint will use. We also designated the `-g` command with the name of the gateway (sometimes, it is advantageous to use the IP address of the gateway instead of the label name), the port of the gateway (`+9494`), and the port of the endpoint (`9495`), and we give the endpoint a label (`-n IBM-7GE0M0XK407-ep1`). This will make it easier to distinguish one endpoint from another. Framework will use the host name, because the default is that no label is designated.

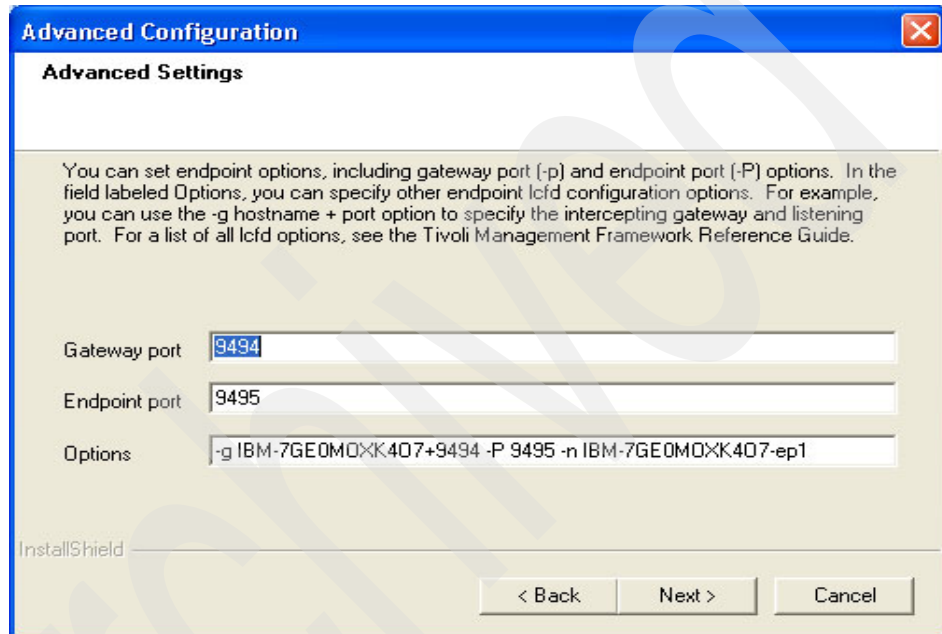


Figure 3-53 Advanced Settings window

8. Figure 3-54 informs us that there is now enough information to start copying the files to the endpoint. At this point, you can review what you have entered by going back one window or proceed by clicking **Next**.

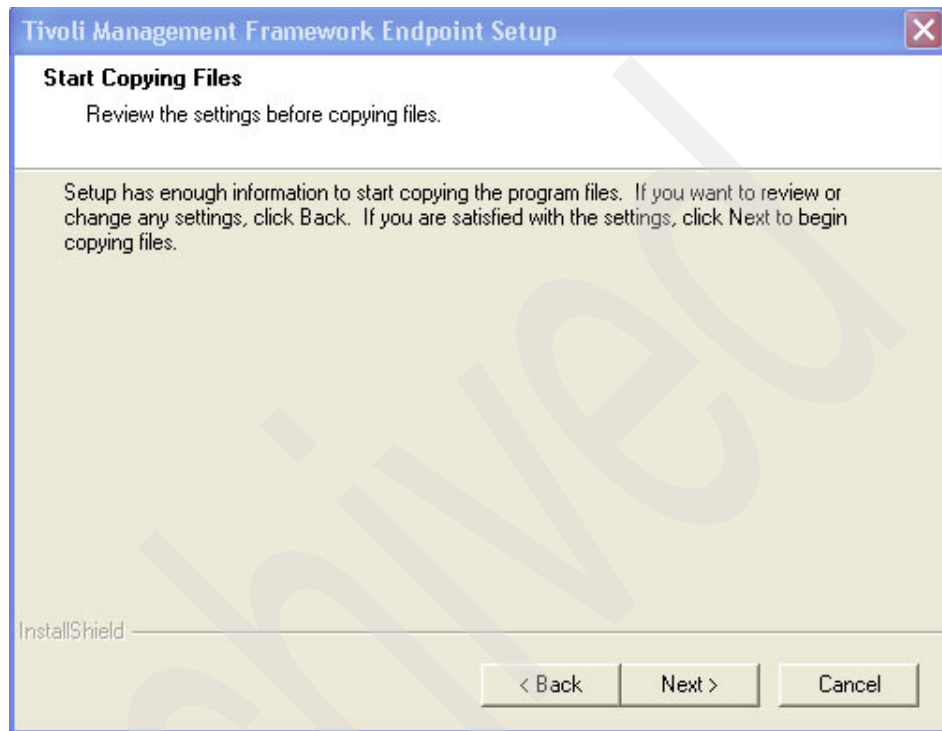


Figure 3-54 Start Copying Files

- In Figure 3-55, we see that the InstallShield Wizard has completed the setup. At the same time, the endpoint engine (lcfed.exe) has started and will contact the gateway. The gateway will, in turn, contact the endpoint manager on the Tivoli server and run a series of scripts to complete the login of the endpoint and register it so that we can do further work with the endpoint, such as inventory scans and software distributions.

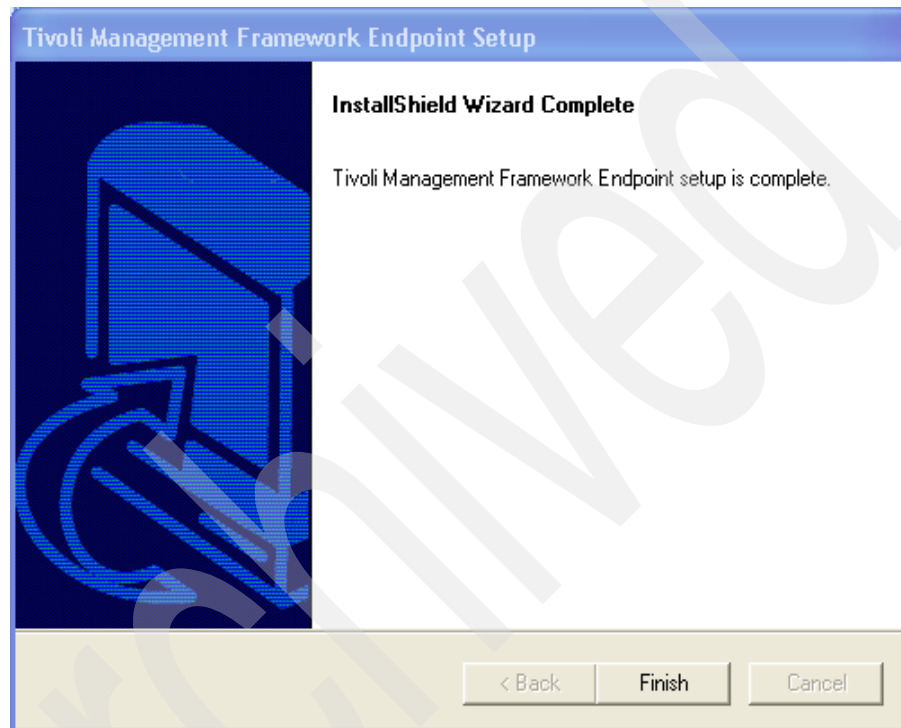


Figure 3-55 Endpoint setup is complete

- If we then go to a command prompt and set up our environment “called sourced” using the following command, we can run some of the commands used to work within the Tivoli environment:

```
c:\WINNT\System32\drivers\etc\Tivoli\setup_env.sh
```

At this point, we can also go to a bash shell that will enable us to use all the commands in *Tivoli Management Framework Reference Manual, Version 4.1.1, SC32-0806*.

- As shown in Figure 3-56, we “sourced” the environment, so we can issue the command `wep ls`. This shows us that the endpoint that we just installed is registered, and we now can use the Tivoli interface to interact with this endpoint.



```
C:\>wep ls
G      1037603094.1.590  IBM-7GE0M0XK407-gw
1037603094.3.522+#TMP_Endpoint::Endpoint# IBM-7GE0M0XK407-ep1
1037603094.4.522+#TMP_Endpoint::Endpoint# Vmwin2000k-ep
1037603094.5.522+#TMP_Endpoint::Endpoint# Linux-ep
1037603094.6.522+#TMP_Endpoint::Endpoint# IBM-7GE0M0XK407-ep2
C:\>
```

Figure 3-56 The `wep ls` command

- On all Microsoft Windows machines, we can install using the `Setup.exe` command if we have physical access to the computer. In most cases, we do not have physical access to the computer, so we install the endpoint remotely. All of the permissions as stated previously for users and accounts will apply to every Microsoft Windows computer that we have.

- If you use the command line interface (CLI) to install these other endpoints, issue the `winstlcf` command.

A typical command looks like this:

```
winstlcf -d "C:\Program Files\Tivoli\lcf9497" -g IBM-7GE0M0XK407-gw+9494 -l 9497 -n VmWin2000k-ep -N IBM-7GE0M0XK407-ep1 win2003k Administrator
```

Where the host name of the machine is `win2003k`, the user that we will use for the installation is `Administrator`, and the `-N` option enables us to use the first Windows machine as a proxy machine. This enables us to put on Tivoli Remote Execution Service or some files so that we can use `rlogin` and `reexec` on the Windows machines.

- Similarly, we issue the following command to install to the Linux computer with the host name `RHLinux`, to the directory `/opt/Tivoli/lcf`, port `9498`, and using the user `Root` to install:

```
winstlcf -d /opt/Tivoli/lcf -g IBM-7GE0M0XK407-gw+9494 -l 9498 -n Linux-ep RHLinux Root
```

15. The install will then ask for a password, and if you are authenticated on the remote computer, there will be a list of files that the process will ask to download. Answering **Yes** will download the files and start the lcfcd process to log in to the gateway.
16. The next screen (Figure 3-57) shows that we have installed all the endpoints successfully.



```
C:\>wep ls
G      1037603094.1.590  IBM-7GE0M0XK407-gw
1037603094.3.522+#TMF_Endpoint::Endpoint#  IBM-7GE0M0XK407-ep1
1037603094.4.522+#TMF_Endpoint::Endpoint#  UMwin2000k-ep
1037603094.5.522+#TMF_Endpoint::Endpoint#  Linux-ep
1037603094.6.522+#TMF_Endpoint::Endpoint#  IBM-7GE0M0XK407-ep2

C:\>bash
bash$ wlookup -ar Endpoint
IBM-7GE0M0XK407-ep1      1037603094.3.522+#TMF_Endpoint::Endpoint#
IBM-7GE0M0XK407-ep2      1037603094.6.522+#TMF_Endpoint::Endpoint#
Linux-ep                1037603094.5.522+#TMF_Endpoint::Endpoint#
UMwin2000k-ep          1037603094.4.522+#TMF_Endpoint::Endpoint#
bash$
```

Figure 3-57 The `wlookup -ar Endpoint` command

17. In Figure 3-58, we can see if the endpoints are registered in the database using the `wlookup -ar Endpoint` command.

Another way that we can look at the endpoints is through the Tivoli desktop GUI. Figure 3-58 shows us the same endpoints as shown in Figure 3-57 on page 109.

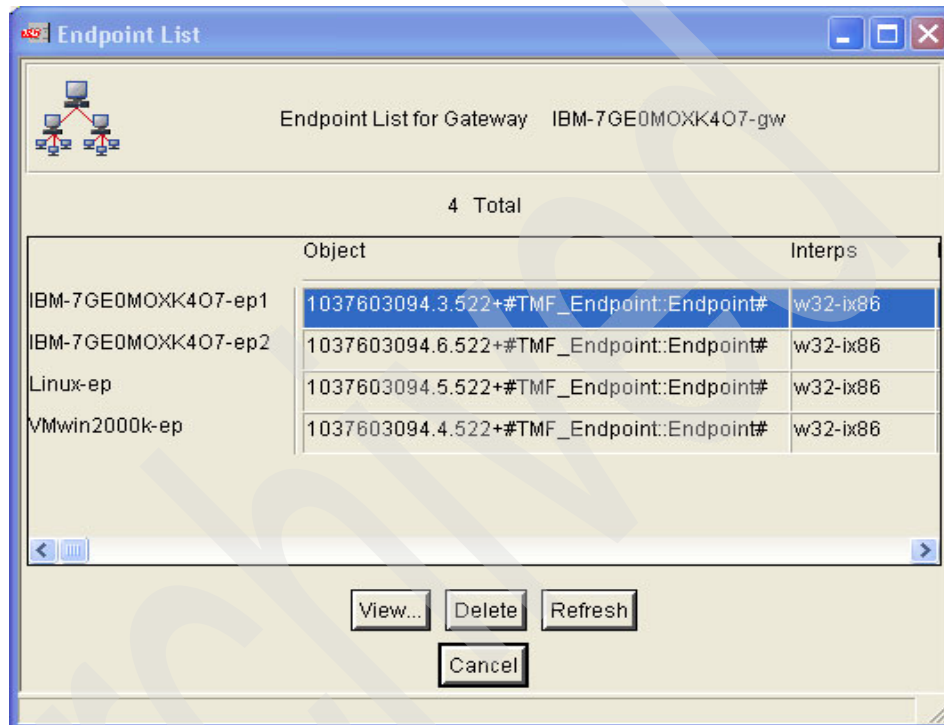


Figure 3-58 Endpoint List

At this point, we have a functional Tivoli server and gateway and four functional endpoints. That concludes our installation for the XYZ Corporation.

Installation of IBM Tivoli Configuration Manager in a large enterprise

In this chapter, we first introduce you a fictitious company, called ABBC. We use ABBC's IBM Tivoli Configuration Manager deployment as a case study and provide all the needed steps in order to implement IBM Tivoli Configuration Manager in ABBC's test environment, which is a representative of the whole company.

We discuss the following topics:

- ▶ The ABBC Company
- ▶ Installation of Tivoli Configuration Manager V4.2.2 server on AIX 5L
- ▶ Configuring DB2
- ▶ Desktop installation
- ▶ Installing a gateway on Linux
- ▶ Installing Tivoli Configuration Manager components on new gateway
- ▶ Installing endpoints

4.1 The ABBC Company

Armando Brothers Banking Corp. (ABBC) is a financial institution that traces its history back to the early days of industrialization. It was a time of radical change and growing financing needs. The Armando brothers were open to new ideas and founded a bank situated in Austin, Texas to help pioneers of those days to finance their business ventures. ABBC, a bank very open to new technologies, expanded rapidly and became one of the most important banks with a tradition in retail and private banking offering customized services for customers around the world. One reason for their success is their ability to explore different markets with specialized teams. ABBC is represented in all continents and all major cities with at least one office.

ABBC soon began to elaborate the new business opportunities offered by electronic banking. They were among the first banks offering their customers online access to their accounts. Additionally, other user groups such as business partners and subsidiaries gained access to their computing environment.

ABBC's current architecture has grown along with the evolution in IT technology.

Note: All names and references to company and other business institutions used in this chapter are purely fictional. Any match with a real company or institution is coincidental.

4.2 IBM Tivoli Configuration Manager implementation at ABBC

ABBC decided to improve the management of their business, and to do that, they decided to implement Tivoli Configuration Manager to get better control of the hardware and software management.

The ABBC company has more than 20,000 Microsoft Windows and Linux desktops and approximately 2000 AIX 5L and Linux servers in their 1,200 branches. There are also 20 IBM AS/400® systems at their headquarters. Before deploying Tivoli Configuration Manager to the branches, ABBC decided to test Tivoli Configuration Manager in a test environment. The test environment will be representation of a typical ABBC branch, with one gateway and four endpoints. One Tivoli server will be installed centrally to manage this environment. Figure 4-1 on page 113 shows the initial environment and which products must be installed on each machine.

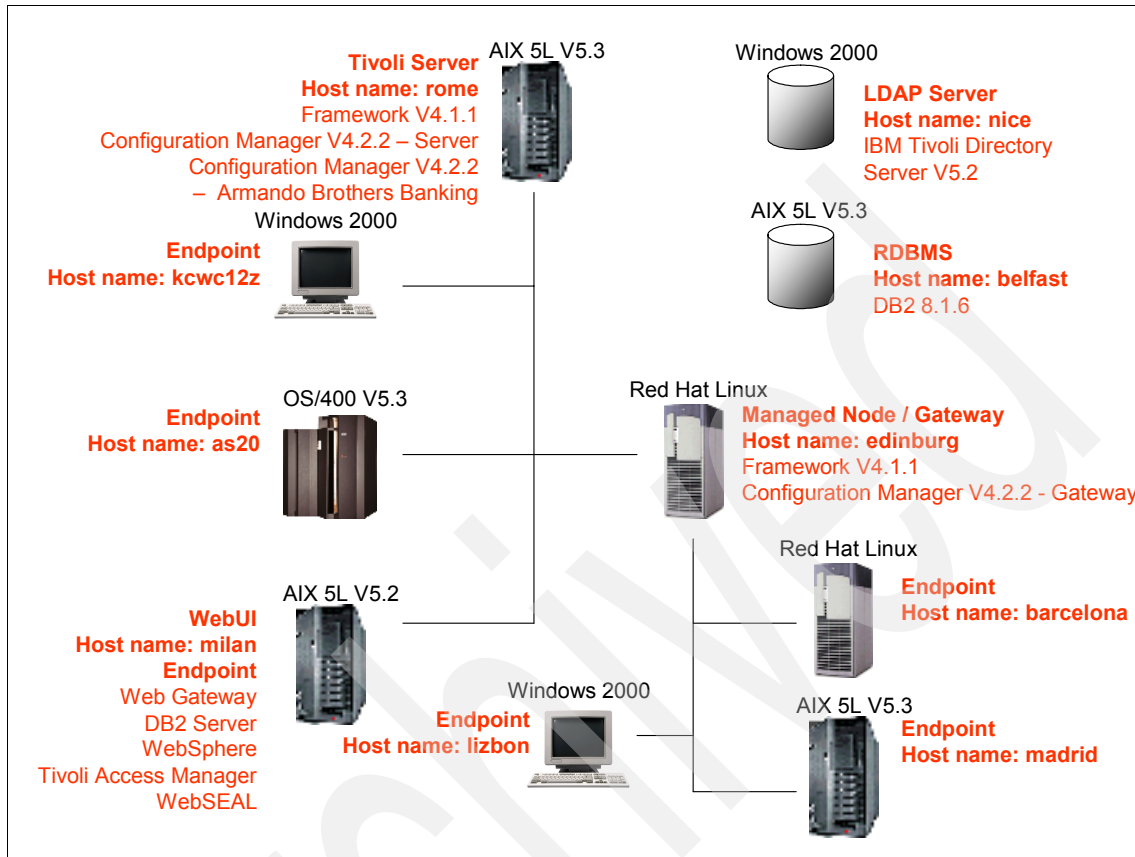


Figure 4-1 ABBC topology

4.3 Installation of Tivoli Configuration Manager V4.2.2 server on AIX 5L

According to Figure 4-1, we install the Tivoli server on the AIX 5L machine called rome. This section describes how to do that.

4.3.1 General prerequisites for the installation

Table 4-1 on page 114 describes the hardware and software prerequisites for the installation of a Tivoli server. It is very important to check the prerequisites before installing Tivoli Configuration Manager V4.2.2.

Table 4-1 Check before installing Tivoli Configuration Manager V4.2.2

Items to be checked	Requirements
Disk space	<ul style="list-style-type: none"> ▶ 120 MB of free space on /tmp for the installation program. ▶ 750 MB of free space to install Tivoli Configuration Manager V4.2.2.
AIX V4.3.3	Install maintenance level 4330-09. Run the <code>oslevel -r</code> command to check it. We recommend that you upgrade your AIX V4.3.3 to an AIX 5L 5.x version once the 4.3.3 version is no longer supported by IBM.
AIX 5L V5.1.0	Install the APAR IY19375 and the maintenance level 5100-01. Run the commands <code>instfix -ik IY19375</code> and <code>oslevel -r</code> to check them.
AIX 5L V5.2.0	No maintenance is needed.
AIX 5L V5.3.0	No maintenance is needed.
DNS	A good DNS server or a good set of hosts files will be an absolute requirement. Tivoli requires both forward and reverse lookup on IP address and host name.
RDBMS	A RDBMS server and client, if necessary, must be installed and configured. Chapter 5, "Extra components" on page 217 describes these procedures.

4.3.2 The server installation

This section provides the steps to start the Tivoli Configuration Manager V4.2.2 installation on the same machine using the InstallShield wizard.

Preparing for the installation

All the steps necessary for the installation will be described in this section. Before continuing, make sure that you have the following images:

- ▶ IBM Tivoli Configuration Manager Installation, Version 4.2.2
- ▶ IBM Tivoli Configuration Manager Server, Version 4.2.2
- ▶ Tivoli Management Framework 1 of 2, Version 4.1.1
- ▶ Tivoli Management Framework 2 of 2, Version 4.1.1

Database preparation

The server installation can run the admin and schema scripts to configure the RDBMS during the installation process.

Before installing Tivoli Configuration Manager, you have to prepare your RDBMS server to store the database and tables that are created during the server installation. However, you also have the option of running the admin and schema scripts manually. You can do this by selecting **None** in the RDBMS configuration window of the server installation program.

You can also choose to have only one database created for all the components of Tivoli Configuration Manager or have one database for each one. Consult to your DBA before continuing with the process.

In our case, we choose to use only one database, called `cm_db`, and to run the the admin and schema scripts during the server installation process.

RDBMS server configuration

Perform the following steps to configure your RDBMS server before starting the server installation. The RDBMS server must be installed before following these steps. Section 5.1, “IBM DB2 UDB installation” on page 218 describes the installation process.

1. Log in to the RDBMS server as the root user.
2. Create a file system called `/home/db2inst1` with 1.8 GB to store the `db2inst1` instance:

```
crfs -v jfs -g rootvg -m /home/db2inst1 -p'rw' -a size='1800M' -a  
frag='4096' -a nbpi='4096' -a ag='8' -A'yes'
```

3. Mount the file system `/home/db2inst1`:

```
mount /home/db2inst1
```

4. Create the `db2grp1` group:

```
mkgroup -A db2grp1
```

5. Create the `db2fgrp1` group:

```
mkgroup -A db2fgrp1
```

6. Create the `db2inst1` user:

```
mkuser pgrp='db2grp1' home='/home/db2inst1' db2inst1
```

7. Create the `db2fenc1` user:

```
mkuser pgrp='db2fgrp1' db2fenc1
```

8. Set the password for the users `db2inst1` and `db2fenc1`, as described in Example 4-1 on page 116.

Example 4-1 Setting the password for the users db2inst1 and db2fenc1

```
# passwd db2inst1
Changing password for "db2inst1"
db2inst1's New password:*****
Enter the new password again:*****
# pwdadm -c db2inst1

# passwd db2fenc1
Changing password for "db2fenc1"
db2fenc1's New password:*****
Enter the new password again:*****
# pwdadm -c db2fenc1
```

9. Create the instance dn2inst1:

```
/usr/opt/db2_08_01/instance/db2icrt -u db2fenc1 db2inst1
```

10. Check if the following lines were included in the /etc/services file; the port number can be different:

```
DB2_db2inst1      50000/tcp
DB2_db2inst1_1    50001/tcp
DB2_db2inst1_2    50002/tcp
DB2_db2inst1_END  50003/tcp
```

11. Log in to the RDBMS as the db2inst1 user, the owner of the instance.

12. Set the value DB2_db2inst1 to the SVCENAME variable:

```
db2 update dbm cfg using SVCENAME DB2_db2inst1
```

13. Set the DB2COMM variable as tcpip:

```
db2set DB2COMM=tcpip
```

14. Start the db2service:

```
db2start
```

15. Create the database cm_db:

```
db2 create database cm_db
```

16. The users invtiv, planner, tivoli, mdstatus, and pristine need be created on the RDBMS server, all of them with the password tivoli, as shown in Example 4-2 on page 117.

Example 4-2 Creating the users invtiv, planner, tivoli, mdstatus, and pristine

```
# mkuser invtiv
# passwd invtiv
Changing password for "invtiv"
invtiv's New password:*****
Enter the new password again:*****
# pwdadm -c invtiv

# mkuser planner
# passwd planner
Changing password for "planner"
planner's New password:*****
Enter the new password again:*****
# pwdadm -c planner

# mkuser tivoli
# passwd tivoli
Changing password for "tivoli"
tivoli's New password:*****
Enter the new password again:*****
# pwdadm -c tivoli

# mkuser mdstatus
# passwd mdstatus
Changing password for "mdstatus"
mdstatus's New password:*****
Enter the new password again:*****
# pwdadm -c mdstatus

# mkuser pristine
# passwd pristine
Changing password for "pristine"
mdstatus's New password:*****
Enter the new password again:*****
# pwdadm -c mdstatus
```

Now your RDBMS server is configured. Follow the steps described in “RDBMS client configuration” on page 117 if your RDBMS was not installed on the system that will host the Tivoli server. In our case, we installed the RDBMS server on the belfast system and the Tivoli server was installed on the rome system.

RDBMS client configuration

The DB2 client must be installed before following these steps. “Installing the DB2 UDB client on AIX 5L” on page 229 describes its installation process.

Perform the following steps:

1. Create the user db2inst1, as shown in Example 4-3.

Example 4-3 Creating the db2inst1 user on the RDBMS client

```
# mkuser db2inst1
# passwd db2inst1
Changing password for "db2inst1"
db2inst1's New password:*****
Enter the new password again:*****
# pwdadm -c db2inst1
```

2. Create a client instance, associating it with the db2inst1 user:

```
/usr/opt/db2_08_01/instance/db2icrt -s client db2inst1
```

3. Log in the RDBMS client as the db2inst1 user.
4. Catalog the client node named rome for the server node named belfast through the server port 60000. This port is defined in the /etc/services file of the server node belfast.

```
db2 catalog tcpip node rome remote belfast server 60000
```

5. Catalog the database cm_db to be known for the client node rome:

```
db2 catalog database cm_db at node rome authentication server
```

Now your RDBMS server and client are configured. The communication can be tested, as shown in Example 4-4.

Example 4-4 Testing RDBMS communication between the client and server

```
$ db2 connect to cm_db user db2inst1 using db2inst1
```

Database Connection Information

```
Database server      = DB2/6000 8.1.2
SQL authorization ID = DB2INST1
Local database alias = CM_DB
```

Operating system preparation

Perform the following steps to start the installation process:

1. Log in to AIX 5L as the root user.
2. Create a file system with 750 MB for the Tivoli Configuration Manager binaries. Tivoli Configuration Manager V4.2.2 will be installed on the /Tivoli file system. We created this file system using the following command:

```
crfs -v jfs -g rootvg -m'/Tivoli' -p'rw' -a size='750M' -a frag='4096' -a
nbpi='4096' -a ag='8' -A'yes'
```

3. Mount the file system /Tivoli:

```
mount /Tivoli
```

4. Mount the *IBM Tivoli Configuration Manager Installation, Version 4.2.2* CD on AIX 5L and run the following command:

```
/cdrom/setup_aix.bin
```

Now, the InstallShield wizard will be initialized. Next, we describe the steps to perform the installation.

Server installation

The server InstallShield wizard installs Tivoli Management Framework and installs the components of Tivoli Configuration Manager V4.2.2.

Perform the following steps:

1. Choose your language, as shown in Figure 4-2, and click **OK**.

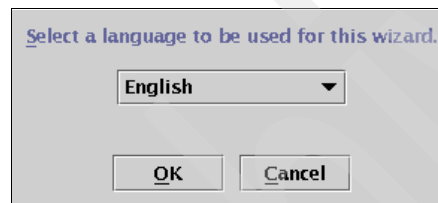


Figure 4-2 Selecting the language

2. Click **Next**, as shown in Figure 4-3, to start the installation.

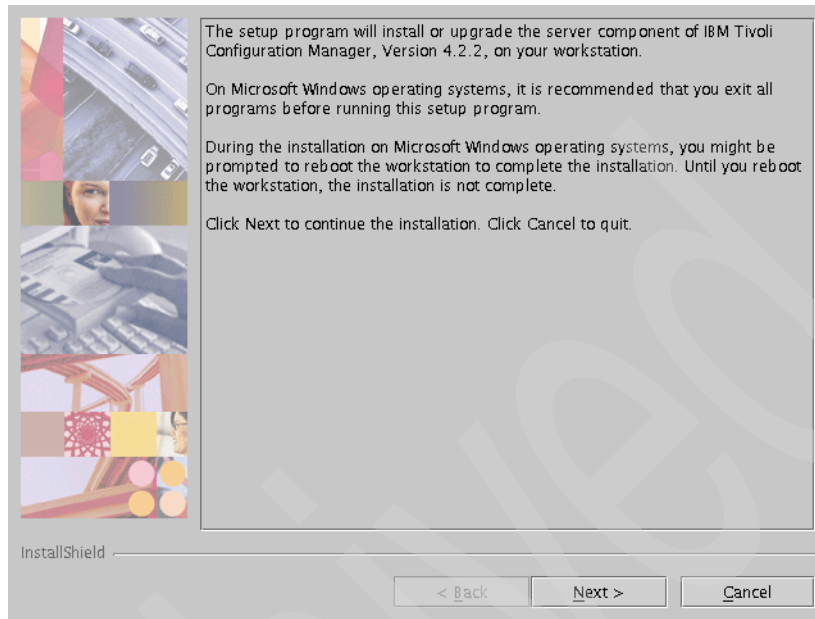


Figure 4-3 InstallShield information window

3. Read and accept the software license agreement, as shown in Figure 4-4. Click **Next** to continue.

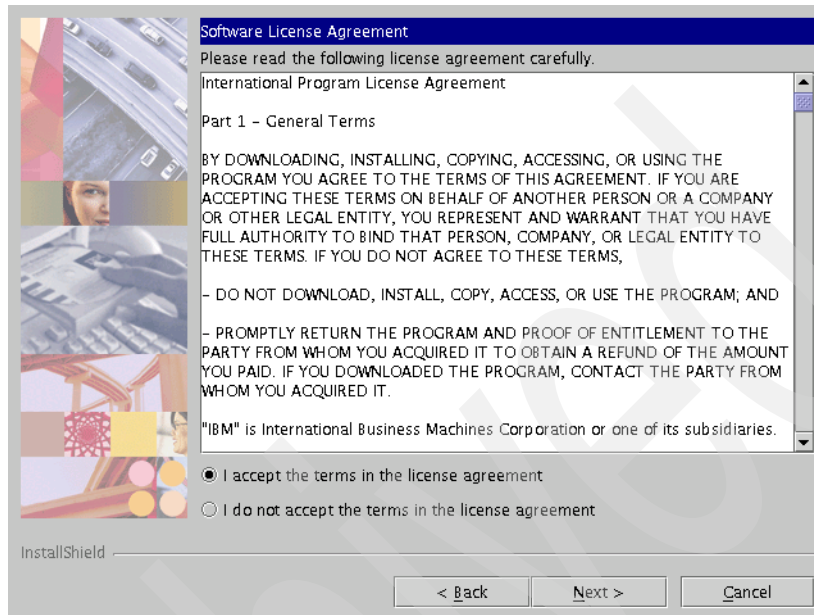


Figure 4-4 Software License Agreement

4. Select the Destination directory for installation, as shown in Figure 4-5, and click **Next** to continue. This directory must be the same one that was created in “Operating system preparation” on page 118.

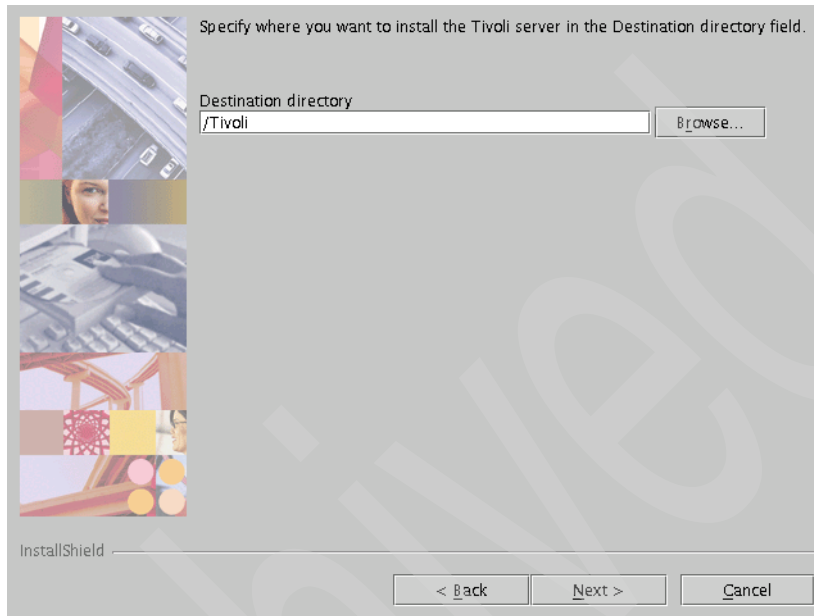


Figure 4-5 Selecting the Destination directory

5. Choose the type of installation, as shown in Figure 4-6, and click **Next** to continue. We select the **Custom** installation.

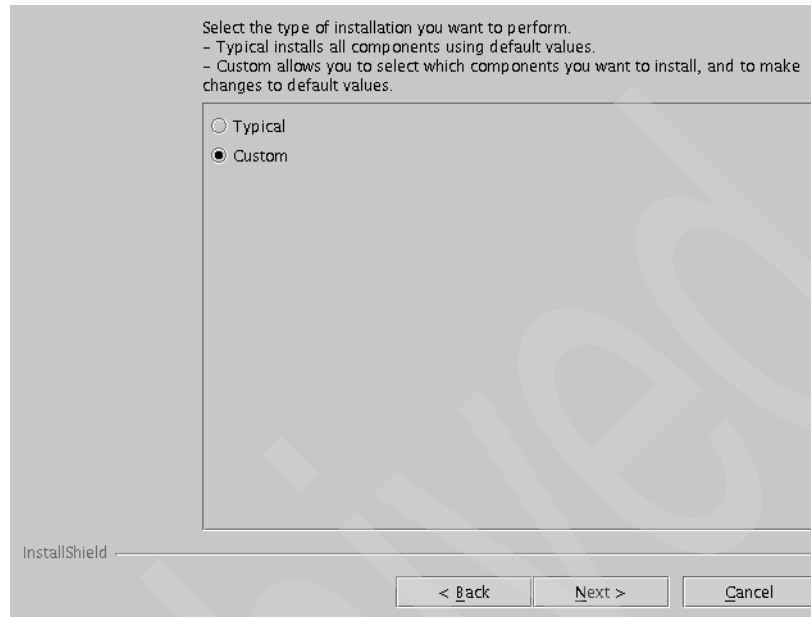


Figure 4-6 Type of installation

6. Select the Tivoli Configuration Manager components that you want to install, as shown in Figure 4-7, and click **Next** to continue.

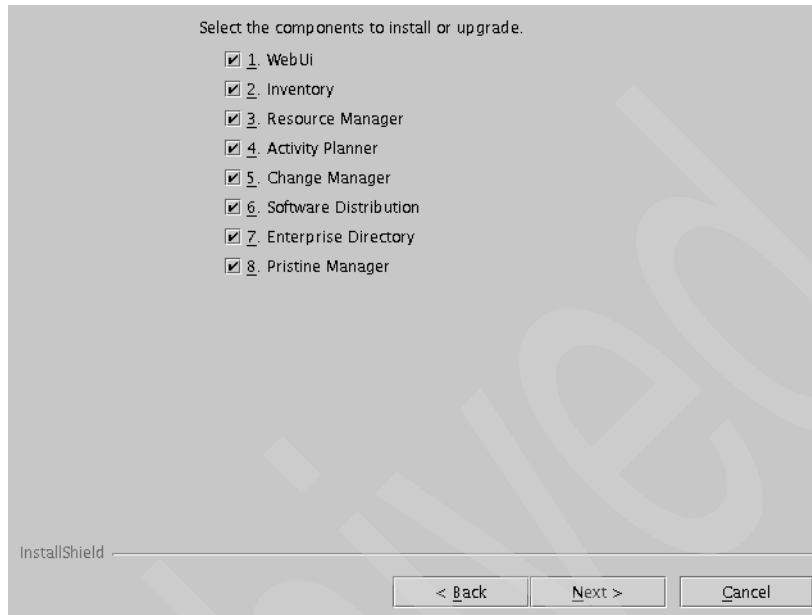


Figure 4-7 Tivoli Configuration Manager components

7. Select any other language if you want in Figure 4-8 and click **Next** to continue.

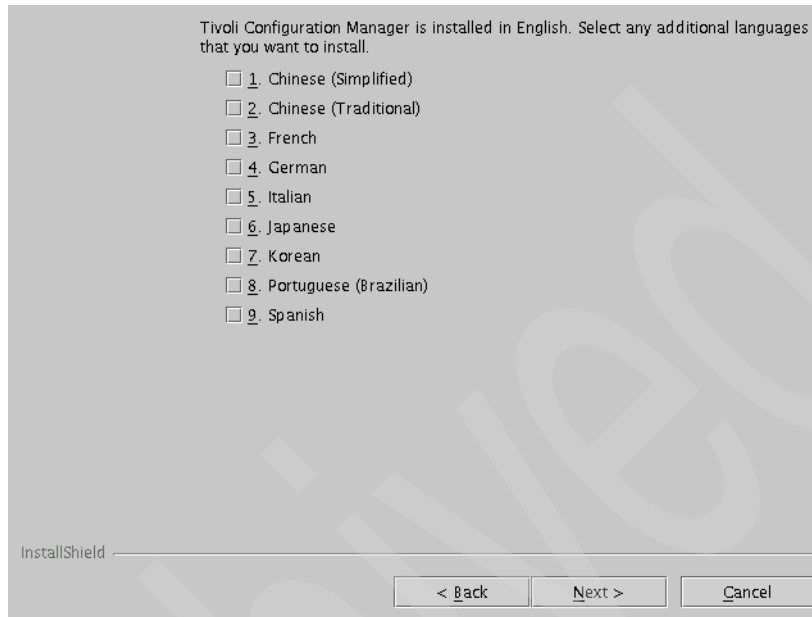


Figure 4-8 Language selection

8. Select **Default tablespaces and run schema scripts**, specify the Database vendor, and click **Next** to continue, as shown in Figure 4-9. If your DBA prefers to run the admin and schema scripts later, select **None** and configure the RDBMS after the server installation, following the steps described in 4.4, “Configuring DB2” on page 141.

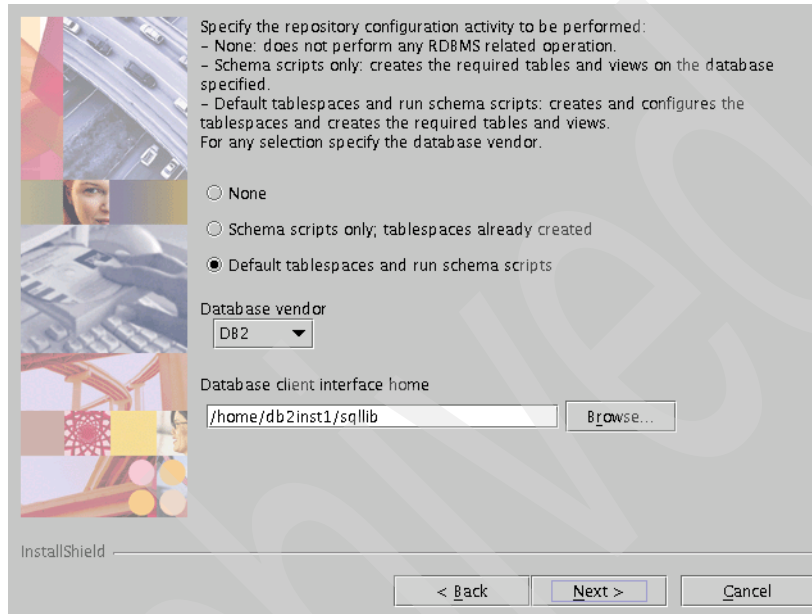


Figure 4-9 RDBMS configuration window

9. Change the Database name to `cm_db`, enter the information related to your RDBMS configuration, and click **Next** to continue, as shown in Figure 4-10.

Specify the RDBMS and RIM information for Distribution Status Console.

RIM name	Database name
mdist2	cm_db
RIM user name	RIM password
mdstatus	*****
Database path	Server ID
/home/db2inst1/sqllib	tcpip
Database vendor	DB2 instance name
DB2	db2inst1
Database administrator name	Database administrator password
db2inst1	*****

InstallShield

< Back Next > Cancel

Figure 4-10 MDist 2 RIM configuration

10. Type a password for the user tivapm in Figure 4-11 and click **Next** to continue. This user will be created in the operating system.

To use the Activity Planner component, you must have a user name and password. The user name is associated with a Tivoli administrator, so that proper authentication occurs when using Activity Planner. If you specify an existing user name for your operating system, make sure that you specify the appropriate password for that user name.

User name
tivapm

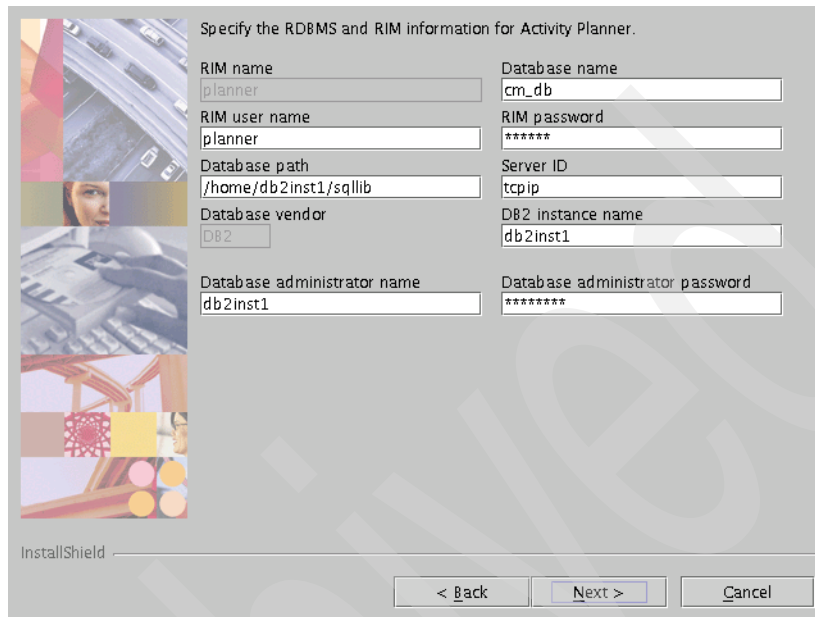
Password

InstallShield

< Back Next > Cancel

Figure 4-11 Creating the user tivapm

11. Change the Database name to `cm_db`, enter the information related to your RDBMS configuration, and click **Next** to continue, as shown in Figure 4-12.



Specify the RDBMS and RIM information for Activity Planner.

RIM name	planner	Database name	cm_db
RIM user name	planner	RIM password	*****
Database path	/home/db2inst1/sqllib	Server ID	tcpip
Database vendor	DB2	DB2 instance name	db2inst1
Database administrator name	db2inst1	Database administrator password	*****

InstallShield

< Back Next > Cancel

Figure 4-12 Activity Planner RIM configuration

12. Change the Database name to `cm_db`, enter the information related to your RDBMS configuration, and click **Next** to continue, as shown in Figure 4-13.

Specify the RDBMS and RIM information for Change Manager.

RIM name	Database name
ccm	cm_db
RIM user name	RIM password
tivoli	*****
Database path	Server ID
/home/db2inst1/sqllib	tcpip
Database vendor	DB2 instance name
DB2	db2inst1
Database administrator name	Database administrator password
db2inst1	*****

InstallShield

< Back Next > Cancel

Figure 4-13 Change Manager RIM configuration

13. Change the Database name to `cm_db`, enter the information related to your RDBMS configuration, and click **Next** to continue, as shown in Figure 4-14.

Specify the RDBMS and RIM information for Inventory.

RIM name	Database name
inv_query	cm_db
RIM user name	RIM password
invtiv	*****
Database path	Server ID
/home/db2inst1/sqllib	tcpip
Database vendor	DB2 instance name
DB2	db2inst1
Database administrator name	Database administrator password
db2inst1	*****

InstallShield

< Back Next > Cancel

Figure 4-14 Inventory RIM configuration

14. Change the Database name to `cm_db`, enter the information related to your RDBMS configuration, and click **Next** to continue, as shown in Figure 4-15.

Specify the RDBMS and RIM information for Pristine Manager.

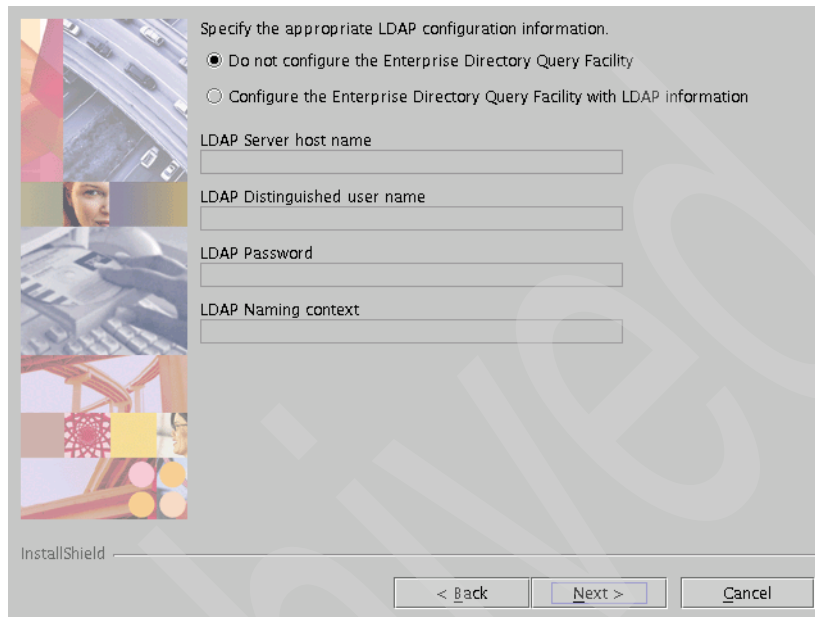
RIM name	Database name
pristine	cm_db
RIM user name	RIM password
pristine	*****
Database path	Server ID
/home/db2inst1/sqllib	tcpip
Database vendor	DB2 instance name
DB2	db2inst1
Database administrator name	Database administrator password
db2inst1	*****

InstallShield

< Back Next > Cancel

Figure 4-15 Pristine RIM configuration

15. Select **Do not configure the Enterprise Directory Query Facility**, as shown in Figure 4-16, and click **Next** to continue. We will do the LDAP configuration in Chapter 6, “Enterprise Directory integration” on page 275.



Specify the appropriate LDAP configuration information.

Do not configure the Enterprise Directory Query Facility

Configure the Enterprise Directory Query Facility with LDAP information

LDAP Server host name

LDAP Distinguished user name

LDAP Password

LDAP Naming context

InstallShield

< Back Next > Cancel

Figure 4-16 LDAP configuration

16. The window in Figure 4-17 shows the list of the operations that will be executed. Review the operations and click **Next** to continue.

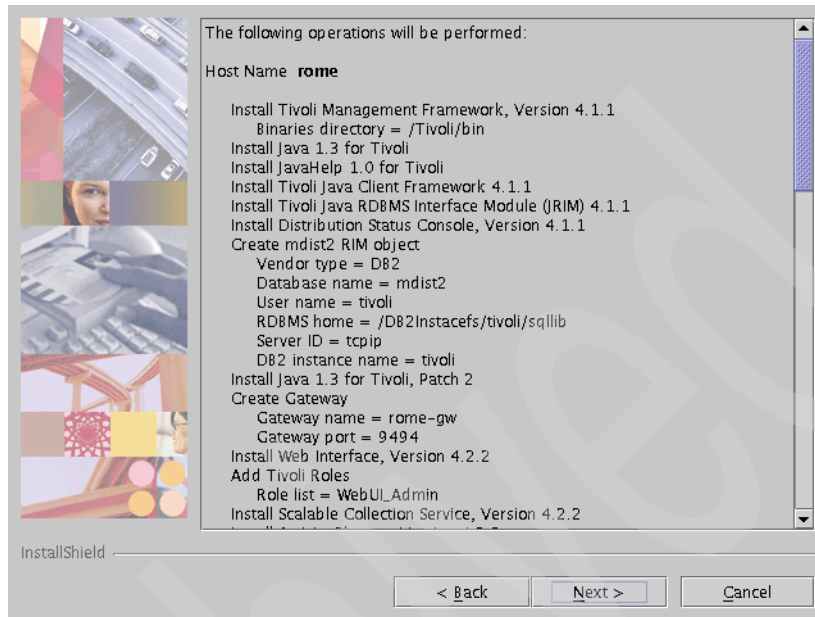


Figure 4-17 Operations summary

17. Select **Query when needed**, as shown in Figure 4-18, and click **Next** to continue.

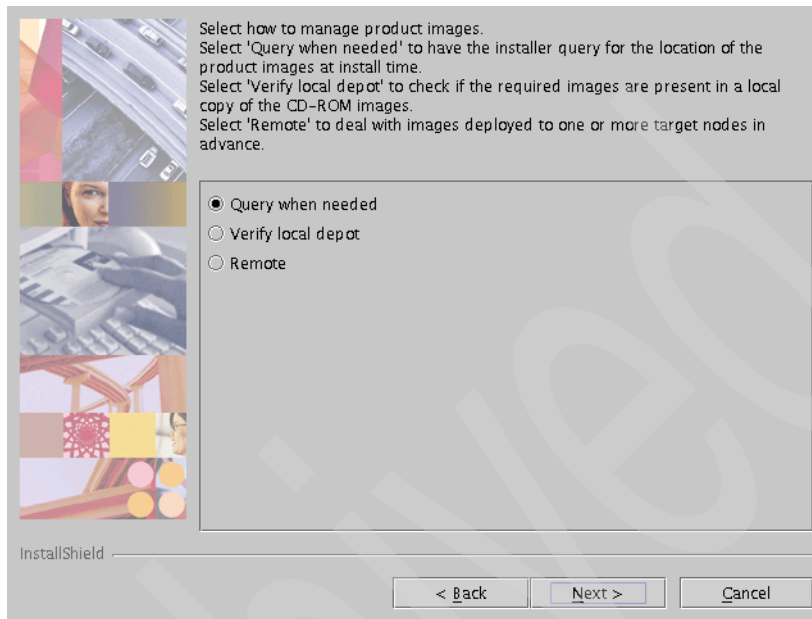


Figure 4-18 Managing source images

18. Figure 4-19 shows all the actions that will be performed by the installation process. In this window, you can click the **Run Next** button or the **Run All** button to start the installation. If you choose **Run Next**, you will have to select it again after each process has been completed.



Figure 4-19 Installation schedule

19. The window in Figure 4-20 can be shown any time a source image is not found. You need to enter the location of the installation image. Click **Open** after entering the correct location.

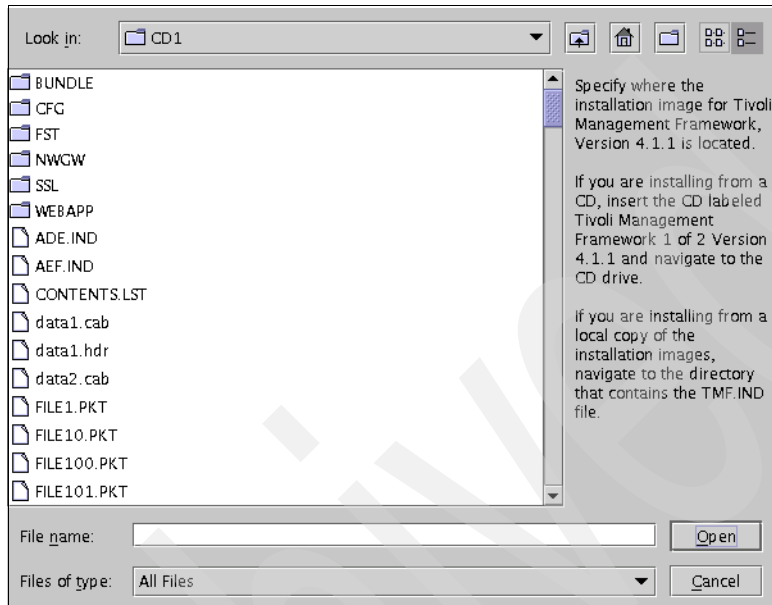


Figure 4-20 Searching for the source image

20. Figure 4-21 shows the Tivoli desktop that is launched just after the Framework installation. You can close this window.

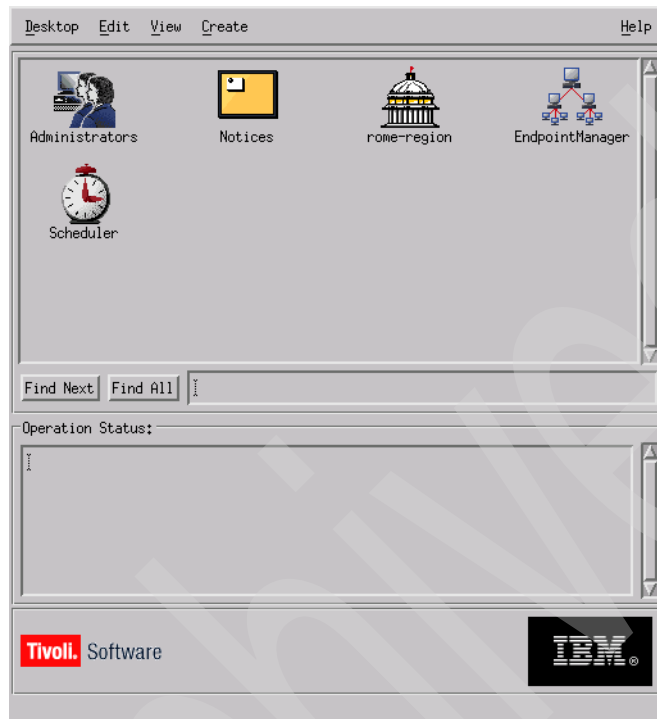


Figure 4-21 Framework desktop

21. Figure 4-22 shows that there are no more actions in Ready status and several actions in Held status. This can happen because sometimes an action is dependent on another that has not finished.

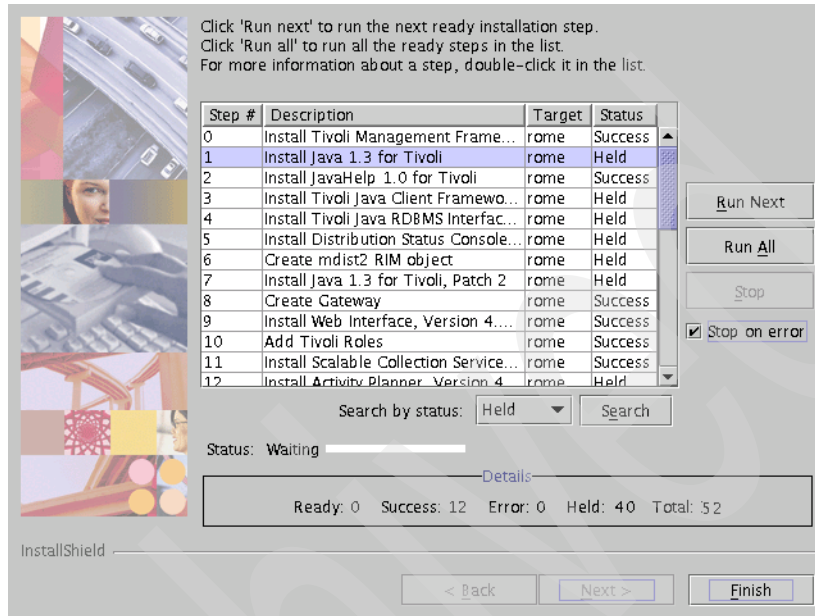


Figure 4-22 Installation in Held status

22. To solve this problem, change the action status from Held to Ready by right-clicking and selecting **Set Status** → **Ready**, as shown in Figure 4-23.

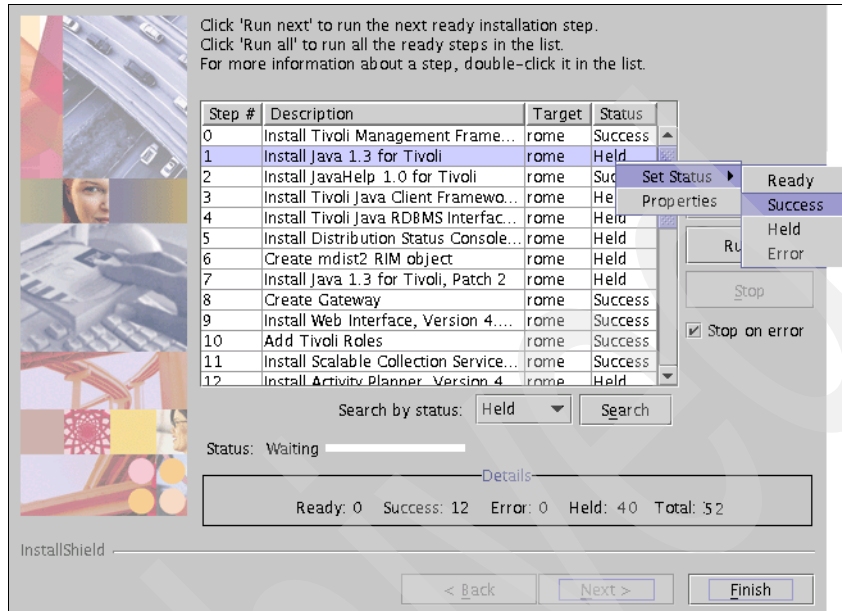


Figure 4-23 Changing the Held status to Ready

23. Figure 4-24 shows that there are no actions in Ready or Held status and all actions finished successfully. Click **Finish** to terminate the installation.

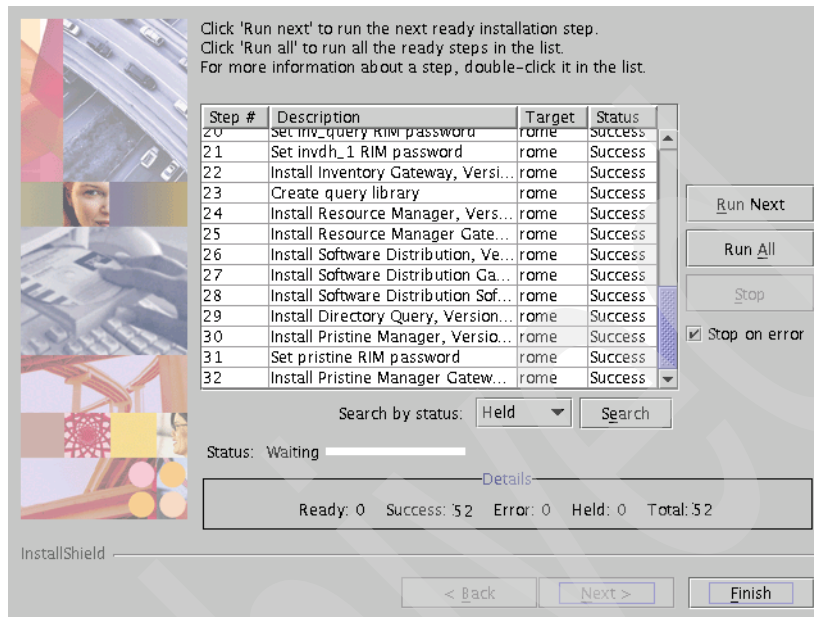


Figure 4-24 Finishing installation

The server installation finished successfully, and IBM Tivoli Configuration Manager V4.2.2 is installed. Now, follow the process described in 4.4, “Configuring DB2” on page 141 if you chose **Do not run the scripts** in the server installation. Otherwise, skip this section and proceed with 4.5, “Setting the Tivoli environment variables” on page 147.

4.4 Configuring DB2

To store data in an RDBMS, you need to run admin and schema scripts in order to create a repository. The applications communicate with the RDBMS through RIM objects, which were created in the steps in “Server installation” on page 119.

You can create only one database, called `cm_db`, to store all the tables, or you can create one database for each component from Tivoli Configuration Manager. Ask your DBA about the best way to do this in your environment. For the environment described in this book, we create one database.

4.4.1 Running the admin scripts

Running an admin script creates the container for all the logical objects (users, views, and so forth) and creates the tablespace that stores all the physical data in the tables. After the Tivoli Configuration Manager V4.2.2 installation, you can find the admin scripts in the following directories:

- ▶ Configuration Manager (integrated database called cm_db):
\$BINDIR/./generic/inv/SCRIPTS/RDBMS/cm_db2_admin.sql
- ▶ Distribution Status:
\$BINDIR/TME/MDIST2/sql/mdist_db2_admin.sql
- ▶ Inventory:
\$BINDIR/./generic/inv/SCRIPTS/RDBMS/inv_db2_admin.sql
- ▶ Activity Planner:
\$BINDIR/TME/APM/SCRIPTS/plans_db2_admin.sql
- ▶ Change Manager:
\$BINDIR/TME/CCM/SCRIPTS/ccm_db2_admin.sql
- ▶ Pristine:
\$BINDIR/TME/PM/SCRIPTS/pristine_db2_admin.sql

According to the Table 4-2, you need about 1.8 GB of free space in the instance file system, and in our case, the /DB2Instancefs, created in 5.1, “IBM DB2 UDB installation” on page 218.

Table 4-2 *Tablespaces created by running the admin scripts*

Component	Database name	Tablespace name	Tablespace size	Log file size
Inventory	inv_db	inv_ts	1024 MB	128 MB
Change Manager	ccm_db	ccm_ts	128 MB	16 MB
Activity Planner	planner	planner_ts	128 MB	16 MB
Distribution Status	mdist2	mdist2_ts	128 MB	16 MB
Pristine	pristine	pristine_ts	128 MB	16 MB

The DB2 admin scripts do not create the default databases, so you must create and catalog them first. You can find this explanation at the beginning of each admin script. Example 4-5 on page 143 shows the creation of the database cm_db.

Example 4-5 Creating the default databases

```
AIX Version 5
(C) Copyrights by IBM and by others 1982, 2004.
login: db2inst1
tivoli's Password: *****
*****
*
*
* Welcome to AIX Version 5.3!
*
*
* Please see the README file in /usr/lpp/bos for information pertinent to
* this release of the AIX Operating System.
*
*
*****
$ db2 create database cm_db
DB20000I The CREATE DATABASE command completed successfully.
```

The next step is to run the admin scripts. To do that, copy the admin script related to the database created in Example 4-5 to the home directory of the instance owner user and perform the following steps:

1. Log in to AIX 5L where the DB2 client is installed as the instance owner user.
2. Start DB2 by running the **db2start** command.
3. Run the following for each database:
 - a. Connect to the database:

```
db2 connect to db_name user remote_instance_owner using
remote_instance_pwd
```
 - b. Run the script according to the database connected:

```
db2 -f script_name -o -t -z script_name.log
```

Important: If you are creating one database for each product, you must change the admin script `ccm_db2_admin.sql`, because there is an error on it. If you create the database `ccm_db`, you must change the database name `ccm` to `ccm_db` in the update lines `ccm_db`:

Original file:

```
-- logfile size in in 4k pages. Total log space is 16MB
UPDATE DATABASE CONFIGURATION FOR ccm USING LOGFILSIZ      1024;
UPDATE DATABASE CONFIGURATION FOR ccm USING LOGPRIMARY     2;
UPDATE DATABASE CONFIGURATION FOR ccm USING LOGSECOND      2;
```

Changed file:

```
-- logfile size in in 4k pages. Total log space is 16MB
UPDATE DATABASE CONFIGURATION FOR ccm_db USING LOGFILSIZ  1024;
UPDATE DATABASE CONFIGURATION FOR ccm_db USING LOGPRIMARY  2;
UPDATE DATABASE CONFIGURATION FOR ccm_db USING LOGSECOND   2;
```

This change is not necessary if you selected the integrated database named `cm_db`.

After running the admin scripts, you can run the schema scripts, as described in the following section.

4.4.2 Running the schema scripts

The schema scripts create the tables and views in the allocated tablespaces and they must be run with the user owner of the database and not with the user owner of the instance. The following list describes the schema scripts:

mdist_db2_schema.sql

Installs the mdist2 repository schema and defines its tables and views.

inv_db2_schema.sql

Installs the inv_db repository schema and defines its tables and views.

h_inv_db2_schema.sql

Installs the Inventory history tables for the configuration repository. (This script is optional. It must be run after the `inv_db2_schema.sql` script.)

plans_db2_schema.sql

Installs the planner repository schema and defines its tables and views.

pristine_db2_schema.sql

Installs the pristine repository schema and defines its tables and views.

ccm_db2_schema.sql

Installs the ccm repository schema and defines its tables and views.

The preceding schema scripts are in the following directories after the Tivoli Configuration Manager V4.2.2 installation:

- ▶ Distribution Status:
\$BINDIR/TME/MDIST2/sql/mdist_db2_schema.sql
- ▶ Inventory:
\$BINDIR/./generic/inv/SCRIPTS/RDBMS/inv_db2_schema.sql
\$BINDIR/./generic/inv/SCRIPTS/RDBMS/h_inv_db2_schema.sql
- ▶ Activity Planner:
\$BINDIR/TME/APM/SCRIPTS/plans_db2_schema.sql
- ▶ Change Manager:
\$BINDIR/TME/CCM/SCRIPTS/ccm_db2_schema.sql
- ▶ Pristine:
\$BINDIR/TME/PM/SCRIPTS/pristine_db2_schema.sql

The next step is to run the schema scripts. To do that, copy all the schema scripts to the home directory of the database owner user and perform the following steps.

Note: The database owner user must be the same one used in 4.3.2, “The server installation” on page 114 when the RIM objects were defined.

1. Log in to AIX 5L where the DB2 client is installed as the instance owner user.
2. Start DB2 by running the **db2start** command, if necessary.
3. Run the following for each database:
 - a. Connect to the database with the owner of the database and not with the owner of the instance:

```
db2 connect to <db_name> user remote_db_user using remote_db_password
```
 - b. Run the script according to the database connected:

```
db2 -f <script_name> -o -t -z <script_name>.log
```

At this point, your RDBMS is configured.

4.4.3 Running the plug-ins

Depending on your installation approach and your RDBMS of choice, you might need to manually run the plug-ins scripts for the Activity Planner and Change Manager products. In our case, it was not needed, because we decided to configure the RDBMS in “Server installation” on page 119. If you chose not to configure the RDBMS during the server installation, you must run the plug-ins scripts manually.

Ensure that the RIM objects are working before running the scripts. To do that, you must run the **wrimtest** command for each RIM object, as shown in Example 4-6.

Example 4-6 Running the wrimtest command

```
Checking the available RIM objects.
# wlookup -ar RIM
ccm 1254603259.1.634#RIM::RDBMS_Interface#
inv_query1254603259.1.702#RIM::RDBMS_Interface#
invdh_11254603259.1.703#RIM::RDBMS_Interface#
mdist2 1254603259.1.590#RIM::RDBMS_Interface#
planner1254603259.1.610#RIM::RDBMS_Interface#
pristine1254603259.1.979#RIM::RDBMS_Interface#

Testing the RIM communication with the RDBMS.
# wrimtest -l ccm
Resource Type : RIM
Resource Label : ccm
Host Name : rome
User Name : tivoli
Vendor : DB2
Database : cm_db
Database Home : /home/db2inst1/sqllib
Server ID : tcpip
Instance Home : ~db2inst1
Instance Name : db2inst1
Opening Regular Session...Session Opened
RIM : Enter Option >x <==== Type x to leave the RIM connection.
Releasing session
```

Follow Example 4-7 on page 147 to register the Activity Planner and Change Manager plug-ins.

Example 4-7 How to run the plug-in scripts

```
Activity Planner plug-ins:
# $BINDIR/TME/APM/SCRIPTS/reg_tl_plugin.sh
Registering plug-in for TaskLibrary...

AMN0140I The plug-in has been successfully registered.
# $BINDIR/TME/APM/SCRIPTS/reg_swd_plugin.sh
Registering plug-in for Software Distribution...

AMN0140I The plug-in has been successfully registered.
# $BINDIR/TME/APM/SCRIPTS/reg_inv_plugin.sh
Registering APM plug-in for Inventory

AMN0140I The plug-in has been successfully registered.
# $BINDIR/TME/APM/SCRIPTS/reg_pristine_apm_plugin.sh
Registering plug-in for Pristine Manager...

AMN0140I The plug-in has been successfully registered.

Change Manager plug-ins:
# $BINDIR/TME/CCM/SCRIPTS/reg_invscan_plugin.sh
Registering CCM plug-in for Inventory
CC00035I  wccmplugin: operation successfully performed.

# $BINDIR/TME/CCM/SCRIPTS/reg_swd_plugin.sh
Registering plug-in for Software Distribution
CC00035I  wccmplugin: operation successfully performed.
Registering plug-in for Inventory (Hardware conditions)
CC00035I  wccmplugin: operation successfully performed.

# $BINDIR/TME/CCM/SCRIPTS/reg_pristine_ccm_plugin.sh
Registering CCM plug-in for Pristine Manager...
CC00035I  wccmplugin: operation successfully performed.
```

You can also run the commands `wapmplugin -l` and `wccmplugin -l` to check the registered plug-ins.

Now, your server environment is ready to work.

4.5 Setting the Tivoli environment variables

After the installation, you need to set some Tivoli environment variables to run Tivoli commands. On AIX 5L, you need to set these variables after each login. To make this easier, you can change your `$HOME/.profile` file, including the lines described in Example 4-8 on page 148.

Example 4-8 Setting Tivoli environment variables

```
# The following three lines will set the Tivoli environment.
if [ -f /etc/Tivoli/setup_env.sh ]; then
    . /etc/Tivoli/setup_env.sh
fi
```

After changing your \$HOME/.profile, you must log off and log in to AIX 5L to make the changes take effect.

Note: If you cannot change your \$HOME/.profile file, you need to run the following command after each login to be able to execute Tivoli commands:

```
. /etc/setup_env.sh
```

4.6 Desktop installation

The desktop installation program installs Tivoli Desktop for Windows and the IBM Tivoli Configuration Manager administrative interfaces. According to our environment, shown in Figure 4-1 on page 113, the Tivoli Desktop for Windows will be installed on the Windows machine called kwc12z.

The desktop installation is needed only if you want to access the Tivoli through a Windows machine or if your Tivoli region is installed on Windows. The Tivoli desktop is automatically installed on UNIX installations.

The images used by this installation program are on IBM Tivoli Configuration Manager Desktop, Version 4.2.2.

4.6.1 Desktop installation procedures

Before starting the installation process, make sure that you have at least 120 MB of free disk space.

Perform the following steps to install the Tivoli desktop on your Windows machine:

1. Insert the *IBM Tivoli Configuration Manager Desktop, Version 4.2.2* CD in the CD-ROM and run the **setup.exe** command.

2. Select the language and click **OK** to continue, as shown in Figure 4-25.

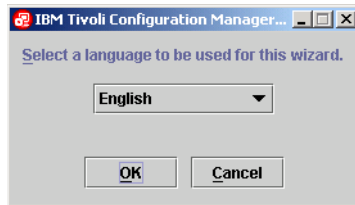


Figure 4-25 Selecting the language

3. Click **Next** in the Welcome page, as shown in Figure 4-26.

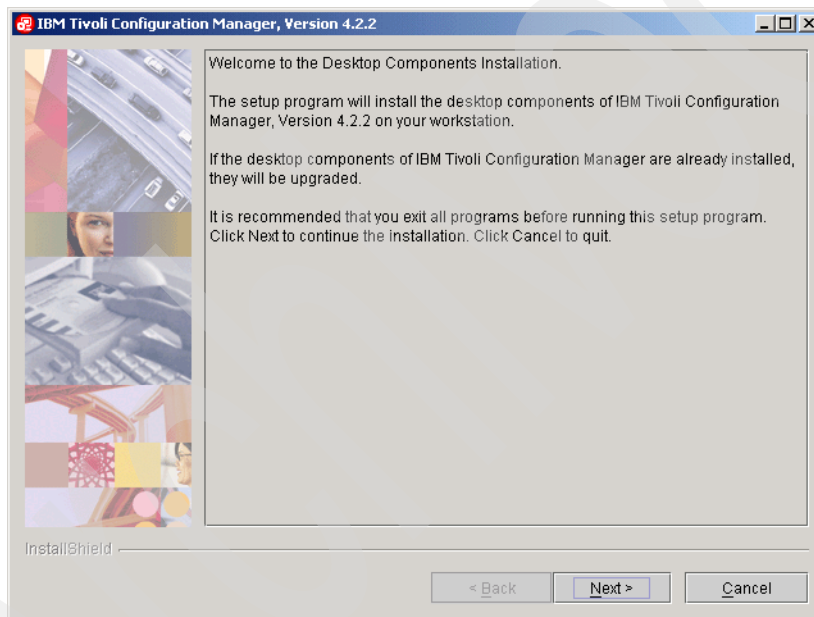


Figure 4-26 Welcome page

4. In Figure 4-27, read and accept the license agreement and click **Next** to continue.

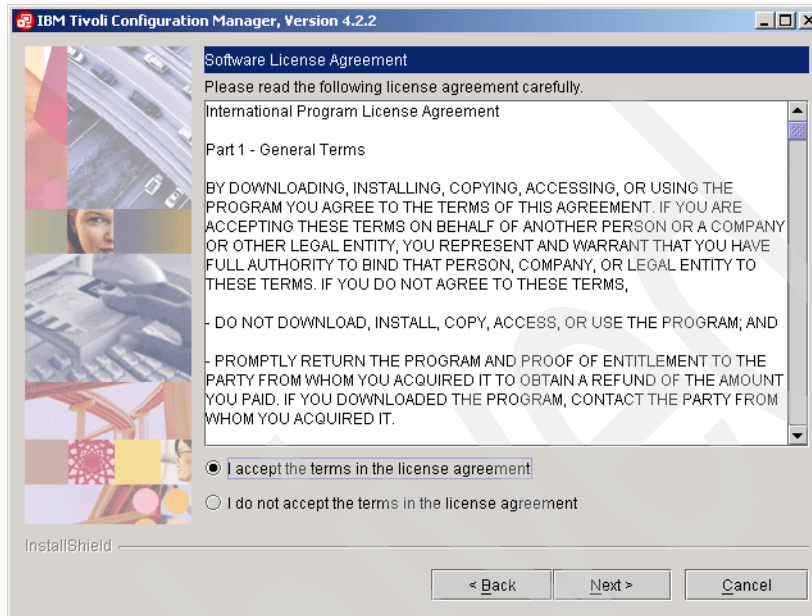


Figure 4-27 License Agreement

5. Change the Destination directory, if necessary, as shown in Figure 4-28, and click **Next** to continue.

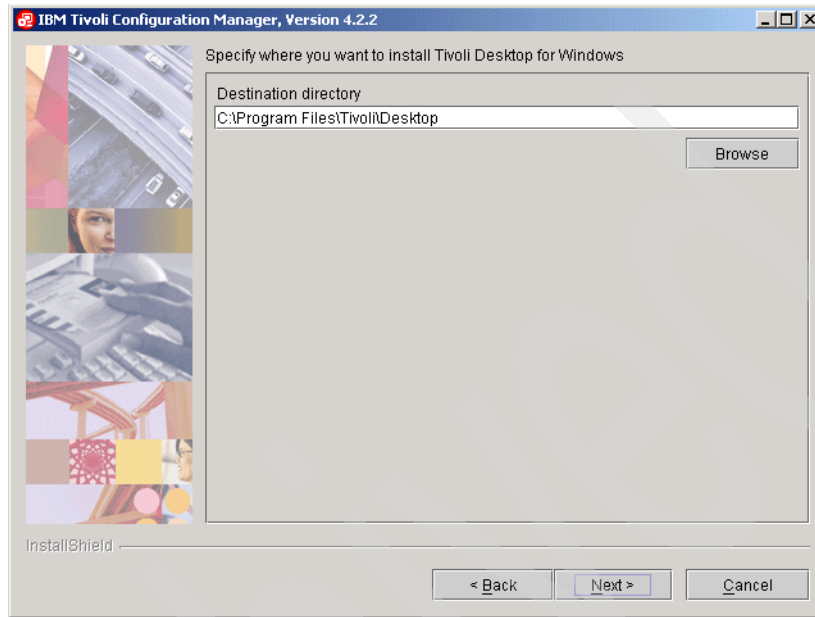


Figure 4-28 Destination directory

- The window shown in Figure 4-29 opens only if you have an endpoint installed on your machine. In our installation, we select **Yes** to install the Software Package Editor. Click **Next** to continue.

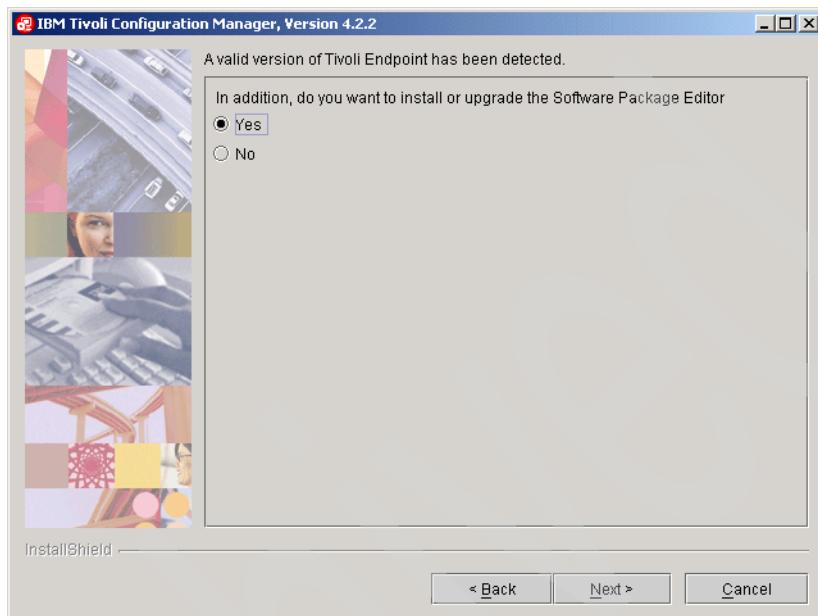


Figure 4-29 Software Package Editor installation

7. Figure 4-30 shows the actions that will be done. Click **Next** to continue.

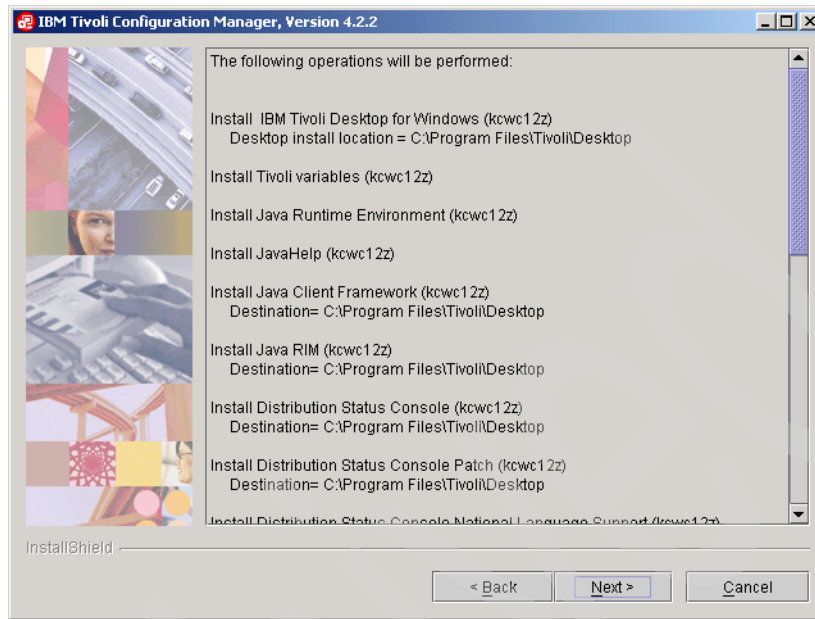


Figure 4-30 Install operations

8. Figure 4-31 shows the installation summary. Click **Finish** to terminate the installation.

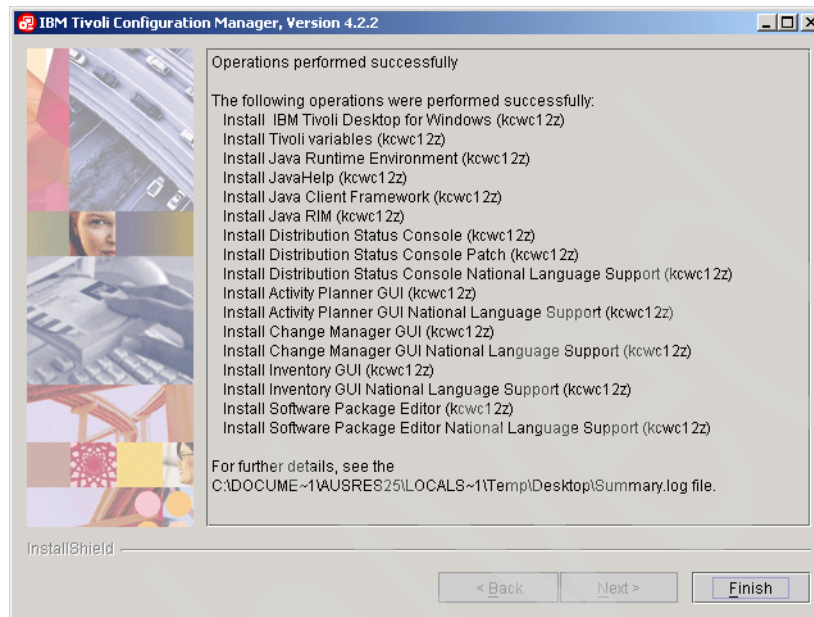


Figure 4-31 Desktop installation summary

Now, your Tivoli Desktop for Windows is installed.

4.6.2 Opening the Tivoli desktop

The Tivoli desktop can be opened either on a UNIX or Windows environment and it can only be opened by a Tivoli administrator user with at least the user role. If you did not create any Tivoli administrator users, you can use the root user to open it.

Opening the Tivoli desktop in a UNIX environment

Perform the following steps to launch the Tivoli desktop in a UNIX environment:

1. Log in to the UNIX environment as a Tivoli administrator.
2. Open an xterm window and set the Tivoli environment variables, as described in 4.5, “Setting the Tivoli environment variables” on page 147.
3. Type `tivo1i` at the command line. The Tivoli Desktop window opens, as shown in Figure 4-32 on page 155.



Figure 4-32 Tivoli Desktop window

Opening the Tivoli desktop in a Windows environment

Perform the following steps to launch the Tivoli desktop in a Windows environment:

1. Go to the Windows Start menu and select **Programs** → **Tivoli** → **Tivoli Desktop**.
2. The window shown in Figure 4-33 opens. Enter the host name of the Tivoli server, user name, and password, and click **OK** to continue.

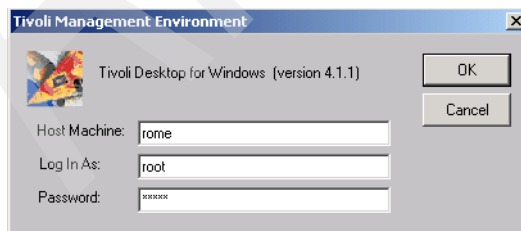


Figure 4-33 Tivoli Desktop password window

The Tivoli desktop opens, as shown in Figure 4-34 on page 156.

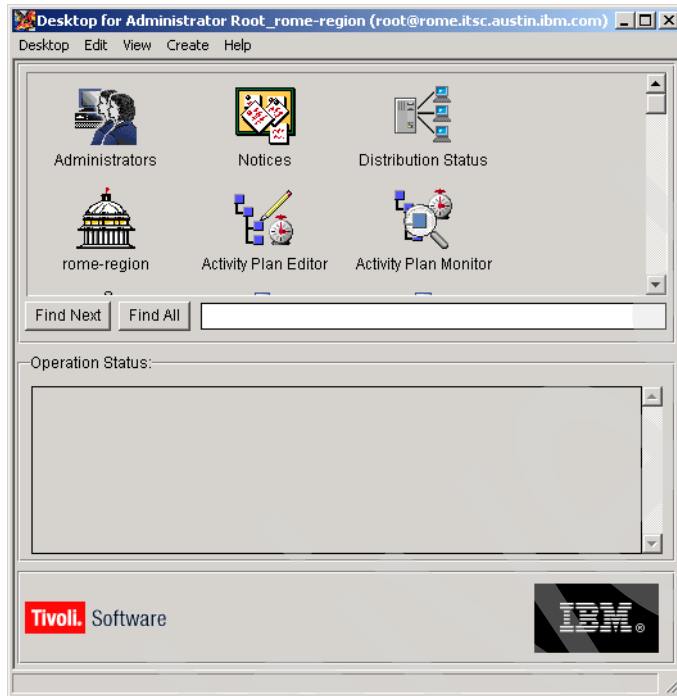


Figure 4-34 Tivoli Desktop window

4.7 Installing a gateway on Linux

In this section, we describe how to create a new managed node and how to promote it as a gateway. According to our scenario, shown in Figure 4-1 on page 113, the managed node will be installed on a Linux machine called edinburgh.

4.7.1 Linux considerations

Before installing a managed node on Linux, you must test the rexec communication between the Tivoli server and the Linux machine that will be a managed node. Perform the following steps if the Tivoli server is not able to communicate to the Linux machine through rexec. Skip to 4.7.2, “Installing a Linux managed node” on page 158 if your environment is already configured for rexec.

1. Start the rexec service on Linux running the **setup** command in a CLI window.

2. Select **System services** and select **Run Tool**, as shown in Figure 4-35.



Figure 4-35 Linux setup windows

3. Press the Spacebar over the rexec line to mark it with an * and select **OK**, as shown in Figure 4-36.



Figure 4-36 Service selection menu

4. Select **Quit** to leave the setup menu, as shown in Figure 4-37 on page 158.



Figure 4-37 Leaving the setup menu

- Restart the xinetd service:
`service xinetd restart`
- Edit the `/etc/securityty` file and include the lines described in Example 4-9 at the end of the file. These lines will permit five root logons through Telnet and will also permit root access through rexec.

Example 4-9 Lines that must be included at the end of the `/etc/securityty` file

```
pts/0
pts/1
pts/2
pts/3
pts/4
rexec
```

- Make sure that the forward and reverse DNS resolution are working for both machines. If not, include the IP address and the host name of the two machines in both `/etc/hosts` files.

After these steps, test the `rexec` command again and start the managed node installation.

4.7.2 Installing a Linux managed node

Perform the following steps to install a managed node on Linux:

- Log in to the Tivoli server as root.
- Mount the CD called *Tivoli Management Framework 4.1.1 CD 1/1* on the Tivoli server.

3. Open an xterm window and type **tivoli**. The Tivoli desktop opens, as shown in Figure 4-38.

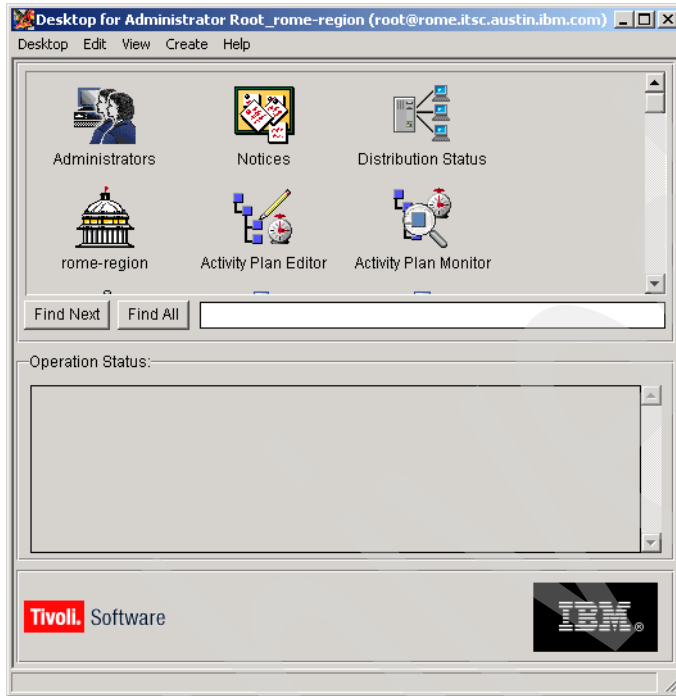


Figure 4-38 Desktop window

4. Right-click in the main region (**rome-region**) and select **Open**, as shown in Figure 4-39.

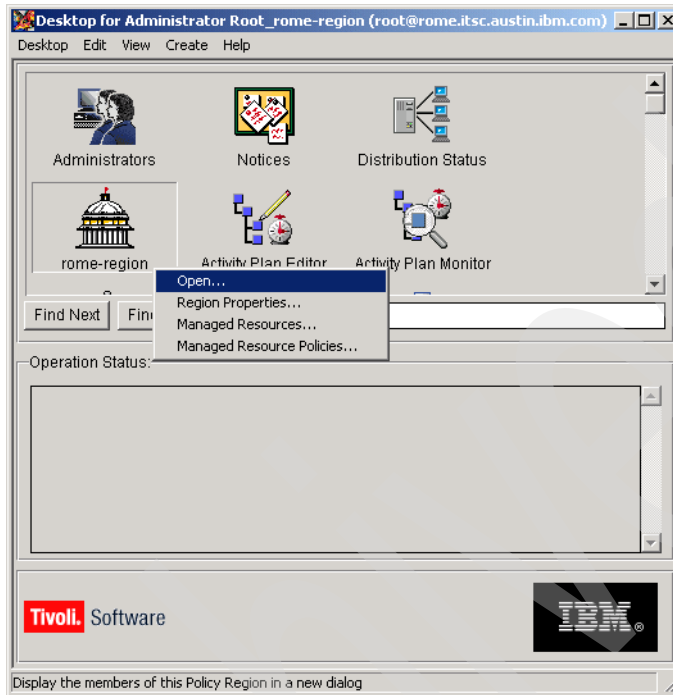


Figure 4-39 Opening the main region to create a managed node

5. Select **Create** and select **ManagedNode**, as shown in Figure 4-40.

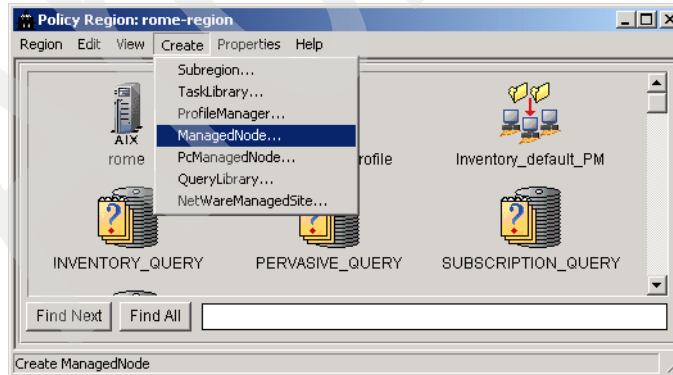


Figure 4-40 Creating a managed node

- The client install window opens, as shown in Figure 4-41. Select **Account**, enter the root user and its password, and click **Add Clients**.

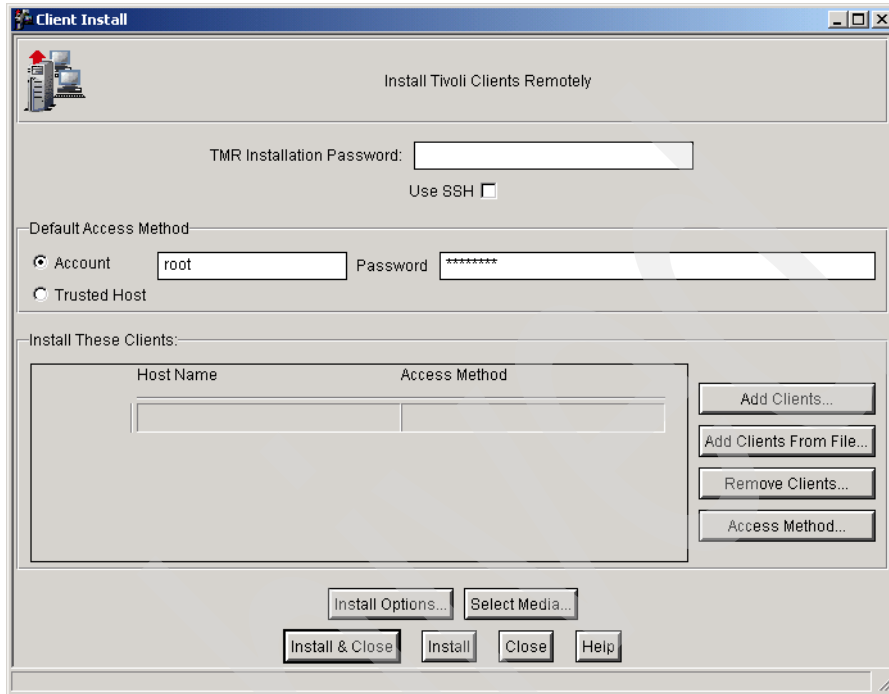


Figure 4-41 Main installation window: Add Clients

- In Figure 4-42, enter the host name in the Add Client field and click **Add & Close** to return to the previous window.

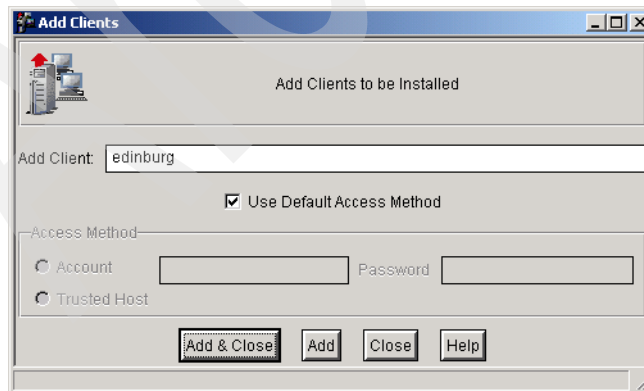


Figure 4-42 Adding a client

8. In the main installation window, click **Select Media**, as shown in Figure 4-43.

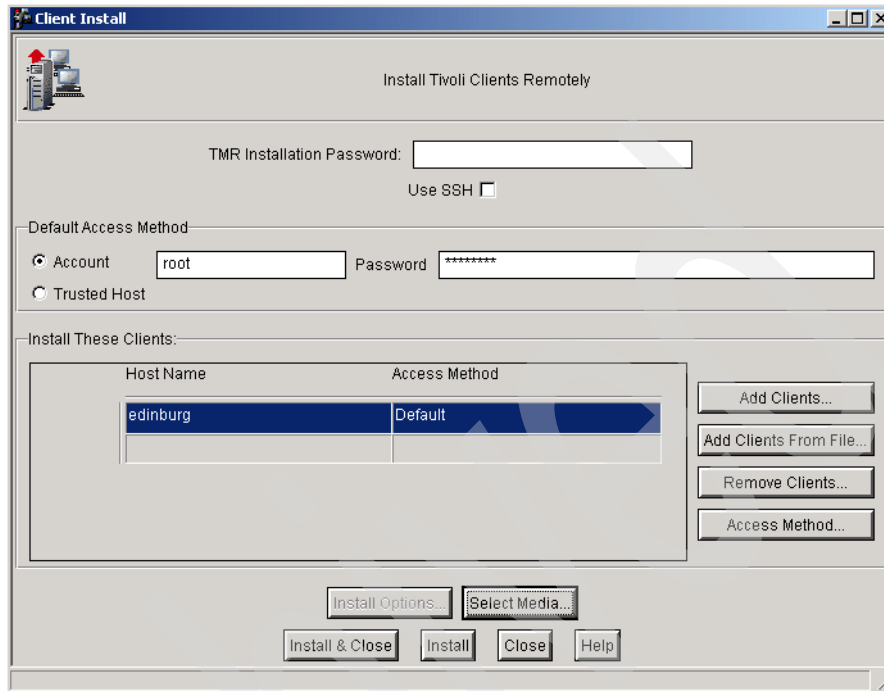


Figure 4-43 Main installation window: Select Media

9. In Figure 4-44, select the path where the *Tivoli Management Framework 4.1.1 CD 1/1* is located and click **Set Media & Close**.

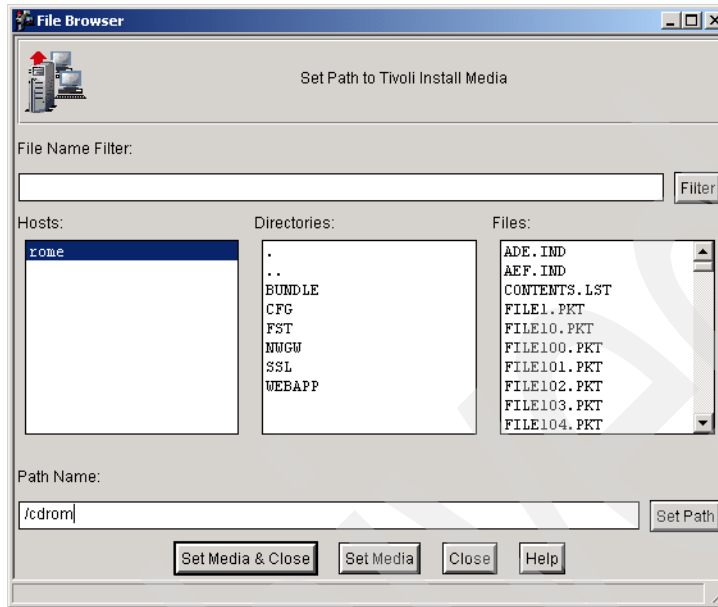


Figure 4-44 Selecting the media

10. In the window shown in Figure 4-45, change the install directories if necessary and click **Set**. You return to the main installation window.

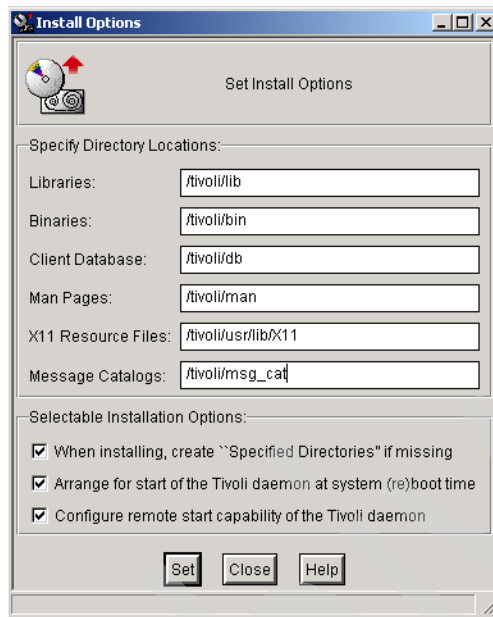


Figure 4-45 Installation options

11. In Figure 4-46, click **Install & Close** to continue the installation process.

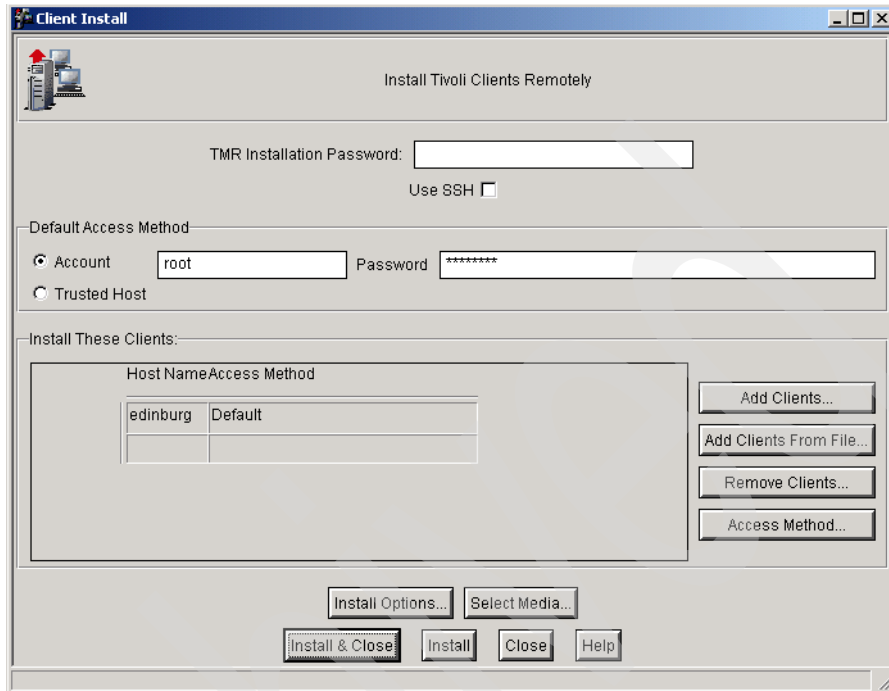


Figure 4-46 Main installation window: Install & Close

12. Click **Continue Install**, as shown in Figure 4-47, to start the installation.



Figure 4-47 Actions to be done for the installation process

13. Figure 4-48 on page 167 shows the status of the installation. Click **Close** to finish the process.

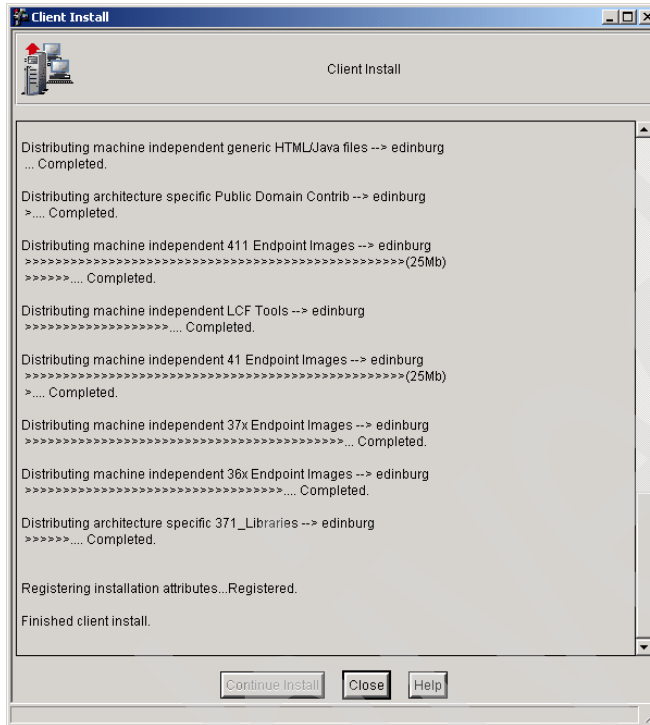


Figure 4-48 Installation summary

Now, the managed node is installed on the Linux machine. To check this, run the **odadmin odlist** command on the Tivoli server. This should show you two entries, one for the Tivoli region and one for the managed node, as shown in Example 4-10.

Example 4-10 The *odadmin odlist* output

```
# odadmin odlist
Region      Disp  Flags  Port   IPAddr  Hostname(s)
1254603259  1     ct-    94     9.3.5.54  rome.itsc.austin.ibm.com,rome
              3     ct-    94     9.3.5.50  edinburg.itsc.austin.ibm.com
```

Note: In the dispatcher number (Disp) column, 1 designates the Tivoli server, which is rome. The Dispatcher number for the newly created managed node is shown as 3. Also ct- in the Flags column (for both machines) shows that the node is connected to the Tivoli server.

In order for this new managed node to server as a gateway, we perform the steps described in the next section.

4.7.3 Defining the managed node as a gateway

You have two ways to define a managed node as a gateway: using the Tivoli desktop or using the command line interface.

Creating a gateway using the Tivoli desktop

To create a gateway using the Tivoli desktop, perform the following steps:

1. Open an xterm window on the Tivoli server and run the `tivoli` command to open the Tivoli desktop.
2. Right-click the **Endpoint Manager** icon and select **Create Gateway**, as shown in Figure 4-49.



Figure 4-49 Creating a gateway

3. In Figure 4-50 on page 169, enter the name of the managed node in the Managed Node Proxy field, and the name of the gateway in the Name field, and click **Create & Close**. By default, the name is `managednode_name-gw`, but you can provide any name.

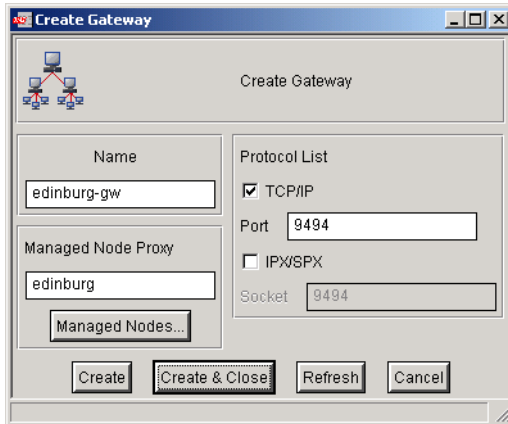


Figure 4-50 Defining the gateway

4. To verify that the gateway was properly created, right-click **Endpoint Manager** on the Tivoli desktop and select **View Gateways**, as shown in Figure 4-51.

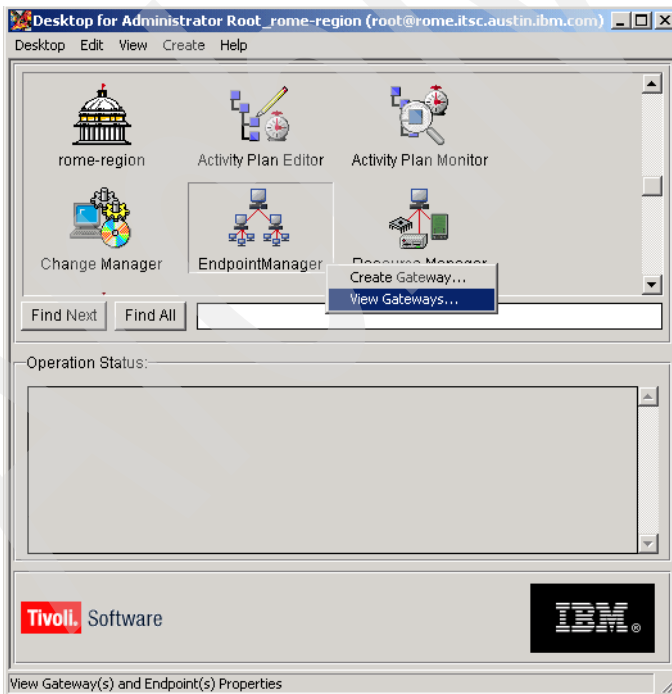


Figure 4-51 Viewing the created gateways

Figure 4-52 shows two gateways created in our environment. The rome-gw gateway was created by the server installation, and the edinburg-gw (Linux) gateway was created using the instructions described in this section.

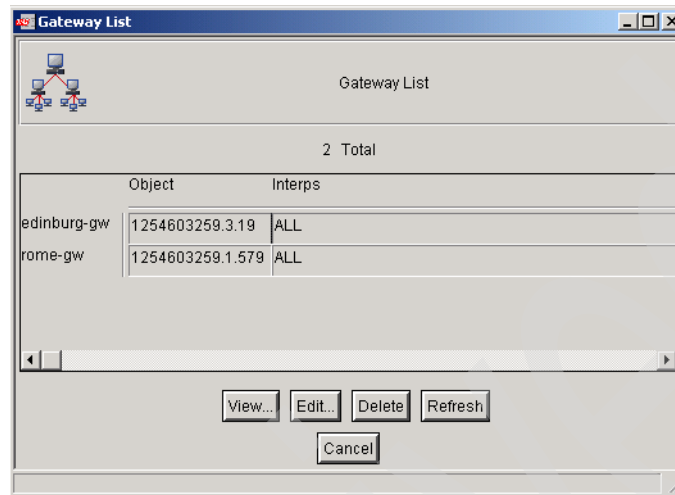


Figure 4-52 Displaying the gateways

After you have verified that the edinburg-gw gateway has been created, click **Cancel** to close the window.

Creating a gateway using the CLI

To create a gateway using the CLI, you must open an xterm window on the Tivoli server and run the following command:

```
wcrtgate -h edinburg -n edinburg-gw
```

Where:

- h** Is the name of the managed node where the gateway will be created.
- n** Is the name of the new gateway. By default, the name of the gateway is composed by the *managednode_name-gw*.

To check if the gateway was properly created, run the **wgateway** command, as shown in Example 4-11 on page 171. The **wgateway** command will show two gateways, rome-gw that was created during the server installation and edinburg-gw that was created by the **wcrtgate** command.

Example 4-11 The wgateway command

Object	Name	Status
1254603259.3.20	edinburg-gw	u
1254603259.1.579	rome-gw	u

As you can see, the gateway edinburg-gw was created successfully.

4.8 Installing Tivoli Configuration Manager components on new gateway

According to our scenario, we install the following products on the new gateway. All of these products are located on IBM Tivoli Configuration Manager Server, Version 4.2.2:

- ▶ Scalable Collection Service, Version 4.2.2 (as patch)
- ▶ Inventory Gateway, Version 4.2.2
- ▶ Pristine Manager Gateway, Version 4.2.2
- ▶ Resource Manager Gateway, Version 4.2.2
- ▶ Software Distribution Gateway, Version 4.2.2

4.8.1 Installing the Scalable Collection Service, Version 4.2.2

Perform the following steps to install the Scalable Collection Services:

1. Open the Tivoli desktop, as described in 4.6.2, “Opening the Tivoli desktop” on page 154.
2. Select **Desktop** → **Install** → **Install Patch**, as shown in Figure 4-53 on page 172.

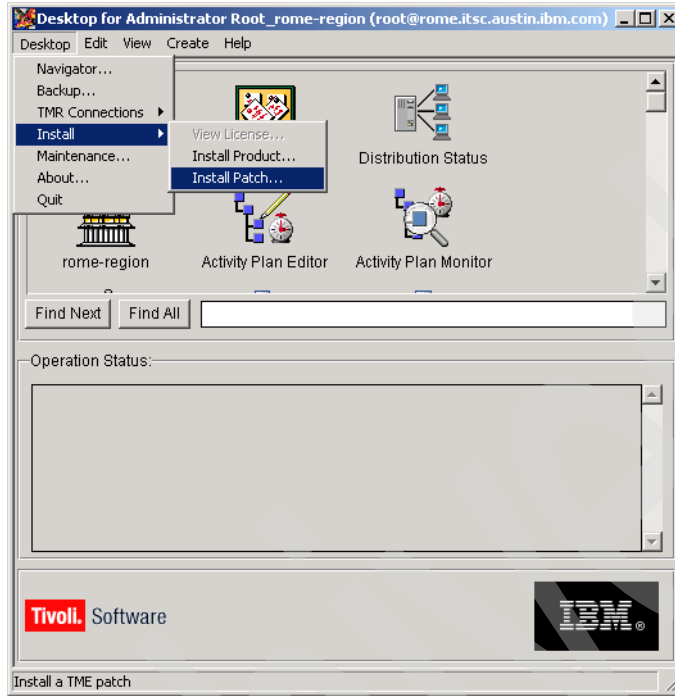


Figure 4-53 Install Patch

3. An error message might be displayed when the Tivoli desktop cannot find the file `patches.lst`, as shown in Figure 4-54. It is just a warning message and you can click **OK**.

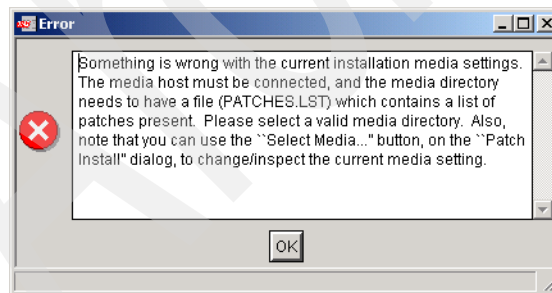


Figure 4-54 Patches.lst error

Note: If you do not get the error window in Figure 4-54 on page 172, this means that the Install Patch program has found the file patches.lst in the current installation media directory. But this might not be the correct the paths.lst for the Scalable Collection Services (that means that you will not see the Scalable Collection Services as one of the installable products in the Tivoli desktop).

In this case, click **Select Media** to set the correct path.

4. Click **Select Media** to set the path for the correct installation image, as shown in Figure 4-55.

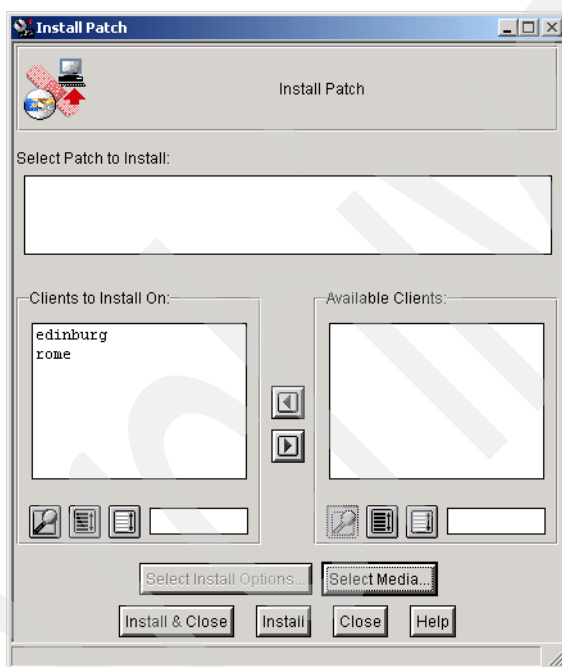


Figure 4-55 Install Patch window

5. In Figure 4-56, select the path where the mcollect directory is located and click **Set Media & Close**.

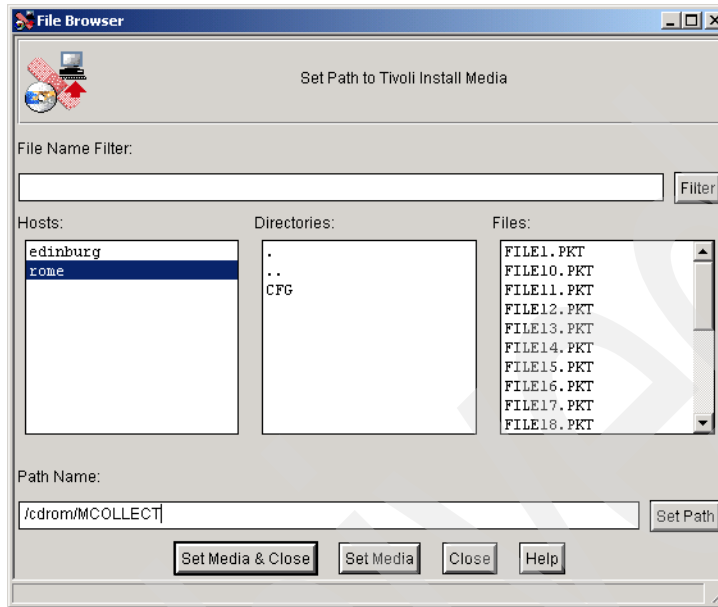


Figure 4-56 Selecting the mcollect directory

6. In Figure 4-57, select the patch to install and make sure that the gateway that will receive the installation is located in the Clients to Install On area. Click the **Install** or **Install & Close** button.

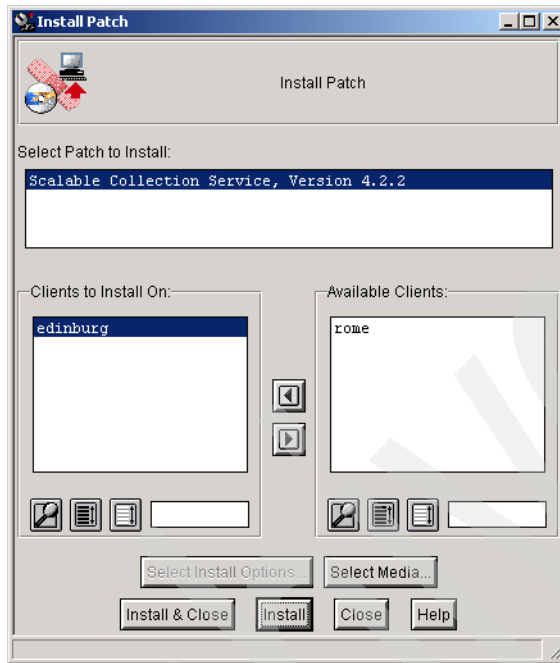


Figure 4-57 Selecting the patch and clients

7. Figure 4-58 shows the summary of actions for the patch installation. Click **Continue Install** to start the installation process.

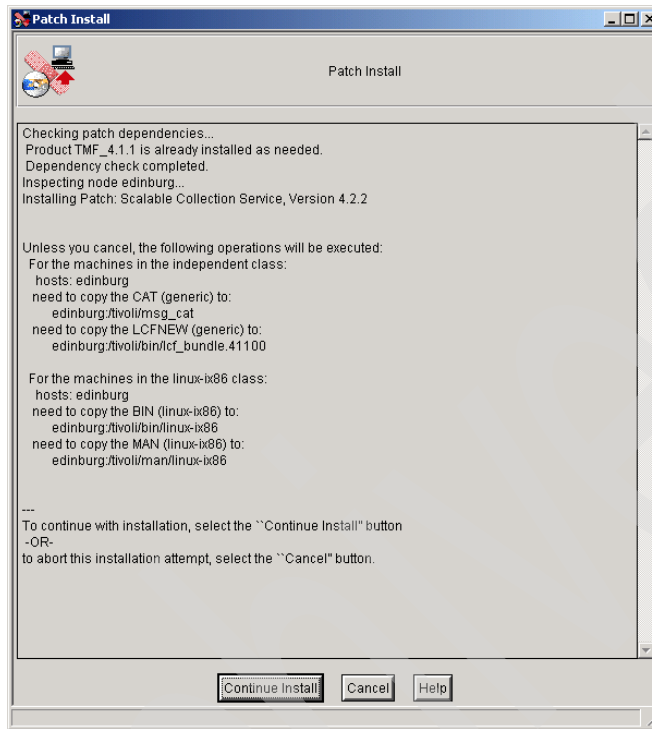


Figure 4-58 Install patch actions

8. Figure 4-59 shows the status of the installation process. Click **Close** to finish the installation.

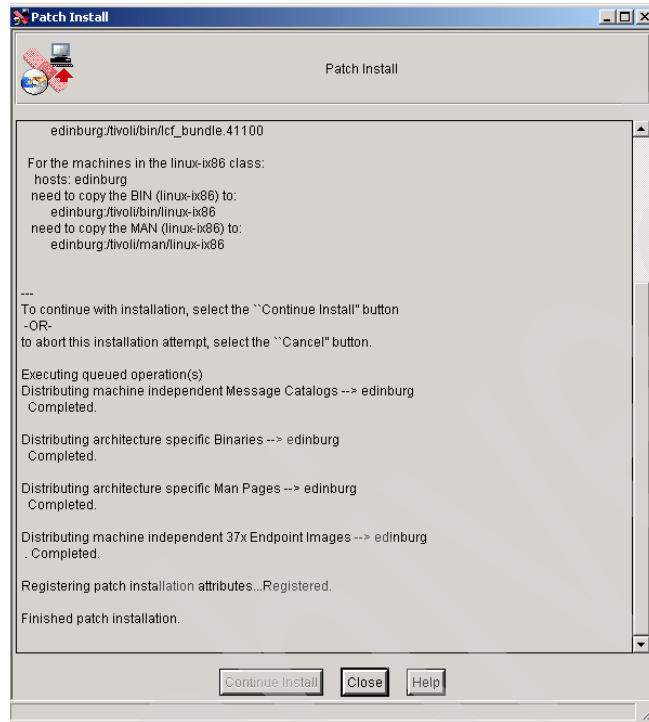


Figure 4-59 Patch installation summary

As shown in Figure 4-59, the Scalable Collection installed successfully.

4.8.2 Installing the Inventory Gateway, Version 4.2.2

Perform the following steps to install the Inventory Gateway:

1. Open the Tivoli desktop, as described in 4.6.2, "Opening the Tivoli desktop" on page 154.
2. Select **Desktop** → **Install** → **Install Product**, as shown in Figure 4-60 on page 178.

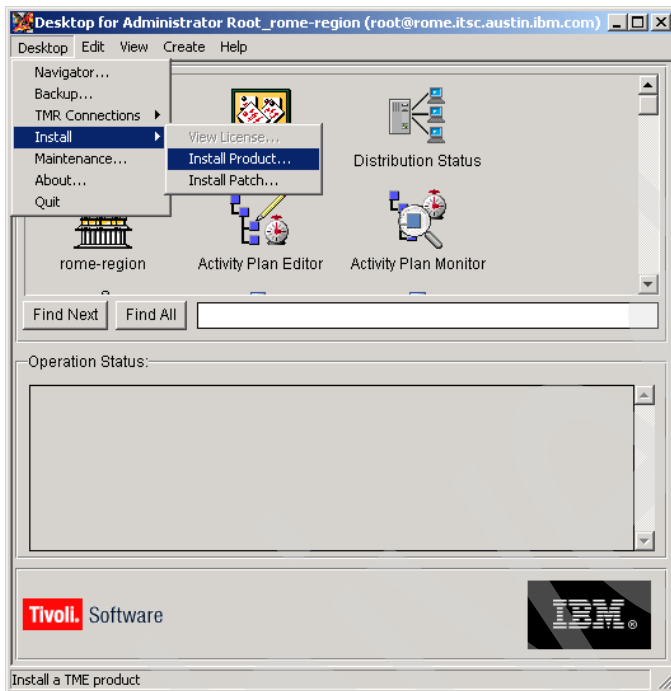


Figure 4-60 Install Product

3. An error message might be displayed when the Tivoli desktop cannot find the file contents.lst, as shown in Figure 4-61. It is just a warning message. Click **OK**.

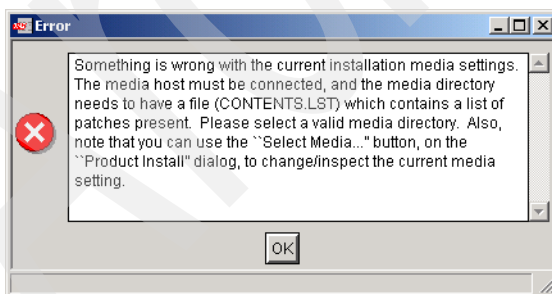


Figure 4-61 Contents.lst error

4. Click **Select Media**, as shown in Figure 4-62.

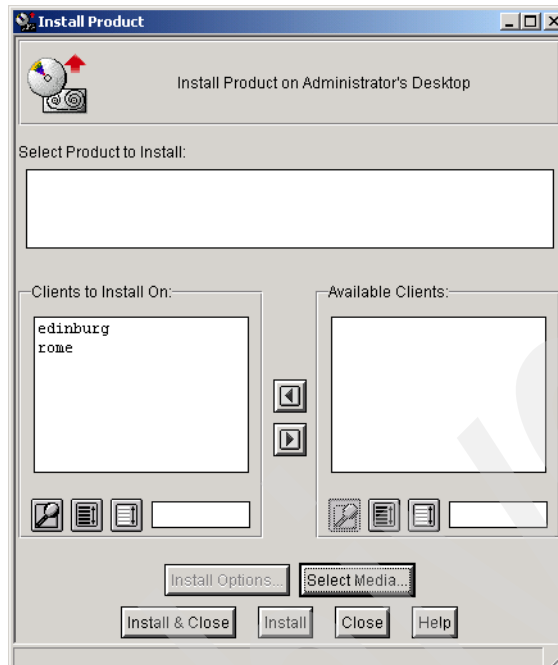


Figure 4-62 *Install Product* window

5. In Figure 4-63, select the path where the Inventory directory is located and click **Set Media & Close**.

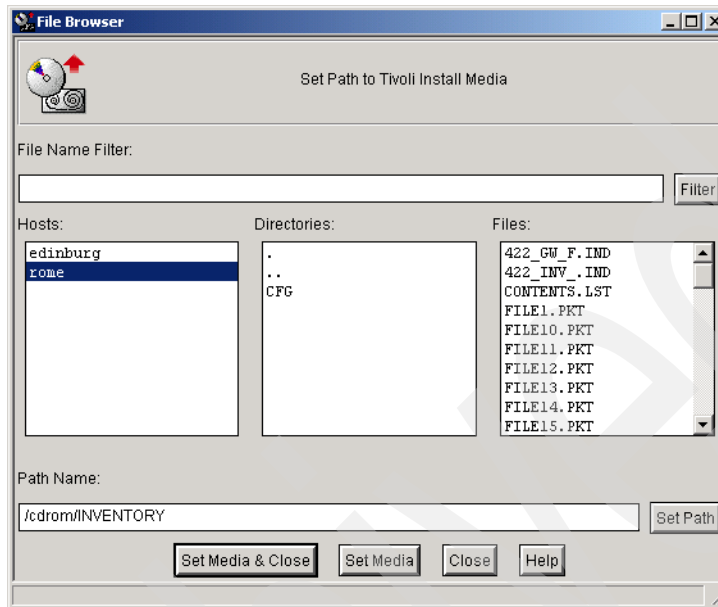


Figure 4-63 Selecting the Inventory directory

6. In Figure 4-64, select the product to be installed and make sure that the gateway is located in the Clients to Install On area. Click the **Install** or **Install & Close** button.

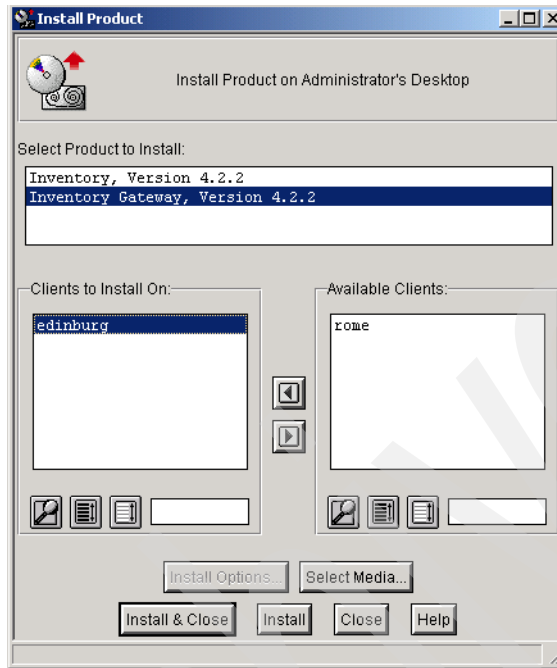


Figure 4-64 Selecting the product and clients

7. Figure 4-65 shows the actions for the product installation. Click **Continue Install** to start the installation process.

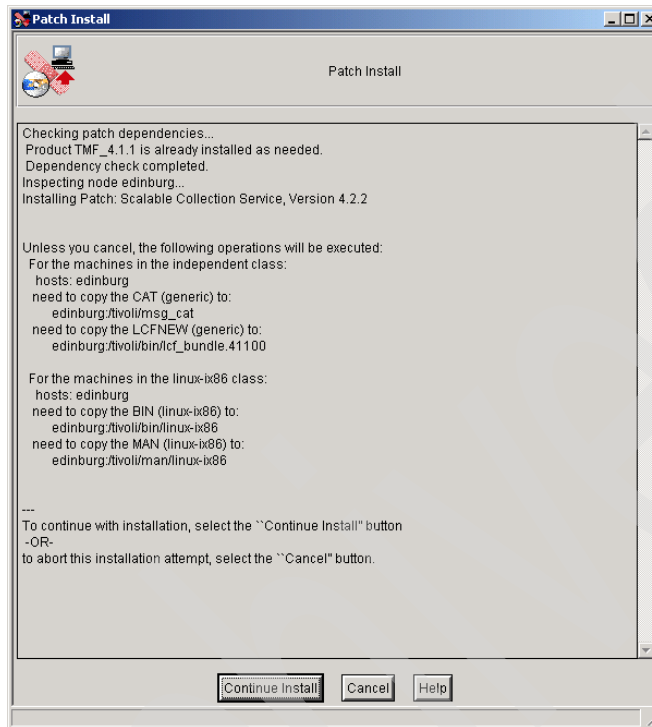


Figure 4-65 Product Install actions

8. Figure 4-66 shows the status of the installation process. Click **Close** to finish the installation.



Figure 4-66 Product installation summary

As shown in Figure 4-66, the Inventory Gateway installed successfully.

4.8.3 Installing the Pristine Manager Gateway, Version 4.2.2

Perform the following steps to install the Pristine Manager Gateway:

1. Open the Tivoli desktop, as described in 4.6.2, “Opening the Tivoli desktop” on page 154.
2. Select **Desktop** → **Install** → **Install Product**, as shown in Figure 4-67 on page 184.

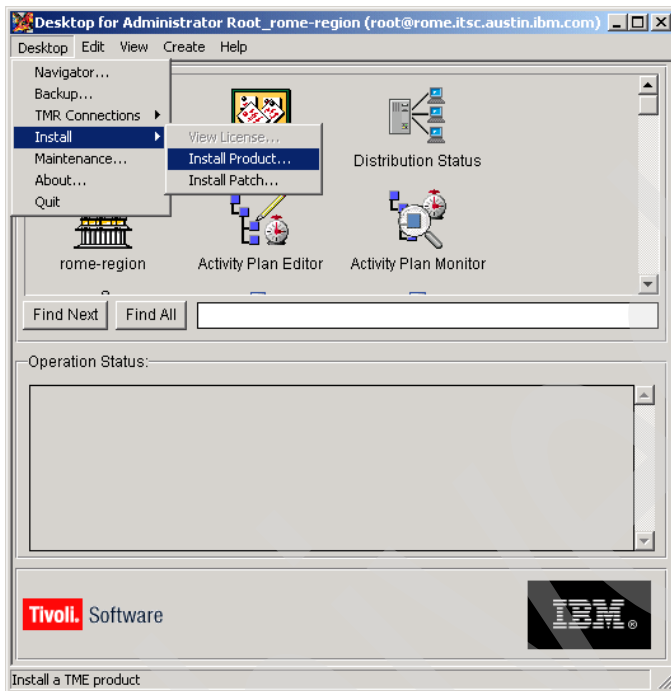


Figure 4-67 Install Product

3. An error message might be displayed when the Tivoli desktop cannot find the file contents.lst, as shown in Figure 4-68. It is just a warning message. Click **OK**.

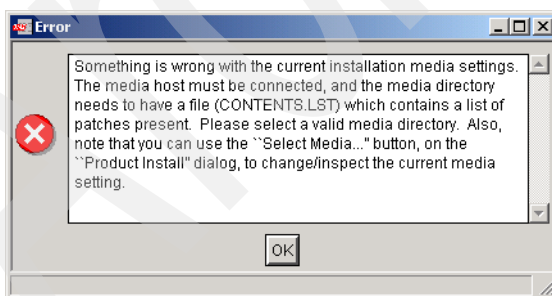


Figure 4-68 Contents.lst error

4. Click **Select Media**, as shown in Figure 4-69.



Figure 4-69 Install Product window

5. In Figure 4-70, select the path where the Pristine directory is located and click **Set Media & Close**.

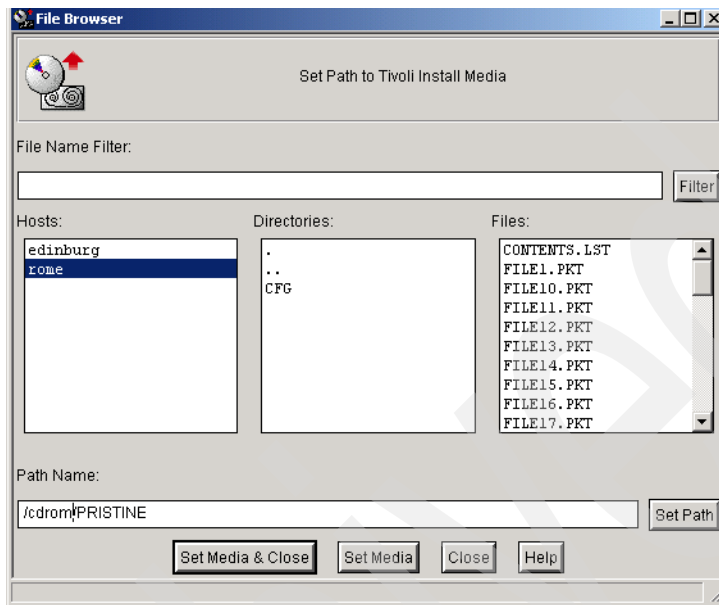


Figure 4-70 Selecting the Pristine directory

6. In Figure 4-71, select the product to be installed and make sure that the gateway is in the Clients to Install On area. Click the **Install** or **Install & Close** button.

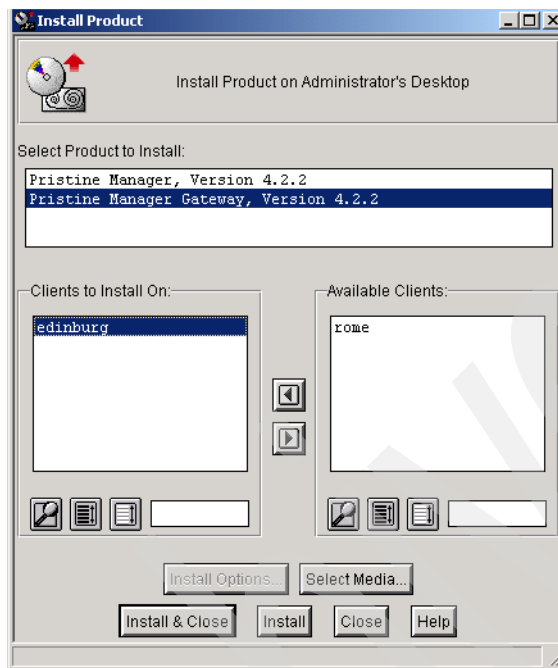


Figure 4-71 Selecting the product and clients

7. Figure 4-72 shows the actions for the product installation. Click **Continue Install** to start the installation process.

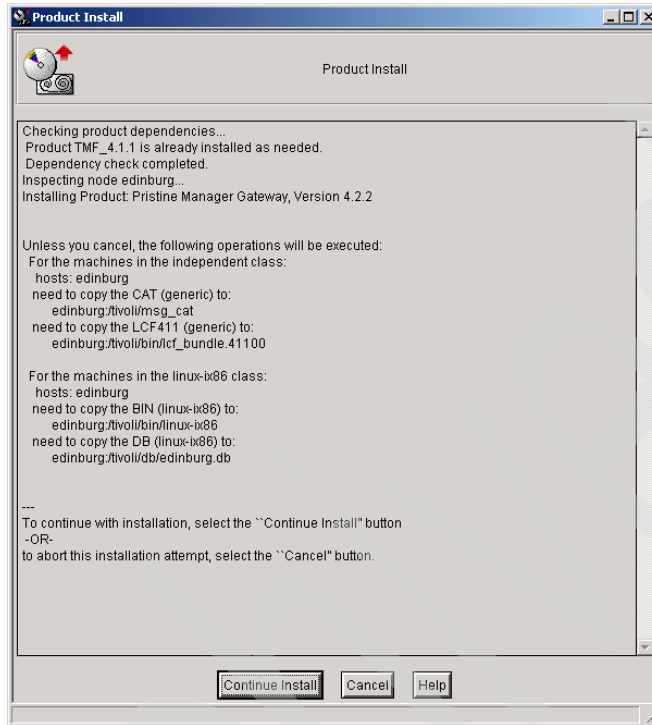


Figure 4-72 Product Install actions

8. Figure 4-73 shows the status of the installation process. Click **Close** to finish the installation.

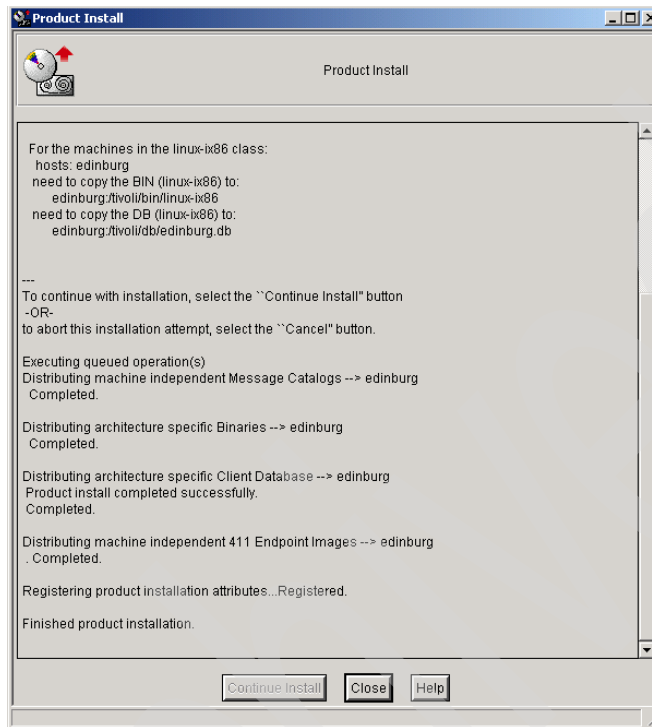


Figure 4-73 Product installation summary

As shown in Figure 4-73, the Pristine Manager Gateway installed successfully.

4.8.4 Installing the Resource Manager Gateway, Version 4.2.2

Perform the following steps to install the Resource Manager Gateway:

1. Open the Tivoli desktop, as described in 4.6.2, "Opening the Tivoli desktop" on page 154.
2. Select **Desktop** → **Install** → **Install Product**, as shown in Figure 4-74 on page 190.

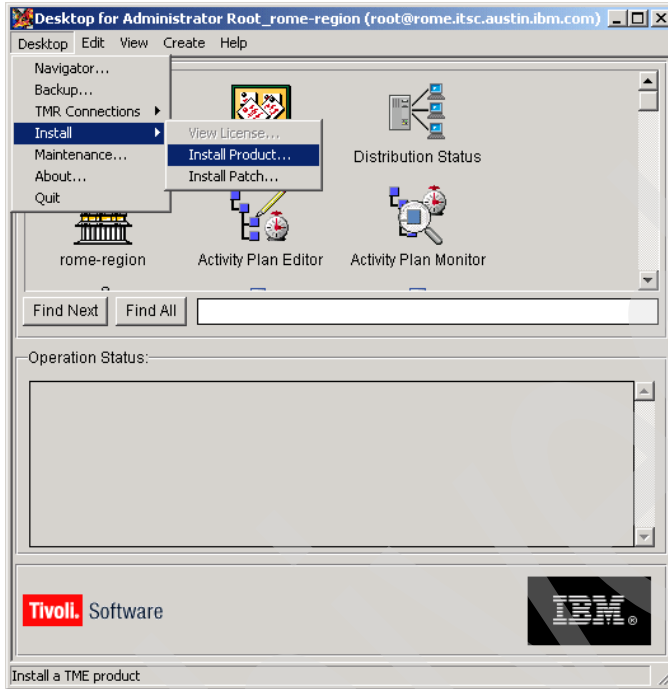


Figure 4-74 Install Product

3. An error message might be displayed when the Tivoli desktop cannot find the file contents.lst, as shown in Figure 4-75. It is just a warning message. Click **OK**.

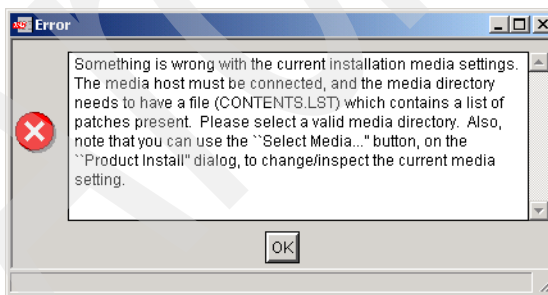


Figure 4-75 Contents.lst error

4. Click **Select Media**, as shown in Figure 4-76.

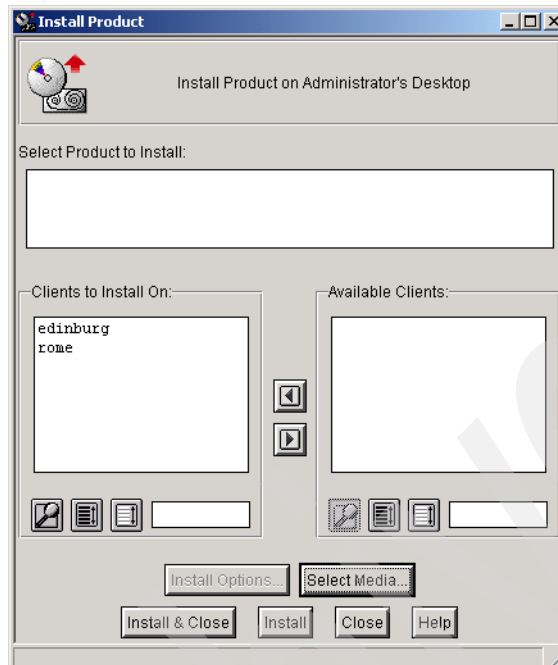


Figure 4-76 *Install Product window*

5. In Figure 4-77, select the path where the Tivoli region directory is located and click **Set Media & Close**.

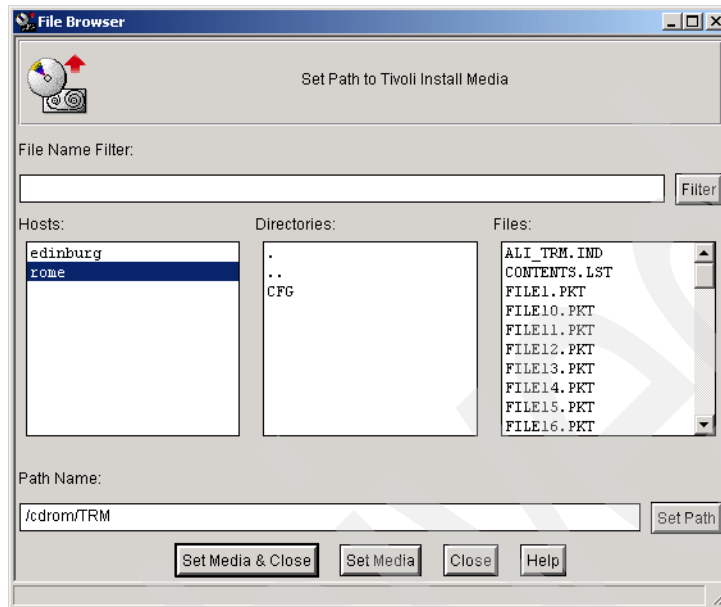


Figure 4-77 Selecting the TRM directory

6. In Figure 4-78, select the product to be installed and make sure that the gateway is located in the Clients to Install On area. Click the **Install** or **Install & Close** button.

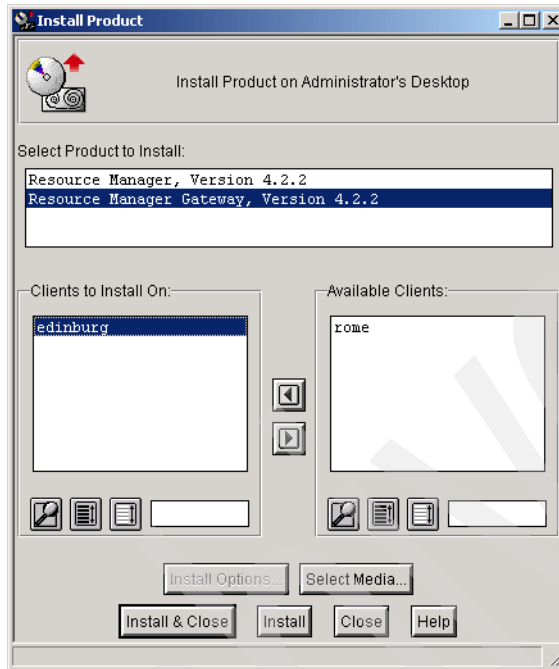


Figure 4-78 Selecting the product and clients

7. Figure 4-79 shows the actions for the product installation. Click **Continue Install** to start the installation process.



Figure 4-79 Product Install actions

8. Figure 4-80 shows the status of the installation process. Click **Close** to finish the installation.

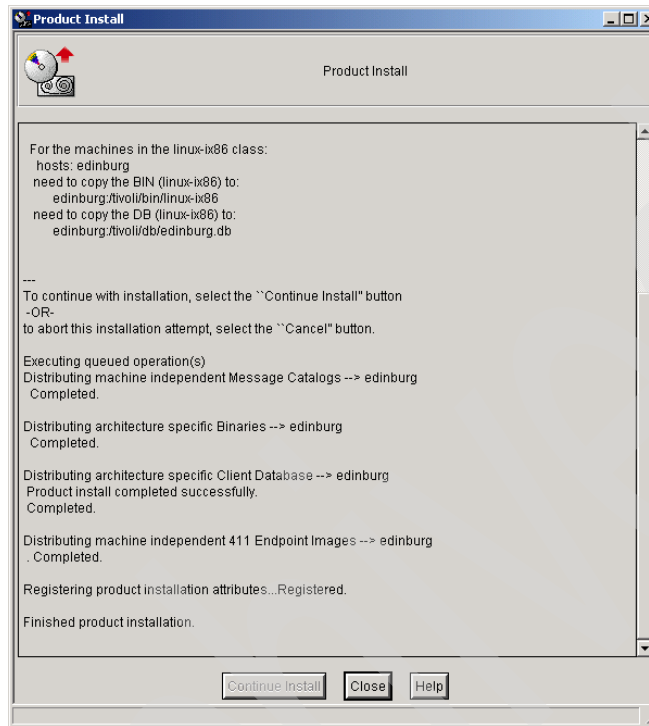


Figure 4-80 Product installation summary

As shown in Figure 4-80, the Resource Manager Gateway installed successfully.

4.8.5 Installing the Software Distribution Gateway, Version 4.2.2

Perform the following steps to install the Software Distribution Gateway:

1. Open the Tivoli desktop, as described in 4.6.2, "Opening the Tivoli desktop" on page 154.
2. Select **Desktop** → **Install** → **Install Product**, as shown in Figure 4-81 on page 196.

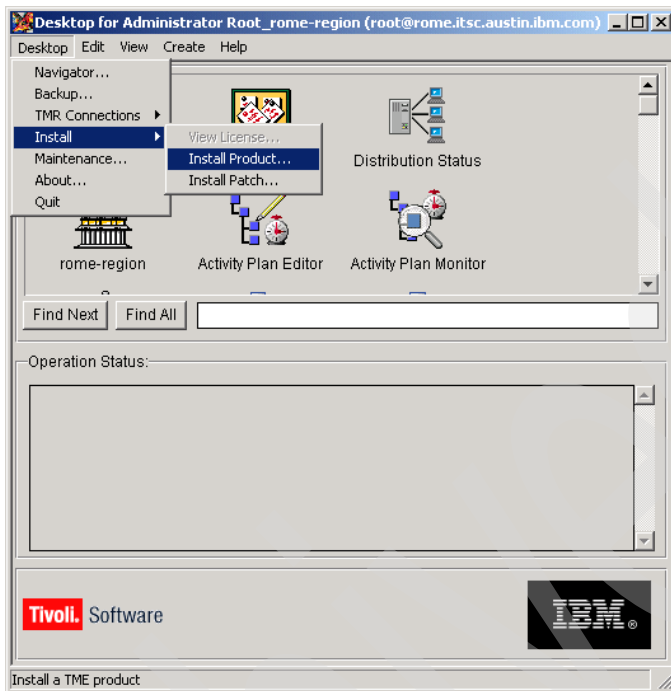


Figure 4-81 Install Product

3. An error message might be displayed when the Tivoli desktop cannot find the file contents.lst, as shown in Figure 4-82. It is just a warning message. Click **OK**.

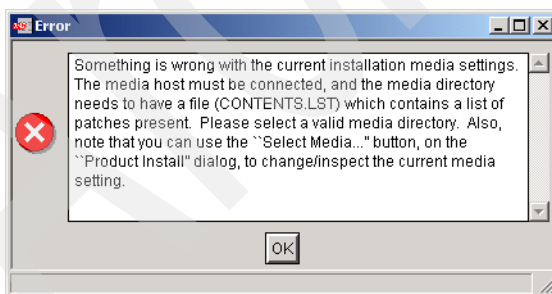


Figure 4-82 Contents.lst error

4. Click **Select Media**, as shown in Figure 4-83.

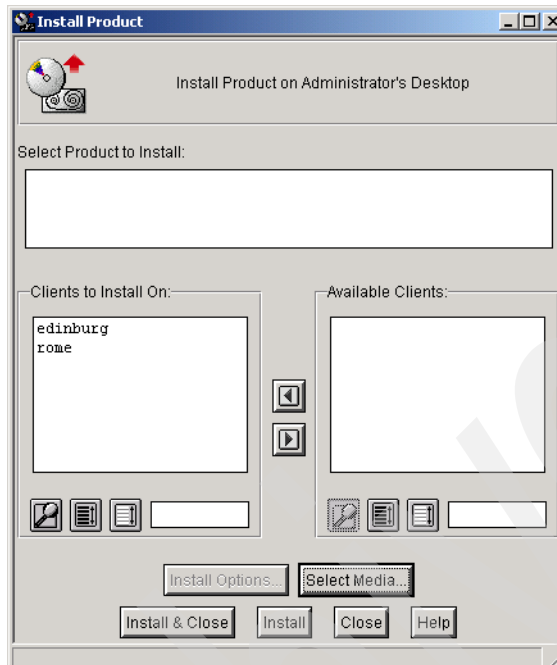


Figure 4-83 *Install Product window*

5. In Figure 4-84, select the path where the SWD directory is located and click **Set Media & Close**.

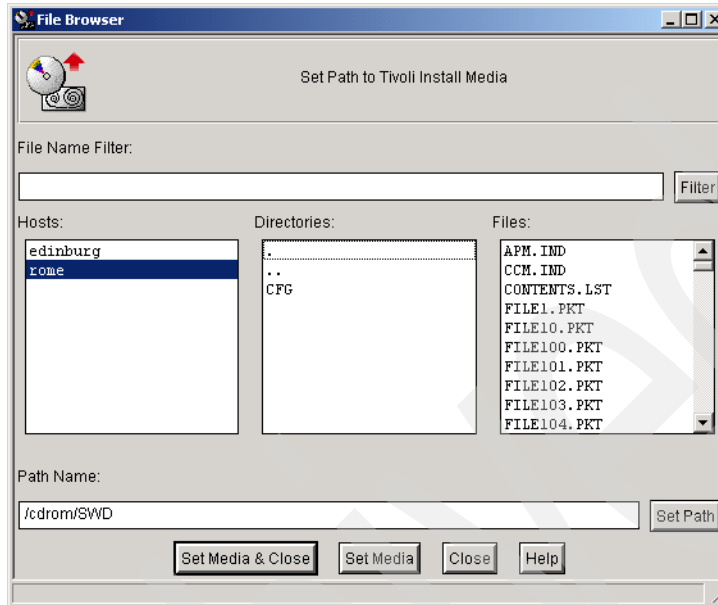


Figure 4-84 Selecting the SWD directory

- In Figure 4-85, select the product to be installed and make sure that the gateway is located in the Clients to Install On area. Click the **Install** or **Install & Close** button.

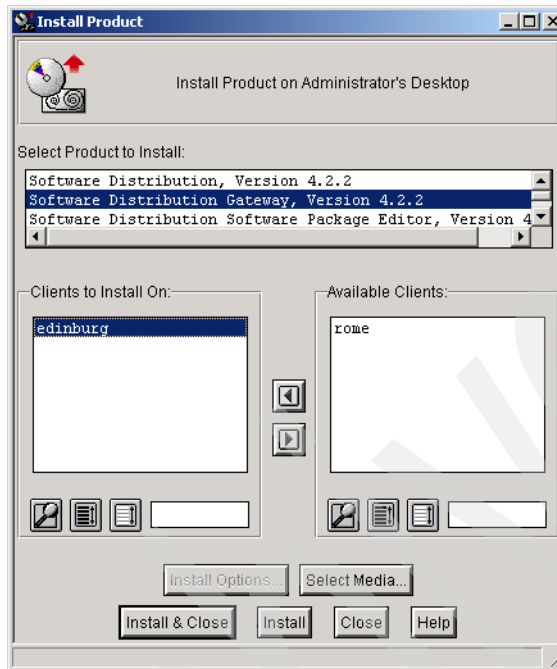


Figure 4-85 Selecting the product and clients

7. Figure 4-86 shows the actions for the product installation. Click **Continue Install** to start the installation process.



Figure 4-86 Product Install actions

8. Figure 4-87 on page 201 shows the status of the installation process. Click **Close** to finish the installation.

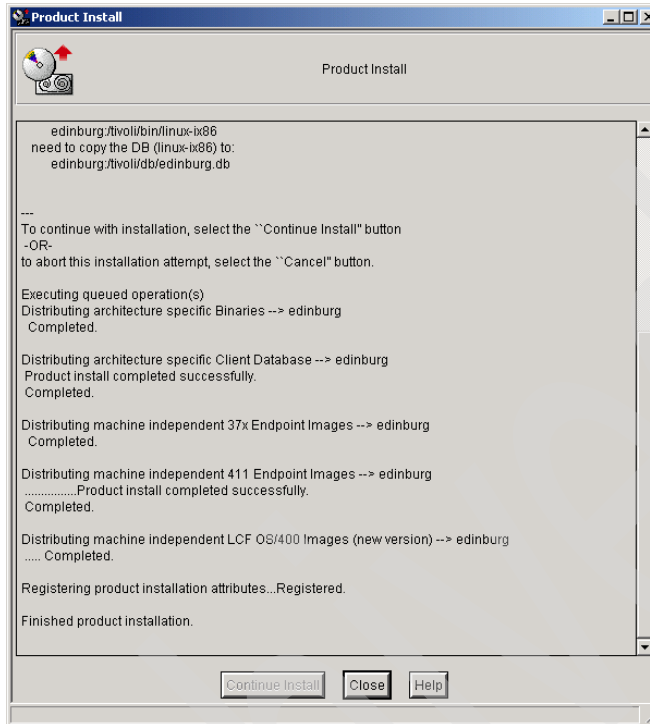


Figure 4-87 Product installation summary

As shown in Figure 4-87, the Software Distribution Gateway installed successfully.

4.8.6 Checking the installed products

To verify that all the products and the patch installed successfully and view the machines on which they were installed, run the `wlsinst` command, as shown in Example 4-12.

Example 4-12 Running the `wlsinst -ah` command

```
# wlsinst -ah
*-----*
*                                     Product List                                     *
*-----*

Tivoli Management Framework 4.1.1
  edinburg      linux-ix86
  rome          aix4-r1
```

Inventory Gateway, Version 4.2.2
 edinburg linux-ix86
 rome aix4-r1

Inventory, Version 4.2.2
 rome aix4-r1

Tivoli Java Client Framework 4.1.1
 rome aix4-r1

Java 1.3 for Tivoli
 rome aix4-r1

Tivoli Java RDBMS Interface Module (JRIM) 4.1.1
 rome aix4-r1

JavaHelp 1.0 for Tivoli
 rome aix4-r1

Pristine Manager, Version 4.2.2
 rome aix4-r1

Pristine Manager Gateway, Version 4.2.2
 edinburg linux-ix86
 rome aix4-r1

Resource Manager, Version 4.2.2
 rome aix4-r1

Resource Manager Gateway, Version 4.2.2
 edinburg linux-ix86
 rome aix4-r1

Activity Planner, Version 4.2.2
 rome aix4-r1

Change Manager, Version 4.2.2
 rome aix4-r1

Directory Query, Version 4.2.2
 rome aix4-r1

Distribution Status Console, Version 4.1.1
 rome aix4-r1

Software Distribution, Version 4.2.2
 rome aix4-r1

Software Distribution Gateway, Version 4.2.2

edinburg	linux-ix86
rome	aix4-r1

Software Distribution Software Package Editor, Version 4.2.2

rome	aix4-r1
------	---------

Web Interface, Version 4.2.2

rome	aix4-r1
------	---------

Patch List

Java 1.3 for Tivoli, Patch 2

rome	aix4-r1
------	---------

Scalable Collection Service, Version 4.2.2

edinburg	linux-ix86
rome	aix4-r1

4.9 Installing endpoints

This section covers the endpoint installation procedure for the Microsoft Windows, Linux, IBM AIX 5L, and IBM OS/400 operating systems.

4.9.1 Installing endpoints on a Windows operating system

You can install Windows endpoints either locally or from the Tivoli server. Locally, we use the **setup.exe** command available on *Tivoli Management Framework Version 4.1.1 CD 2/2*, and remotely, we run the **winstlcf** command.

Installing the Windows endpoint locally

Perform the following steps to install the Windows endpoint called kcwc12z connecting to the gateway rome-gw:

1. Insert the CD called *Tivoli Management Framework 4.1.1 CD 2/2* in the CD-ROM device and run the **setup.exe** command located in the LCF\WINNT directory, as shown in Figure 4-88 on page 204.

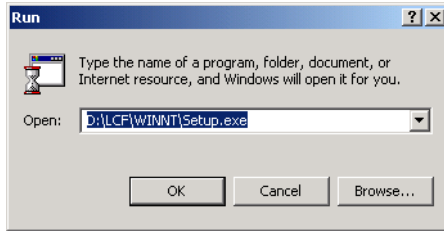


Figure 4-88 Running the setup.exe command

2. The Welcome window opens, as shown in Figure 4-89. Click **Next** to continue.

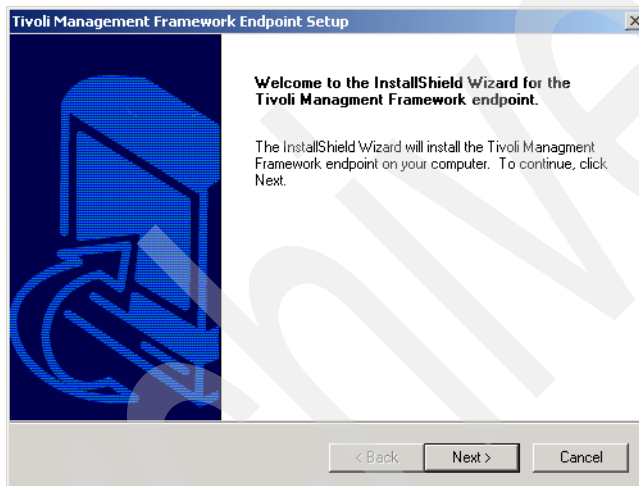


Figure 4-89 Welcome window

3. Read the license agreement, as shown in Figure 4-90, and click **Yes** to continue the installation.

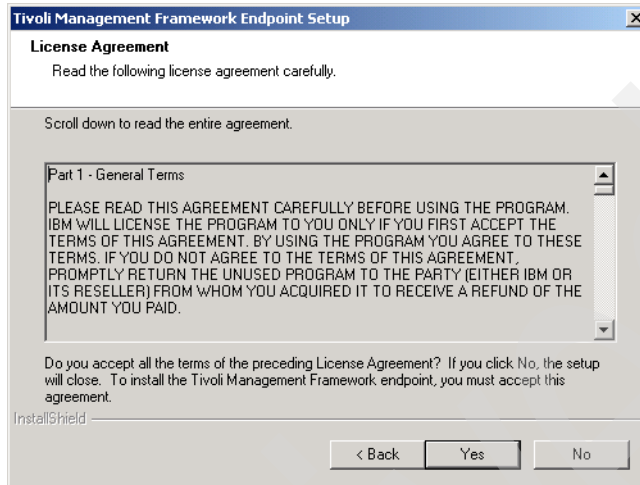


Figure 4-90 License Agreement

4. Figure 4-91 shows the accounts that will be created. Click **Next** to continue.

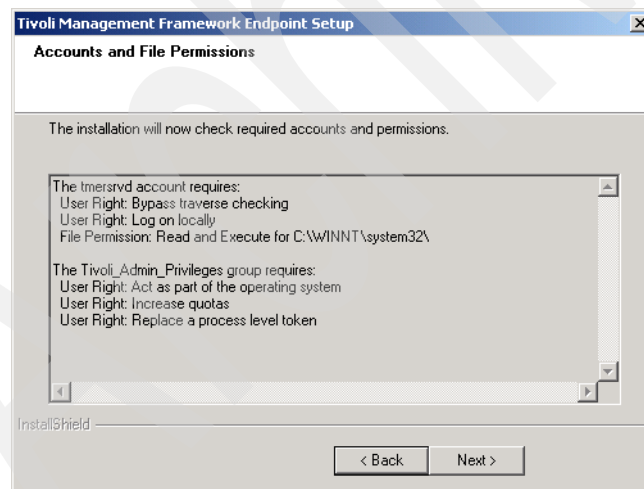


Figure 4-91 Accounts and File Permissions

5. Select the destination directory, as shown in Figure 4-92, and click **Next** to continue.

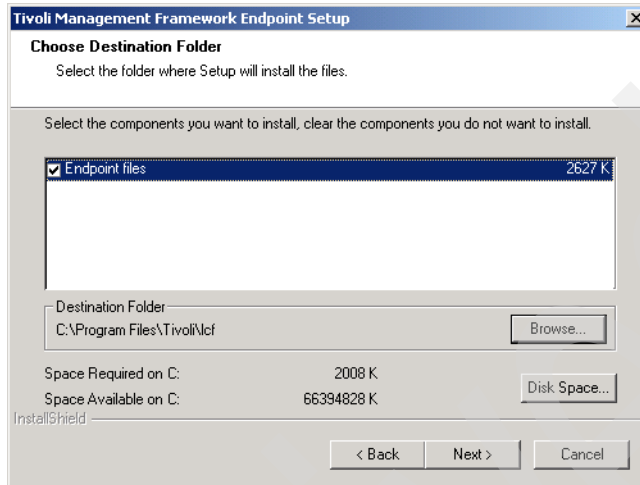


Figure 4-92 Destination directory

6. The window shown in Figure 4-93 requests a Tivoli remote access account. This user is not needed unless you are installing the product from a remote drive. Leave the fields empty and click **Next** to continue.

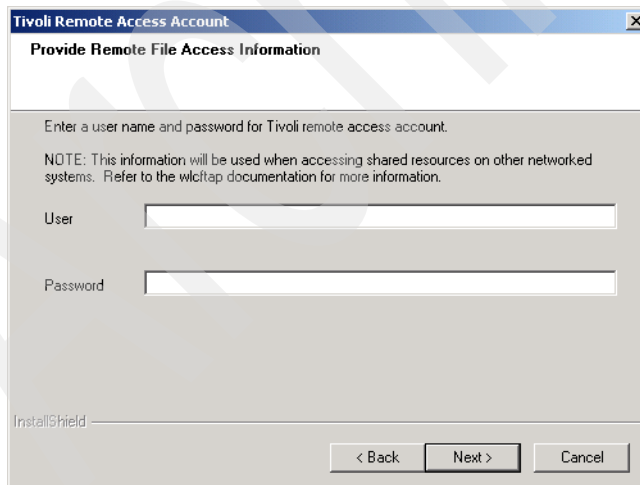


Figure 4-93 Tivoli remote access account information

7. In Figure 4-94, enter 9494 for the Gateway port, 9495 for the Endpoint port, and -g rome+9494 to indicate the host name of the gateway where the endpoint will connect in the Options field. Click **Next** to continue.

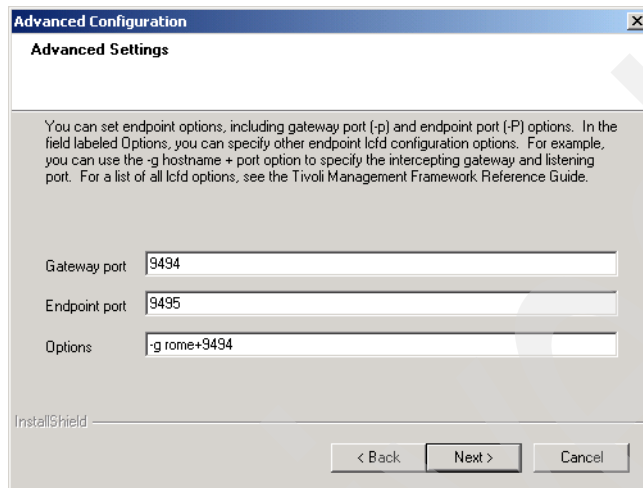


Figure 4-94 Endpoint configuration options

8. Figure 4-95 shows the actions that will be done. Click **Next** to continue.

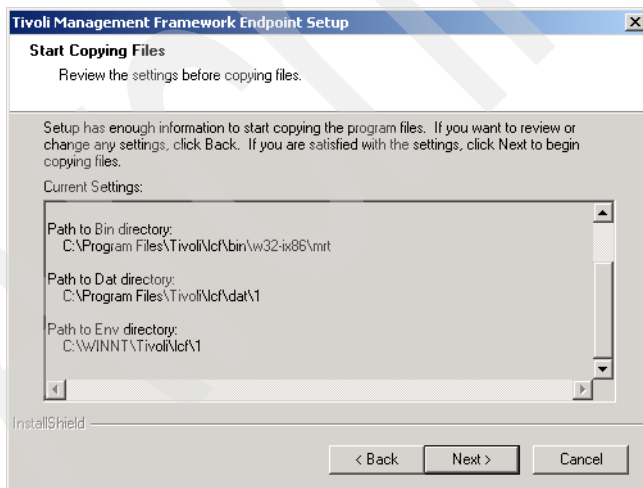


Figure 4-95 Actions to be done

9. After copying the files, the lcmd process will start on the new endpoint and try to connect to the gateway. Figure 4-96 shows the successful the connection window. Click **Next** to continue.

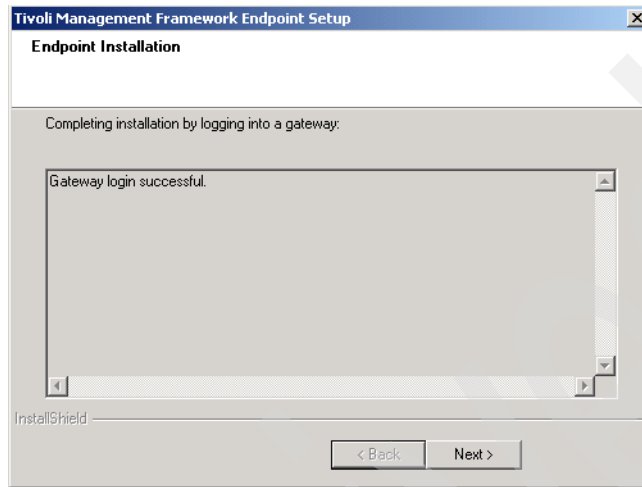


Figure 4-96 Login status

10. You need to reboot after the installation, as shown in Figure 4-97. Select **Yes** and then click **Finish** to reboot Windows and terminate the installation process.

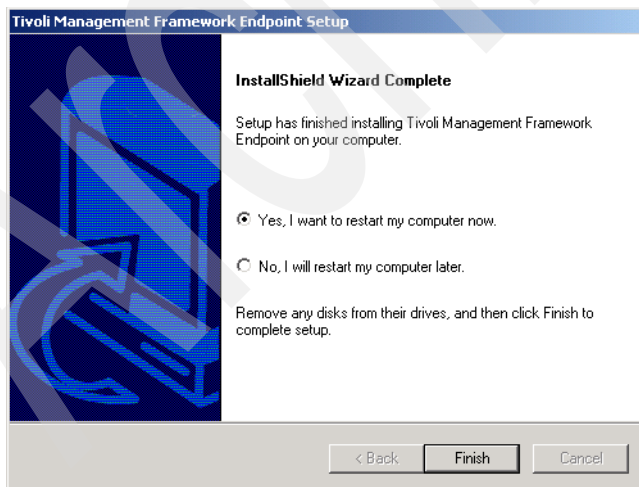


Figure 4-97 Reboot window

Installing the Windows endpoint using the `winstlcf` command

You should have at least one Windows endpoint installed on your Tivoli environment before using the `winstlcf` command for a new Windows endpoint installation. This previously installed Windows endpoint serves as a proxy for the `winstlcf` command when you install endpoints for Windows.

Perform the following steps to install the Windows endpoint called `lizbon`, connecting to the gateway `edinburg-gw`:

1. Make sure that you already have a Windows endpoint installed to serve as a proxy. In our case, we use the endpoint `kcwc12z` as a proxy.
2. The forward and reverse resolution must be configured among the endpoint, Tivoli server, and gateways.
3. Run the `winstlcf` command:

```
winstlcf -g gateway+9494 -N proxy -S C$ -d "C:\Tivoli\lcf" target
```

Where:

- g** Indicates the host name plus port of the gateway where the endpoint will connect.
- N** Indicates the endpoint to serve as a proxy for the installation.
- S** Indicates the destination share directory. The default is `C$`, but it was needed according to APAR IY62898.
- d** Indicates the destination directory for the endpoint installation. The default is `C:\Tivoli\lcf`, but it is needed according to APAR IY62898.
- target** Indicates the host name of the new endpoint.

Example 4-13 shows the installation of the `lizbon` endpoint connecting to the gateway `edinburg`.

Example 4-13 Running the `winstlcf` command to install a Windows endpoint

```
# winstlcf -g edinburg+9494 -N kcwc12z -S C$ -d "C:\Tivoli\lcf" lizbon
```

```
Trying lizbon...
```

```
password for Administrator:
```

```
*****
```

```
All target endpoint operating systems must be Windows (w32-ix86)...
```

```
Ready to copy files to host lizbon:
```

```
source:
```

```
rome:/Tivoli/bin/aix4-r1/./lcf_bundle.41100:/Tivoli/bin/aix4-r1
```

```
././lcf_bundle.41000:/Tivoli/bin/aix4-r1/./lcf_bundle.40
```

```
destination: lizbon:C:/Tivoli/lcf
```

```
files:
```

```
bin/w32-ix86/mrt/lcfd.exe
bin/w32-ix86/mrt/lcfep.exe
bin/w32-ix86/mrt/lcfdiag.exe
bin/w32-ix86/mrt/epproto.dat
bin/w32-ix86/mrt/TivoliAP.dll
bin/w32-ix86/mrt/wlcfcp.exe
bin/w32-ix86/mrt/ntconfig.exe
lib/w32-ix86/mrt/libacct60.dll
lib/w32-ix86/mrt/libatrc.dll
lib/w32-ix86/mrt/libcpl.dll
lib/w32-ix86/mrt/libcpl60.dll
lib/w32-ix86/mrt/libdes.dll
lib/w32-ix86/mrt/libdes60.dll
lib/w32-ix86/mrt/libipx60.dll
lib/w32-ix86/mrt/libmem60.dll
lib/w32-ix86/mrt/libmd2ep60.dll
lib/w32-ix86/mrt/libtos.dll
lib/w32-ix86/mrt/libtthred.dll
lib/w32-ix86/mrt/libtcp60.dll
lib/w32-ix86/mrt/libmrt.dll
lib/w32-ix86/mrt/libmrt60.dll
lib/w32-ix86/mrt/libguid60.dll
lib/w32-ix86/mrt/libccms_lcf.dll
lib/w32-ix86/mrt/libccms60_lcf.dll
lib/w32-ix86/mrt/msvcrt40.dll
lib/w32-ix86/mrt/msvcrt.dll
lib/w32-ix86/mrt/msvcirt.dll
```

Note: dll's will be moved to bin/w32-ix86/mrt

Continue? [yYna?]y

Endpoint(s) on lizbon installed. Configuring...

Reboot of lizbon needed to complete installation.

Proceed? [yYn?]n

Not
rebooting.

After Windows reboots, the installation process is finished.

4.9.2 Installing endpoints on a Linux operating system

The installation of an endpoint on Linux is done through the `winstlcf` command. This section shows you how to install the Linux endpoint called barcelona in our scenario. This endpoint will also connect to the gateway edinburgh.

Perform the following steps to install the Linux endpoint:

1. Make sure that the rexec is working on the Linux machine on which you will install the endpoint. To do that, see 4.7.1, “Linux considerations” on page 156.
2. Run the **winstlcf** command:

```
winstlcf -g gateway+9494 target
```

Where:

-g Indicates the host name plus port of the gateway where the endpoint will connect.

target Indicates the host name of the new endpoint

Example 4-14 shows the installation of the barcelona endpoint connecting to the gateway edinburg.

Example 4-14 Installation of the barcelona endpoint

```
# winstlcf -g edinburg+9494 barcelona
```

```
Trying barcelona...
```

```
password for root:
```

```
*****
```

```
locating files in /Tivoli/bin/lcf_bundle.41100...
```

```
locating files in /Tivoli/bin/lcf_bundle...
```

```
Ready to copy files to host barcelona:
```

```
destination: barcelona:/opt/Tivoli/lcf
```

```
source: rome:/Tivoli/bin/lcf_bundle.41100
```

```
files:
```

```
generic/lcfd.sh
generic/epinst.sh
generic/as.sh
generic/lcf_env.sh
generic/lcf_env.csh
generic/lcf_env.cmd
generic/lcf.inv
bin/linux-ix86/mrt/lcfd
lib/linux-ix86/libatrc.so
lib/linux-ix86/libcpl272.so
lib/linux-ix86/libdes272.so
lib/linux-ix86/libmd2ep272.so
lib/linux-ix86/libguid272.so
lib/linux-ix86/libmrt272.so
lib/linux-ix86/libtis272.so
lib/linux-ix86/libtos.so
lib/linux-ix86/libtthred.so
```

```
Continue? [yYna?]y
```

```
Tivoli Light Client Framework starting on barcelona
Jun 21 14:07:08 1 lcf Command line
argv[0]='/opt/Tivoli/lcf/bin/linux-ix86/mrt/lcf'
Jun 21 14:07:08 1 lcf Command line
argv[1]='-Dlcs.login_interfaces=edinburg+9494'
Jun 21 14:07:08 1 lcf Command line
argv[2]='-Dlib_dir=/opt/Tivoli/lcf/lib/linux-ix86'
Jun 21 14:07:08 1 lcf Command line
argv[3]='-Dload_dir=/opt/Tivoli/lcf/bin/linux-ix86/mrt'
Jun 21 14:07:08 1 lcf Command line argv[4]='-C/opt/Tivoli/lcf/dat/1'
Jun 21 14:07:08 1 lcf Command line argv[5]='-Dlcs.machine_name=barcelona'
Jun 21 14:07:08 1 lcf Command line
argv[6]='-Dlcs.login_interfaces=edinburg+9494'
Jun 21 14:07:08 1 lcf Starting Unix daemon
Performing auto start configuration
Tivoli LCF daemon master autostart file is /etc/init.d/Tivoli_lcf1.
Done.
```

4.9.3 Installing endpoints on an AIX 5L operating system

The installation of an endpoint on AIX 5L is done through the `winstlcf` command. This section shows you how to install the endpoint called `madrid` on an AIX 5L environment. This endpoint will also connect to the gateway `edinburg`.

Perform the following steps to install the AIX 5L endpoint:

1. Verify that there is 100 KB of free space on the / file system and 2 MB of free space on the /opt file system.
2. Run the `winstlcf` command:

```
winstlcf -g gateway+9494 target
```

Where:

-g Indicates the host name plus port of the gateway where the endpoint will connect.

target Indicates the host name of the new endpoint.

Example 4-15 shows the installation of the `madrid` endpoint connecting to the gateway `edinburg`.

Example 4-15 Running the `winstlcf` command to install an AIX 5L endpoint

```
# winstlcf -g edinburg+9494 madrid
```

```
Trying madrid...
```

```
password for root:
```

```
*****
```

```
locating files in /Tivoli/bin/lcf_bundle.41100...
locating files in /Tivoli/bin/lcf_bundle...
```

```
Ready to copy files to host madrid:
  destination: madrid:/opt/Tivoli/lcf
  source: rome:/Tivoli/bin/lcf_bundle
  files:
    lib/aix4-r1/libmrt.a
    lib/aix4-r1/libcpl.a
    lib/aix4-r1/libdes.a
  source: rome:/Tivoli/bin/lcf_bundle.41100
  files:
    generic/lcfd.sh
    generic/epinst.sh
    generic/as.sh
    generic/lcf_env.sh
    generic/lcf_env.csh
    generic/lcf_env.cmd
    generic/lcf.inv
    bin/aix4-r1/mrt/lcfd
    lib/aix4-r1/libatrc.a
    lib/aix4-r1/libcpl272.a
    lib/aix4-r1/libdes272.a
    lib/aix4-r1/libmd2ep272.a
    lib/aix4-r1/libguid272.a
    lib/aix4-r1/libmrt272.a
    lib/aix4-r1/libtis272.a
    lib/aix4-r1/libtos.a
    lib/aix4-r1/libtthred.a
```

```
Continue? [yYna?]y
```

```
Tivoli Light Client Framework starting on madrid
Jun 21 13:36:27 1 lcfd Command line
argv[0]='/opt/Tivoli/lcf/bin/aix4-r1/mrt/lcfd'
Jun 21 13:36:27 1 lcfd Command line
argv[1]='-Dlcs.login_interfaces=edinburg+9494'
Jun 21 13:36:27 1 lcfd Command line
argv[2]='-Dlib_dir=/opt/Tivoli/lcf/lib/aix4-r1'
Jun 21 13:36:27 1 lcfd Command line
argv[3]='-Dload_dir=/opt/Tivoli/lcf/bin/aix4-r1/mrt'
Jun 21 13:36:27 1 lcfd Command line argv[4]='-C/opt/Tivoli/lcf/dat/1'
Jun 21 13:36:27 1 lcfd Command line argv[5]='-Dlcs.machine_name=madrid'
Jun 21 13:36:27 1 lcfd Command line
argv[6]='-Dlcs.login_interfaces=edinburg+9494'
Jun 21 13:36:27 1 lcfd Starting Unix daemon
Performing auto start configuration
Done.
```

4.9.4 Installing endpoints on an OS/400 operating system

The installation of an endpoint on OS/400 is done through the `w4inslcf` command. This section shows you how to install an endpoint called `as20` in an OS/400 environment. This endpoint will connect to the gateway `rome`.

OS/400 endpoint prerequisites

Before installing the OS/400 endpoint, you must check the following prerequisites:

- ▶ Make sure that the QShell is already installed on the OS/400 system.
- ▶ You must have an OS/400 user with FTP access, SAVSYS authority, and permissions to execute the commands RSTOBJ and RSTLICPGM.

Installing the OS/400 endpoint

Perform the following steps to install the OS/400 endpoint:

1. Log in to the Tivoli region as the root user.
2. Go to the directory `$BINDIR/../lcf_bundle.41100/generic`:

```
cd $BINDIR/../lcf_bundle.41100/generic
```

3. Run the command `w4inslcf`:

```
perl ./w4inslcf -g ip_gateway+9494 -I -s source ip_endpoint
```

Where:

ip_gateway	IP address where the endpoint will connect
source	The source of the endpoint bin files, located in the <code>\$BINDIR/../lcf_bundle.41100</code> directory if you are using the bin files from the Framework installation
ip_endpoint	IP address of the endpoint

Example 4-16 shows the installation of the endpoint named `as20` (IP address 9.3.5.107) connecting to the gateway `rome` (IP address 9.3.5.54). During the installation process, you will be asked for the administrator user with the permissions described in “OS/400 endpoint prerequisites” on page 214 and you will also be asked for an install confirmation.

Example 4-16 The `w4inslcf` command

```
#perl ./w4inslcf.pl -g 9.3.5.54+9494 -I -s  
/Tivoli/bin/aix4-r1/../lcf_bundle.41100 9.3.5.107  
InitConfig done
```

```
We need a user account, please type or just hit enter to use [default]  
Login for 9.3.5.107 ([QSECOFR]): QSECOFR  
Password for account QSECOFR: *****
```

```
Getting files to install from
/Tivoli/bin/lcf_bundle.41100/bin/os400/upgrade/EPUPG.INF
```

Ready to install product:

```
    source: rome:/Tivoli/bin/lcf_bundle.41100
    destination: 9.3.5.107+9495:QTMELCF:/QIBM/UserData/Tivoli/lcf
login gateway: 9.3.5.54+9494
    User: QSECOFR
    files:
        /Tivoli/bin/lcf_bundle.41100/bin/os400/mrt/lcfd
    NLS: 2924
    LCFOPTS:
lcs.login_interfaces=9.3.5.54+9494
gateway_port=9494
```

Continue? [yYna?] Y

Checking Prerequisites on 9.3.5.107

```
Restoring objects in file /Tivoli/bin/lcf_bundle.41100/bin/os400/mrt/lcfd
Command to restore: RSTLICPGM
```

```
Binary restored successfully in 9.3.5.107
File does not exist : /Tivoli/bin/lcf_bundle.41100/nls/os400
Trying : /TIVOLI/BIN/LCF_BUNDLE.41100/NLS/OS400
File does not exist : /TIVOLI/BIN/LCF_BUNDLE.41100/NLS/OS400
Trying : /TIVOLI/BIN/LCF_BUNDLE.41100/NLS/OS400
```

Restoring Languages 2924

```
Language 2924 restored successfully in 9.3.5.107
```

Deleting temporary library (QTMELCFINS)

```
Installation log created /tmp/tivoli.9.3.5.107.w4log
```

4. Configure the endpoint manager to recapture orphaned endpoint:

```
wepmgr set epmgr_flags 1
```

5. Restart the epmgr:

```
wepmgr restart
```

6. Log in to the OS/400 machine as QSECOFR.

7. Go to QSHELL environment:

```
CALL QCMD
```

8. Start the endpoint:

```
STRTMEEPT LGNINTRFC('ip_gateway' 9494) GATEWAY('ip_gateway' 9494)  
EPTNAME(endpoint_name) PORT(endpoint_port)  
OPTIONS('local_ip_interface=ip_endpoint')
```

Where:

- ip_gateway** IP address where the endpoint will connect.
- endpoint_name** A label for the new endpoint; this can be the host name.
- endpoint_port** The communication port of the endpoint. The default is 9495.
- ip_endpoint** IP address of the endpoint.

The following command shows the start of the endpoint installed in Example 4-16 on page 214:

```
STRTMEEPT LGNINTRFC('9.3.5.54' 9494) GATEWAY('9.3.5.54' 9494) EPTNAME(as20)  
PORT(9495) OPTIONS('local_ip_interface=9.3.5.107')
```

9. On the Tivoli region, run the command `wep 1s` to check if the new endpoint is already connected.

Note: In OS/400, the `lcf.d.log` file is in the `/QIBM/UserData/Tivoli/LCF` directory. Run the `wrk1nk` command to navigate through the directories.

That concludes our installation.

Extra components

This chapter provides step-by-step instructions to install the IBM DB2 Universal Database (UDB), LDAP server, WebSphere Application Server, Tivoli Access Manager, and Tivoli Access Manager WebSEAL products on an AIX 5L system.

We discuss the following topics:

- ▶ IBM DB2 UDB installation
- ▶ LDAP server installation
- ▶ IBM WebSphere Application Server installation
- ▶ Tivoli Access Manager and Access Manager WebSEAL installation

5.1 IBM DB2 UDB installation

This section covers the DB2 UDB V8.1 server installation on an AIX 5L environment. The following steps install the DB2 UDB on the /usr file system and create a instance called tivoli in the /DB2Instancefs. This instance will be used later to store all the databases from IBM Tivoli Configuration Manager Version 4.2.2.

Before installing the DB2, create the file system /usr/opt/db2_08_01 with 500 MB to store the DB2 bin files:

```
crfs -v jfs -g rootvg -m'/usr/opt/db2_08_01' -p'rw' -a size='500M' -a frag='4096' -a nbpi='4096' -a ag='8' -A'yes'
```

Now, we can install DB2 UDB V8.1 for AIX binaries:

1. Mount the cdrom on your AIX machine.

```
mount -v cdrfs -r /dev/cd0 /cdrom
```

2. Create a temporary file system on the local machine with 750 MB to store the installation binaries files:

```
smitty crjfsstd  
or  
crfs -v jfs -g rootvg -m'/source_dir' -p'rw' -a size='750M' -a frag='4096' -a nbpi='4096' -a ag='8'
```

3. Mount the new file system:

```
mount /source_dir
```

4. Go to the /source_dir directory:

```
cd /source_dir
```

5. Uncompress the binary file:

```
zcat /cdrom/DB2ESE.V812/ese.sbcsaix1.tar.Z | tar -xf -
```

6. Run the **db2setup** command to start the install process:

```
/source_dir/ese.sbcsaix1/db2setup
```

7. Figure 5-1 on page 219 shows the first window of the DB2 UDB installation. Select **Install Products** from the right menu initiate the install process of DB2 UDB V8.1.

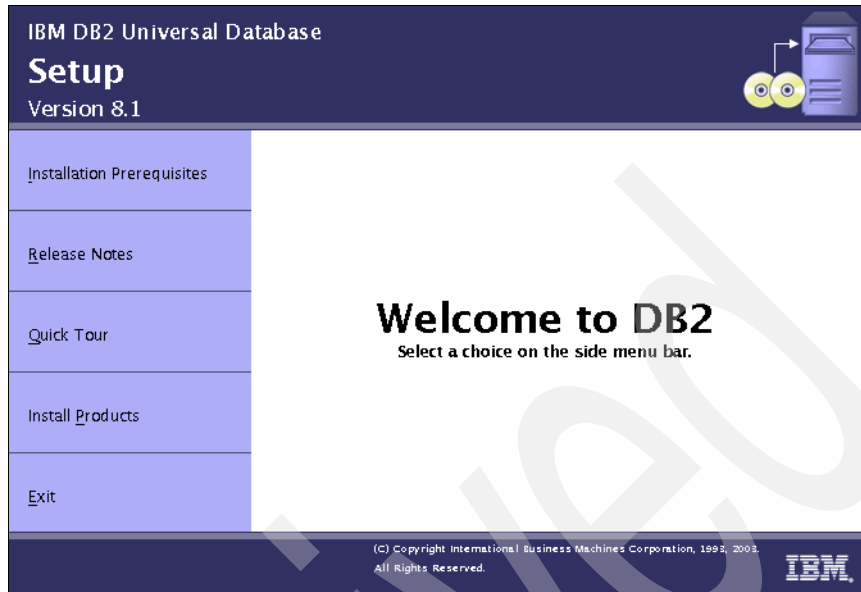


Figure 5-1 Welcome window

8. In Figure 5-2, select **DB2 UDB Enterprise Server Edition** and click **Next** to continue.

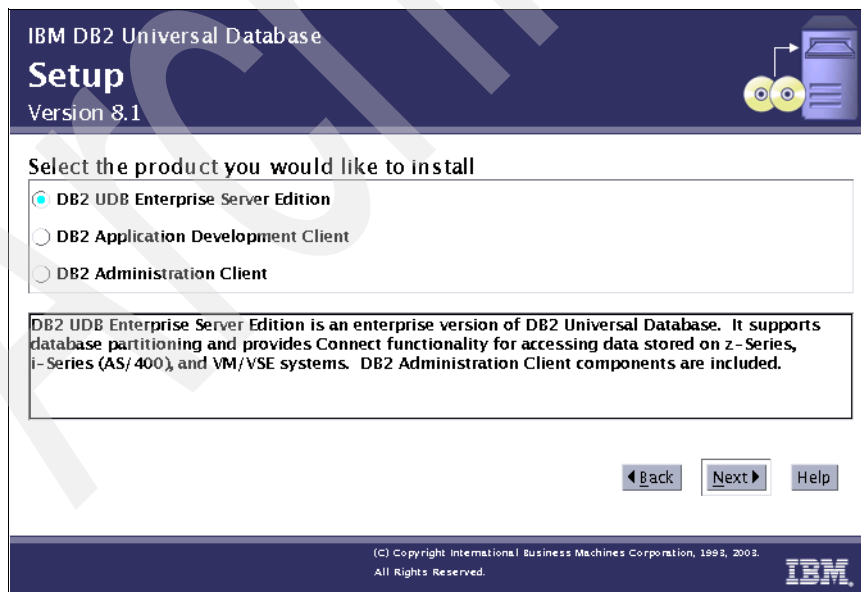


Figure 5-2 Selecting the product to install

9. Click **Next** to continue, as shown in Figure 5-3.



Figure 5-3 Welcome to the DB2 Setup wizard window

10. Read the software license agreement, select **Accept** and click **Next**, as shown in Figure 5-4.



Figure 5-4 Software License Agreement window

11. Select the **Typical** installation and click **Next**, as shown in Figure 5-5.



Figure 5-5 Selecting the installation type

12. Select **Install DB2 UDB Enterprise Server Edition on this computer** and click **Next**, as shown in Figure 5-6.

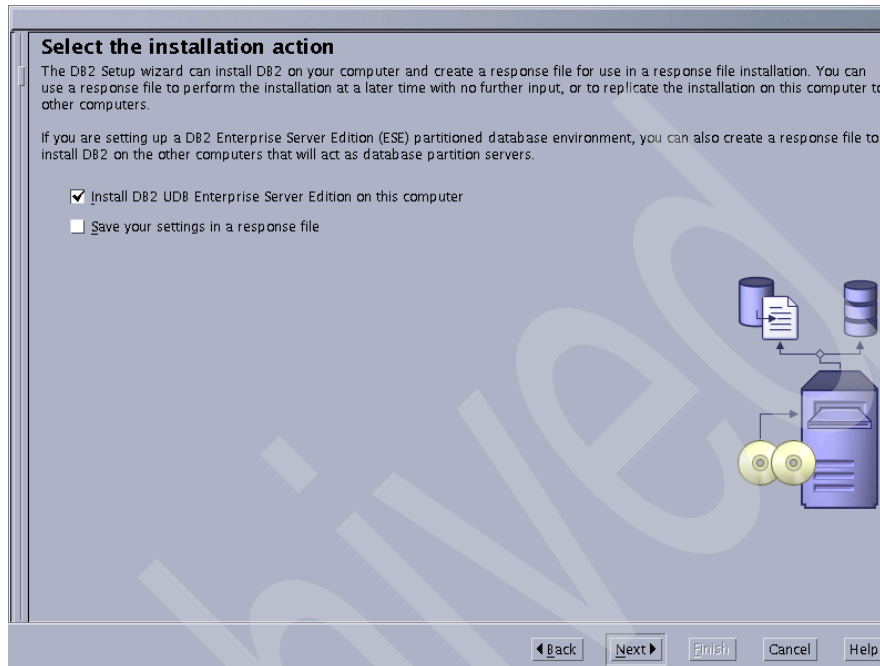


Figure 5-6 Selecting the installation action

13. The user `dasusr1` will be created, as shown in Figure 5-7. Select **New User**, **Use default UID** and **Use default GID**. Set and confirm a password for the `dasusr1` user.

Set user information for the DB2 Administration Server

The DB2 Administration Server (DAS) runs on your computer to provide support required by the DB2 tools. A user account with a minimal set of privileges is required to run the DAS. Specify the required user information for the DAS.

New user

User name:

UID: Use default UID

Group name:

GID: Use default GID

Password:

Confirm password:

Home directory: ...

Existing user

User name: ...

For users of NIS or similar management systems:

If the user information in your environment is managed remotely by NIS or a similar system, you must specify an existing user.

◀ Back Next ▶ Finish Cancel Help

Figure 5-7 Setting the `dasusr1` user

14. Select **Do not create a DB2 instance** and click **Next**, as shown in Figure 5-8.

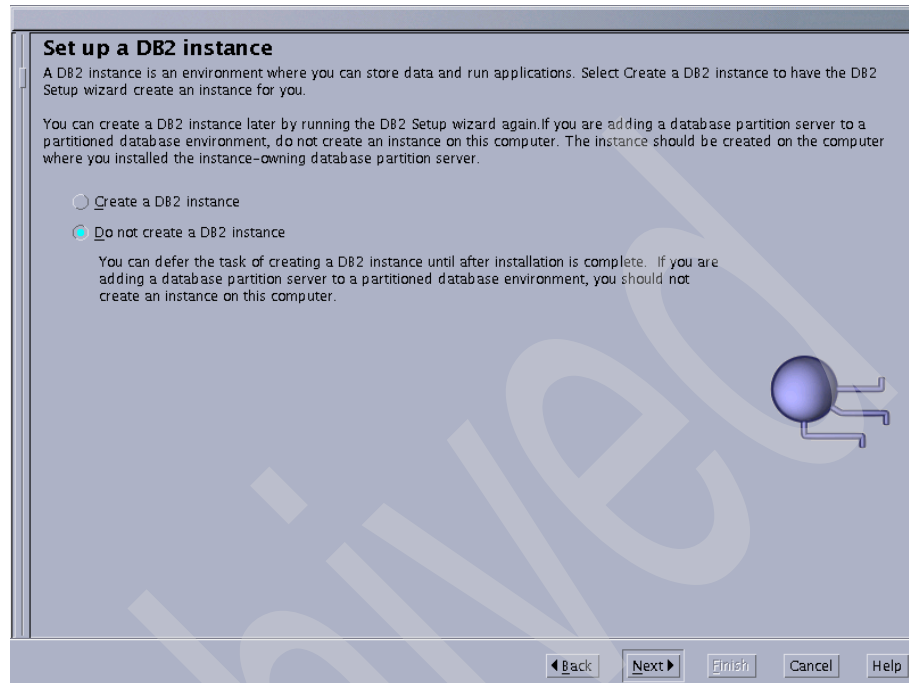


Figure 5-8 Creating a DB2 instance

15. Configure the administration contact list, as shown in Figure 5-9. Select **Local** in the Administration contact list location area, select **Enable notification** in the Notification SMTP server area, enter the host name of the SMTP server in the Notification SMTP server field, and click **Next** to continue.

Set up the administration contact list

The administration contact list will store contact information used to notify administrators that a database requires attention. Specify whether you want to create a local contact list or use an existing global contact list that resides on a remote DB2 server.

Administration contact list location

Local - Create a contact list on this system

Remote - Use an existing contact list that resides on a remote DB2 server

Remote DB2 server host name

Notification SMTP server

Type the SMTP server that will send email and pager notifications to your administration contact.

Enable notification

Notification SMTP server: belfast

Back Next Finish Cancel Help

Figure 5-9 Setting the administration contact list

16. Figure 5-10 shows the DB2 wizard configuration summary. Review the configuration and click **Finish** to begin copying files.

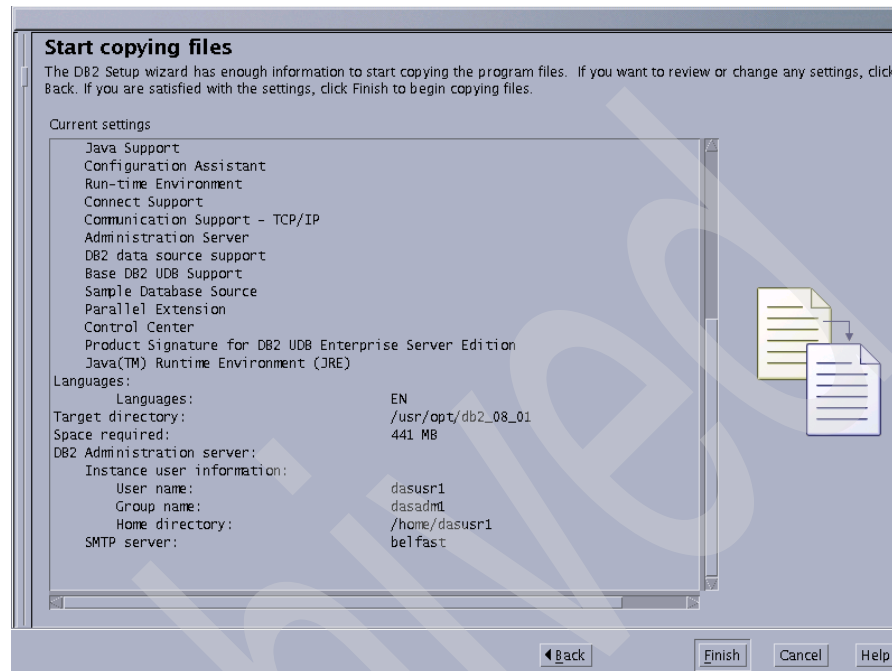


Figure 5-10 DB2 wizard configuration summary

17. Figure 5-11 and Figure 5-12 show the end of the installation process. Click **Finish** to terminate the DB2 wizard installation.

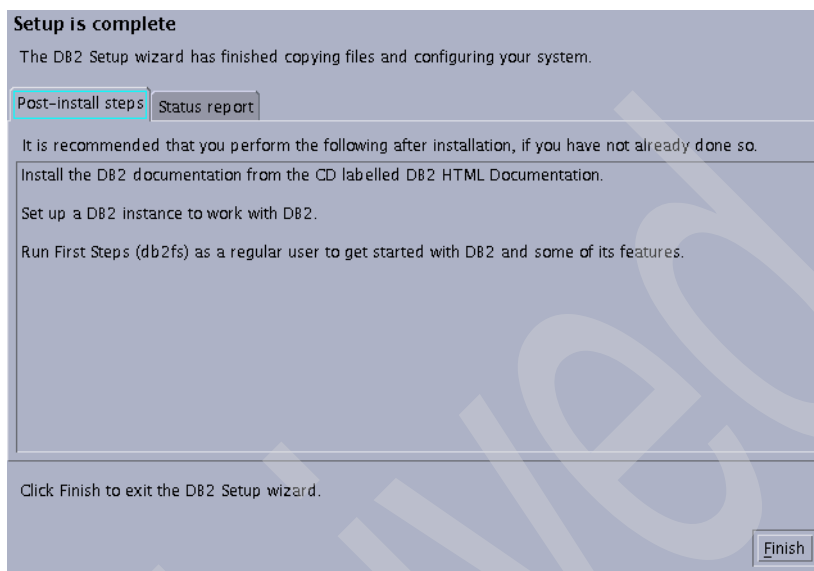


Figure 5-11 Finishing DB2 wizard installation: Post-install steps

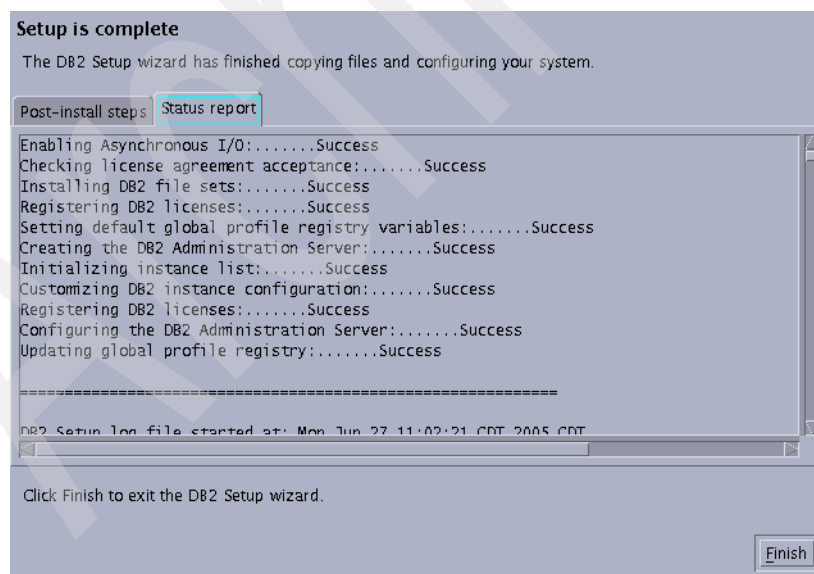


Figure 5-12 Finishing DB2 wizard installation: Status report

Now, your DB2 UDB is installed.

Installing the DB2 UDB client on AIX 5L

This section covers the DB2 V8.1 client installation on an AIX 5L environment. Perform the following steps to install DB2 UDB on the /usr file system:

1. Before the client installation, first create the file system /usr/opt/db2_08_01 with 200 MB to store the DB2 bin files:

```
crfs -v jfs -g rootvg -m'/usr/opt/db2_08_01' -p'rw' -a size='200M' -a frag='4096' -a nbpi='4096' -a ag='8' -A'yes'
```

Note: Creating a file system for the DB2 installation can ease your DB2 maintenance tasks.

2. Mount the cdrom on your AIX 5L machine:

```
mount -v cdrfs -r /dev/cd0 /cdrom
```

3. Create a temporary file system on the local machine with 750 MB to store the installation binaries files:

```
smitty crjfsstd  
or  
crfs -v jfs -g rootvg -m'/source_dir' -p'rw' -a size='750M' -a frag='4096' -a nbpi='4096' -a ag='8'
```

4. Mount the new file system:

```
mount /source_dir
```

5. Go to /source_dir directory:

```
cd /source_dir
```

6. Uncompress the binary file:

```
zcat /cdrom/DB2ESE.V812/ese.sbcsaix1.tar.Z | tar -xf -
```

7. Run the **db2setup** command to start the install process:

```
/source_dir/ese.sbcsaix1/db2setup
```

8. Figure 5-13 on page 230 shows the first window of the DB2 installation. Select **Install Products** from the right menu initiate the install process of DB2 UDB V8.1.

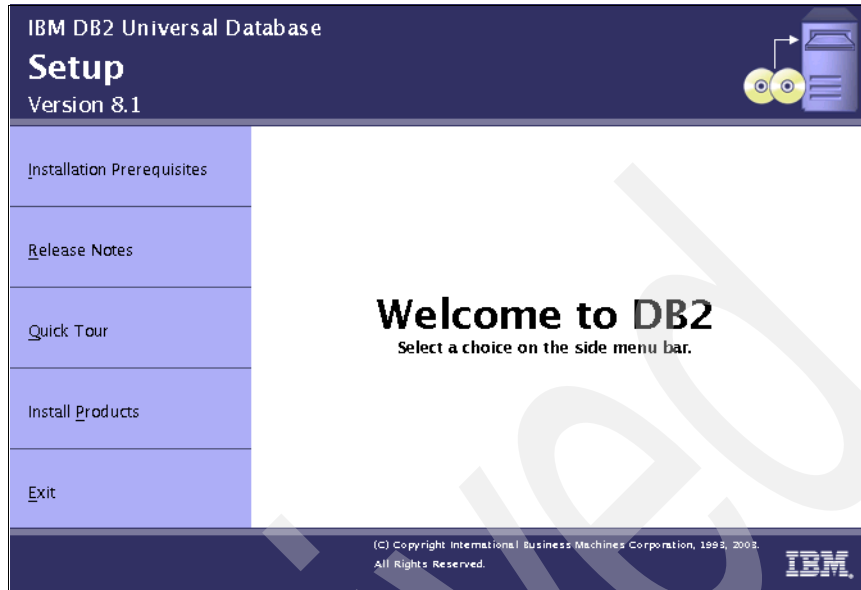


Figure 5-13 Welcome window

9. In Figure 5-14, select **DB2 Administration Client** and click **Next** to continue.

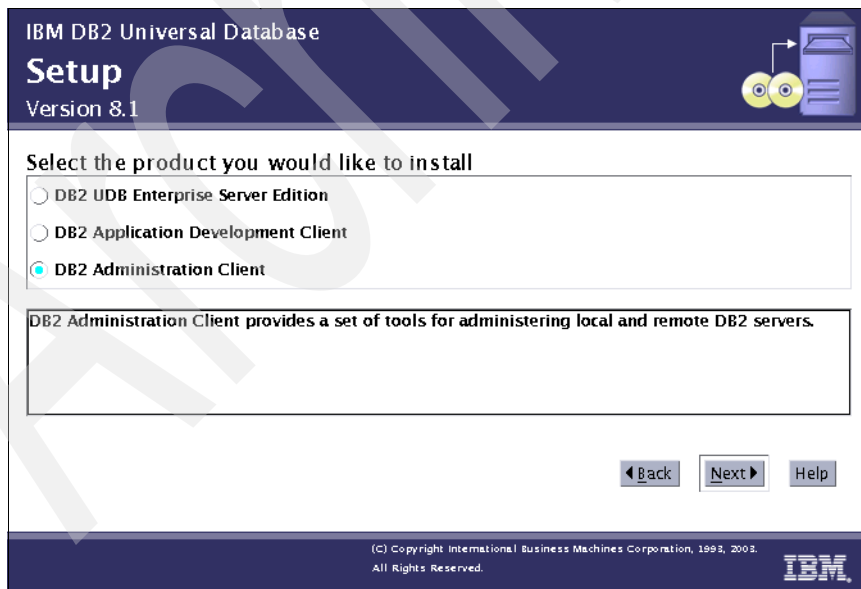


Figure 5-14 Selecting the product to install

10. Click **Next** to continue, as shown in Figure 5-15.

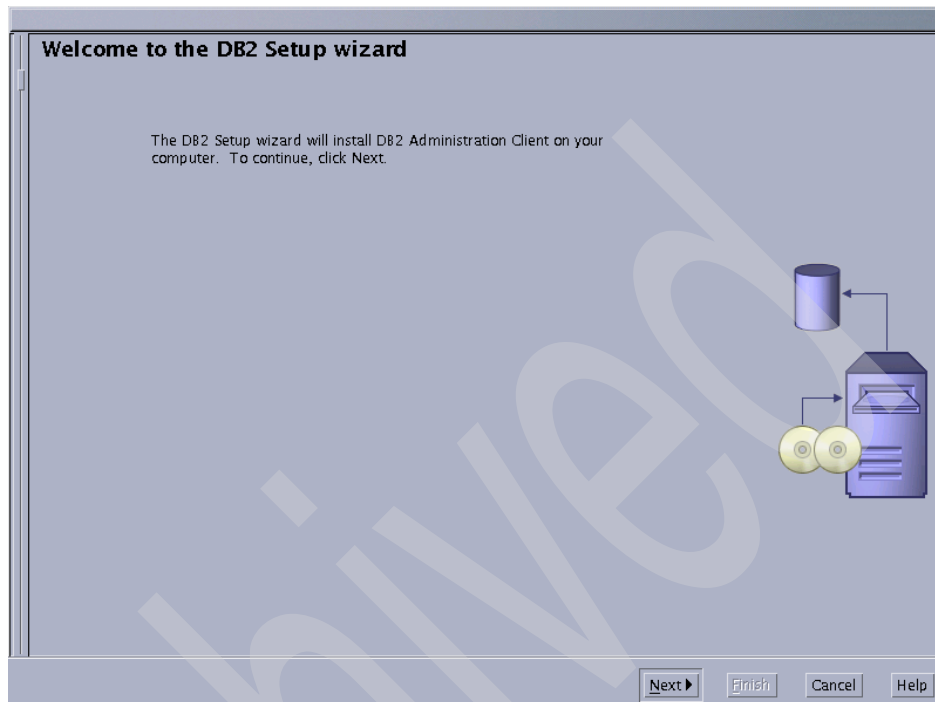


Figure 5-15 Welcome DB2 Setup wizard installation window

11. Read the software license agreement, select **Accept**, and click **Next**, as shown in Figure 5-16.



Figure 5-16 Software License Agreement window

12. Select the **Typical** installation and click **Next**, as shown in Figure 5-17.



Figure 5-17 Selecting the installation type

13. Select **Defer this task until after installation is complete** and click **Next**, as shown in Figure 5-18.

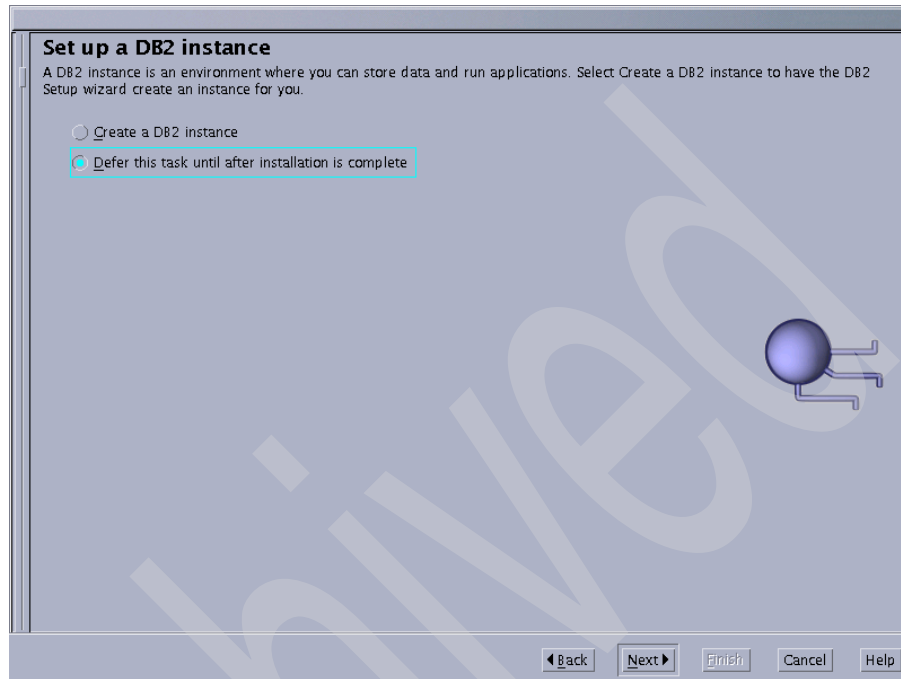


Figure 5-18 Setting up a DB2 instance

14. Figure 5-19 shows the DB2 wizard configuration summary. Review the configuration and click **Finish** to begin copying files.

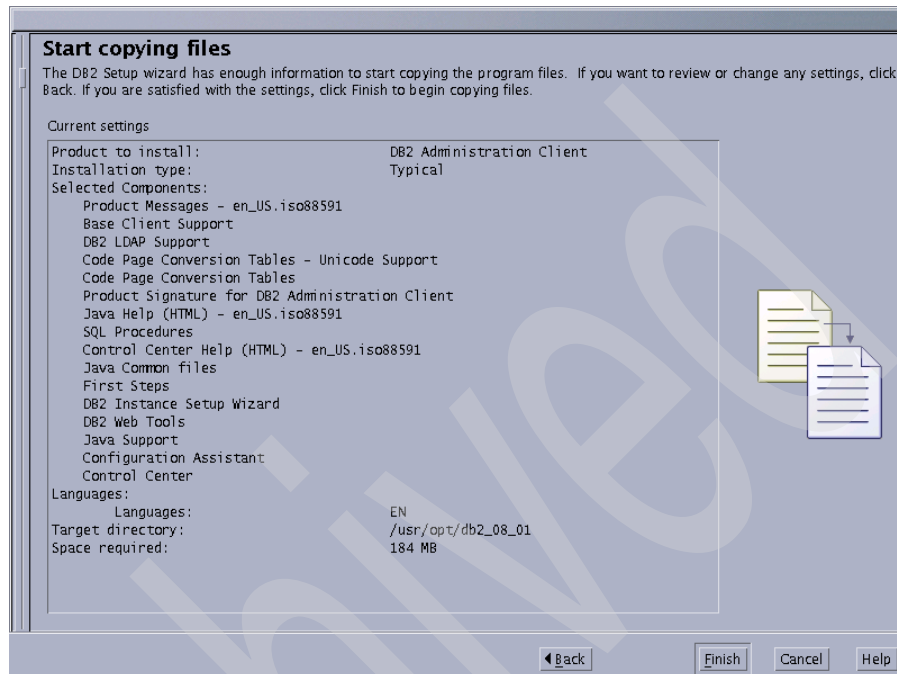


Figure 5-19 DB2 wizard configuration summary

15. Figure 5-20 and Figure 5-21 show the end of the installation process. Click **Finish** to terminate the DB2 wizard installation.

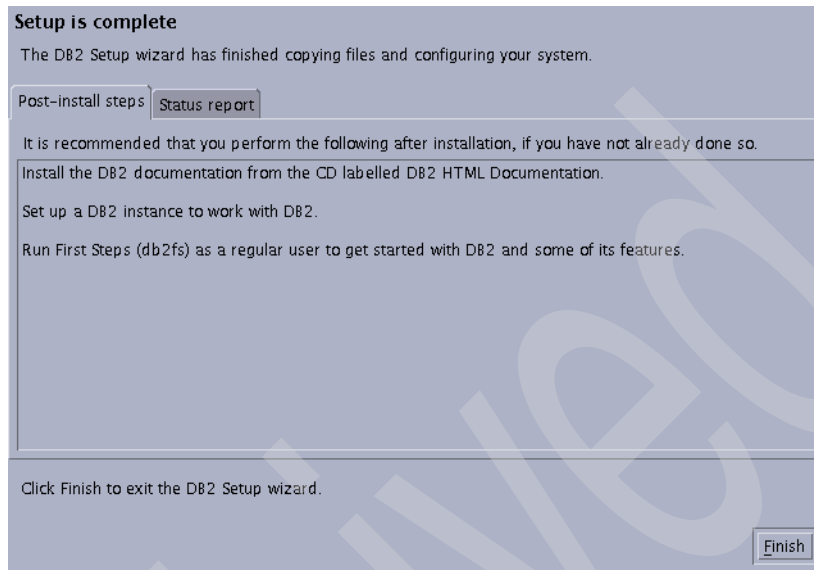


Figure 5-20 Finishing DB2 wizard installation: Post-install steps

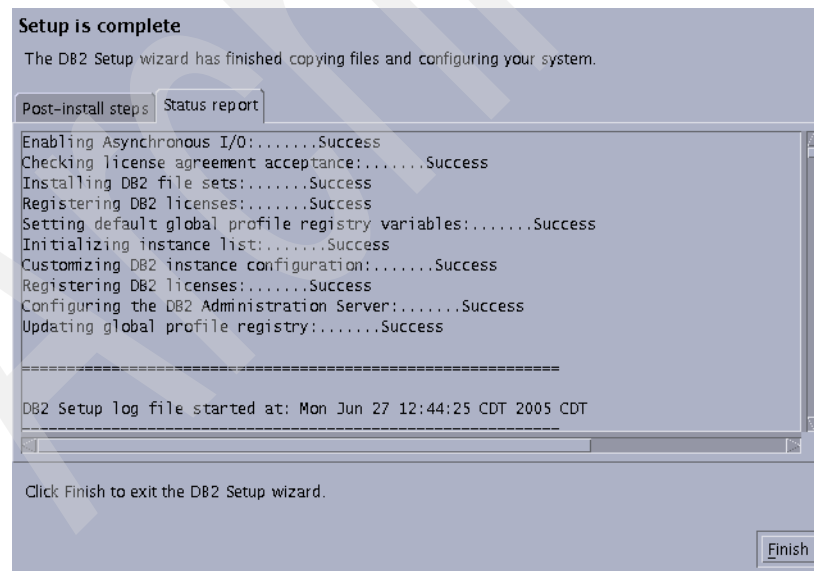


Figure 5-21 Finishing DB2 wizard installation: Status report

Your DB2 UDB client has been installed.

5.2 LDAP server installation

This section covers the Lightweight Directory Access Protocol (LDAP) installation and configuration steps on Microsoft Windows 2000 Server.

5.2.1 System requirements

For the full server, IBM Tivoli Directory Server requires about 512 MB of memory and 2 GB of disk space. The disk space might increase based on the number of entries and the size of each entry for your installation. It is also necessary install DB2 before installing the IBM Tivoli Directory Server.

5.2.2 Basic LDAP concepts

LDAP is used to store relatively static information. The phrase *write once, read many times* describes the best use of LDAP. LDAP is structured as a directory optimized for lookups. Its tree structure is useful for conceptualizing organizational structures.

Each entry in the LDAP tree is composed of one or more object classes, as shown in Figure 5-22 on page 238. Every object class has attributes stored in name-value pairs.

An example of an LDAP attribute is `cn=root`. An attribute also specifies the types of operations that it supports.

The structure of this tree is based on the LDAP schema.

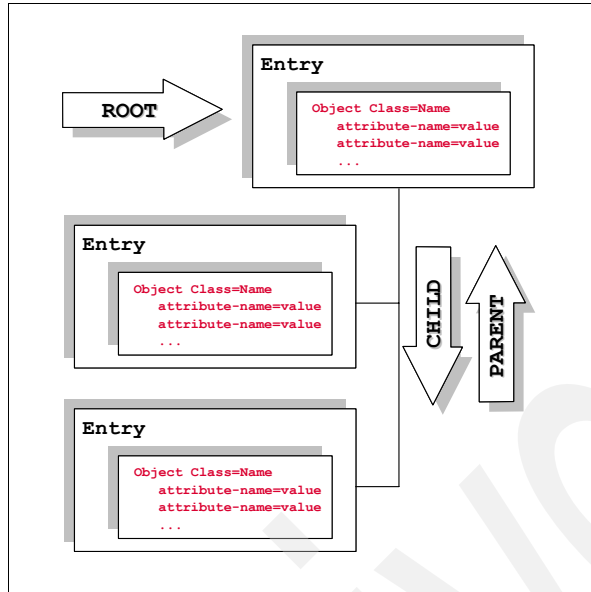


Figure 5-22 General structure of a LDAP tree

5.2.3 Installing IBM Tivoli Directory Server V5.2 on a Windows server

This section provides the steps to install IBM Tivoli Directory Server on a Windows 2000 Server called nice.

Perform the following steps to install IBM Tivoli Directory Server Version 5.2:

1. Make sure that you already have IBM DB2 UDB installed on the machine.
2. Put the installation CD in the CD-ROM drive, go to the ismp directory and run **setup.exe**.
3. Select the language, as shown in Figure 5-23, and click **OK** to continue.



Figure 5-23 InstallShield language selection

4. Click **Next** in the Welcome window shown in Figure 5-24.



Figure 5-24 Welcome window

5. Read and accept the license agreement shown in Figure 5-25 and click **Next** to continue.

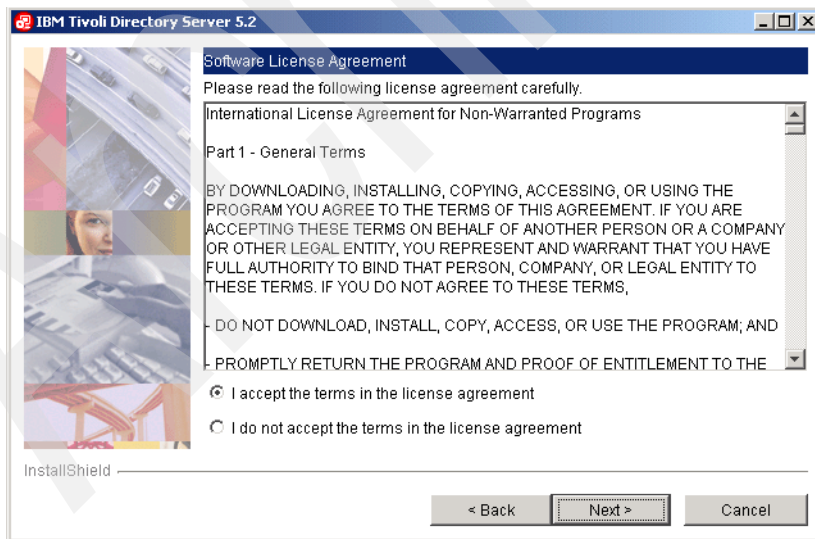


Figure 5-25 License Agreement window

- Click **Next** in Figure 5-26 to continue. This window shows the products that are installed.

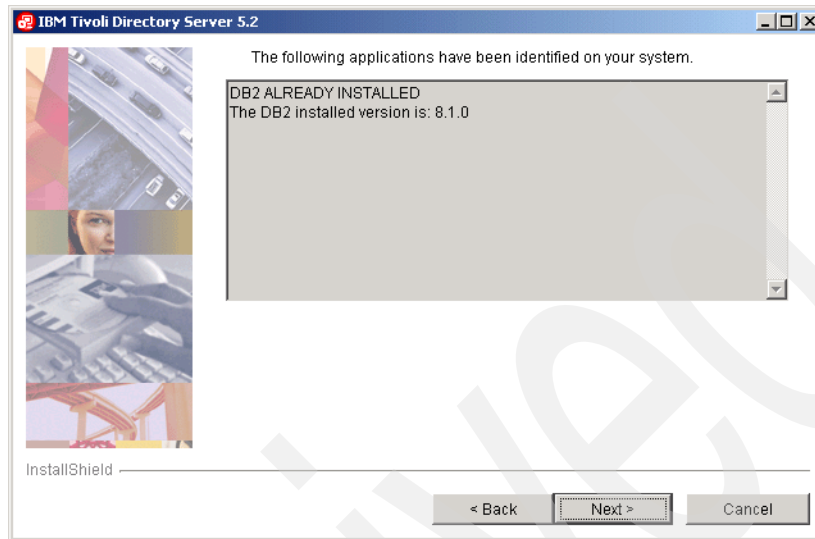


Figure 5-26 Products detected

- Choose the destination directory, as shown in Figure 5-27, and click **Next** to continue.

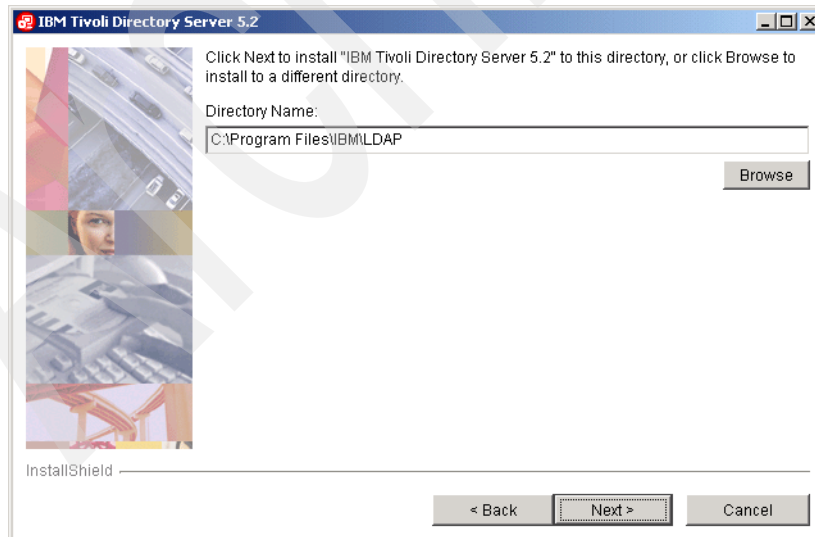


Figure 5-27 Destination directory

8. Select the language, as shown in Figure 5-28, and click **Next** to continue.

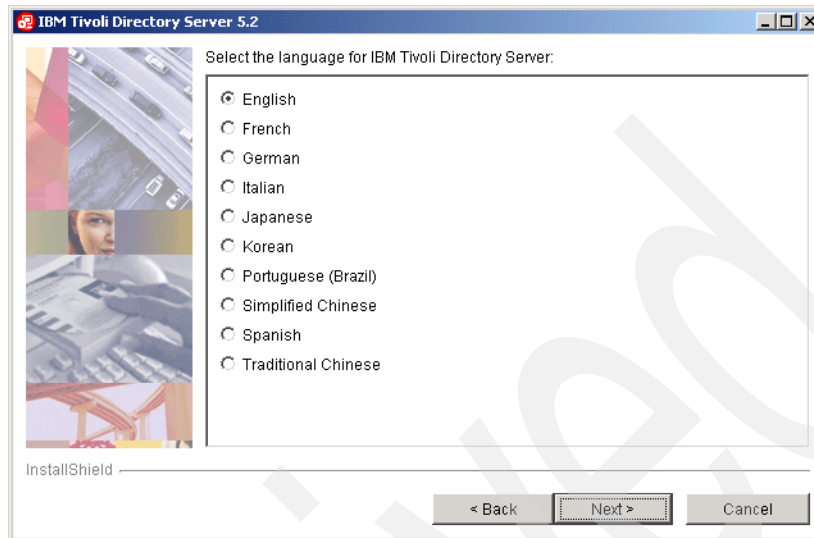


Figure 5-28 Installation language

9. Choose the products that you want to install, as shown in Figure 5-29, and click **Next** to continue. For our environment, we choose **Client SDK 5.2**, **Server 5.2**, and **GSKit**.

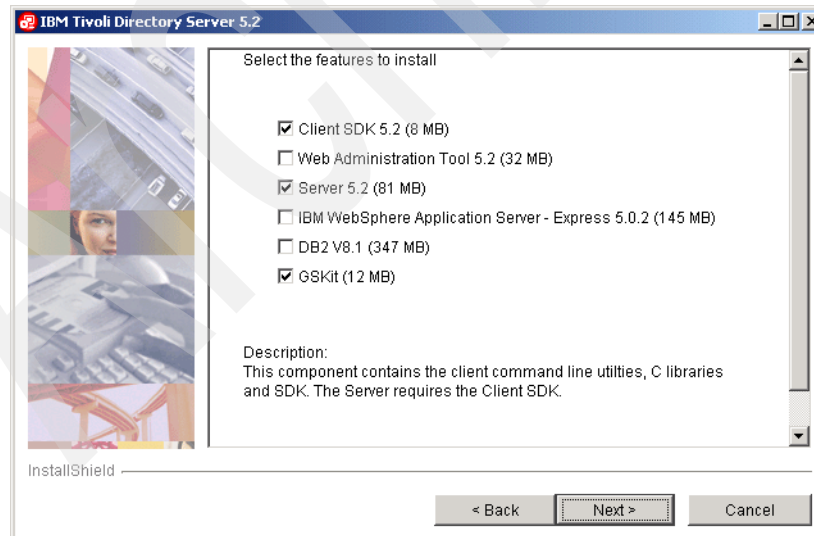


Figure 5-29 Products to install

10. Figure 5-30 shows the actions that will be performed by the installation. Click **Next** to continue.

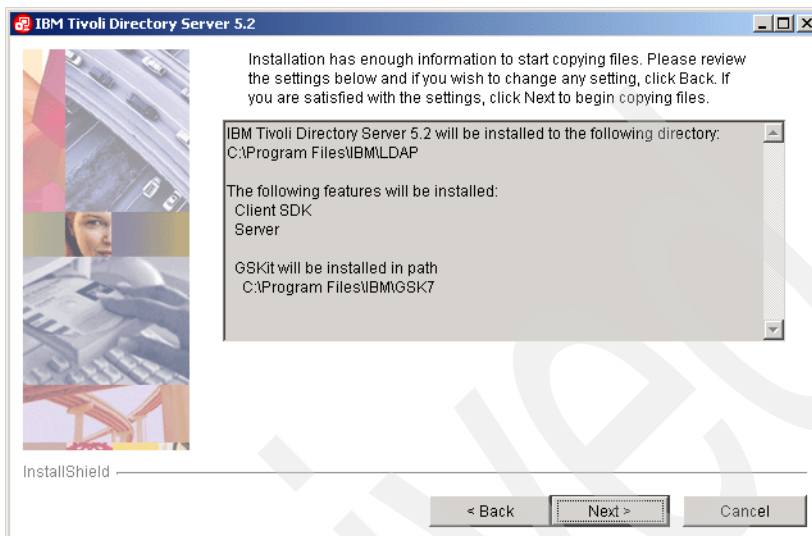


Figure 5-30 Installation actions

11. Figure 5-31 shows the GSKit installation progress.

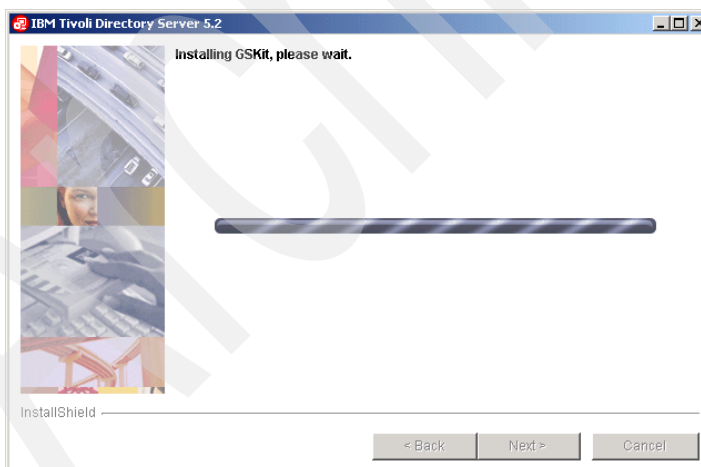


Figure 5-31 GSKit installation window

12. Figure 5-32 shows the IBM Tivoli Directory Server installation progress.

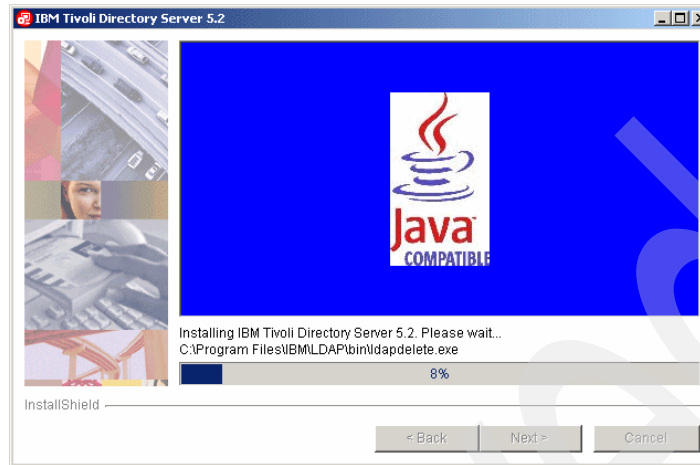


Figure 5-32 IBM Tivoli Directory Server installation window

13. Figure 5-33 shows the *Client Readme*. Click **Next** to continue.

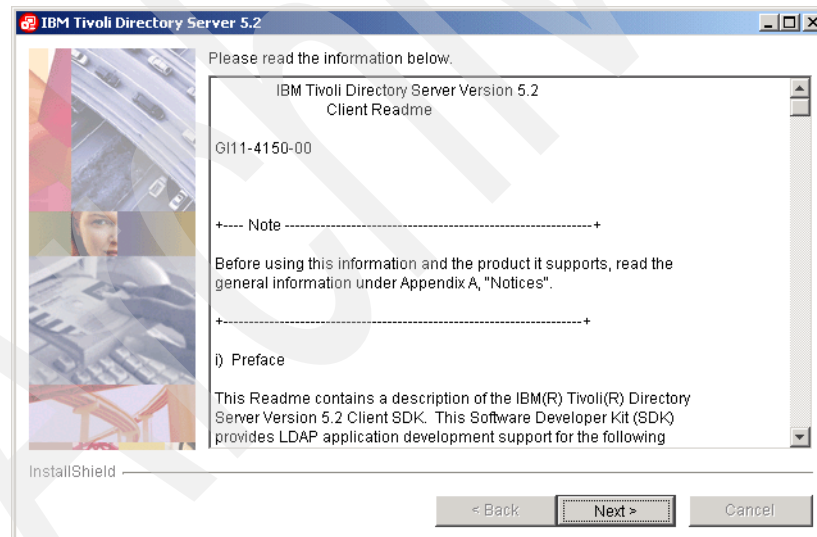


Figure 5-33 Client Readme window

14. Figure 5-34 shows the *Server Readme* file. Click **Next** to continue.

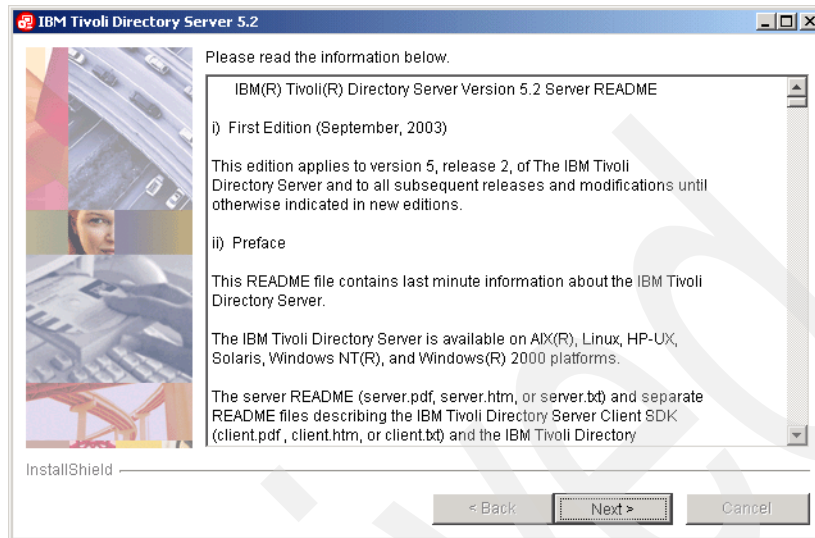


Figure 5-34 *Server Readme window*

15. Figure 5-35 shows the end of the installation. Click **Finish** to terminate the installation process.

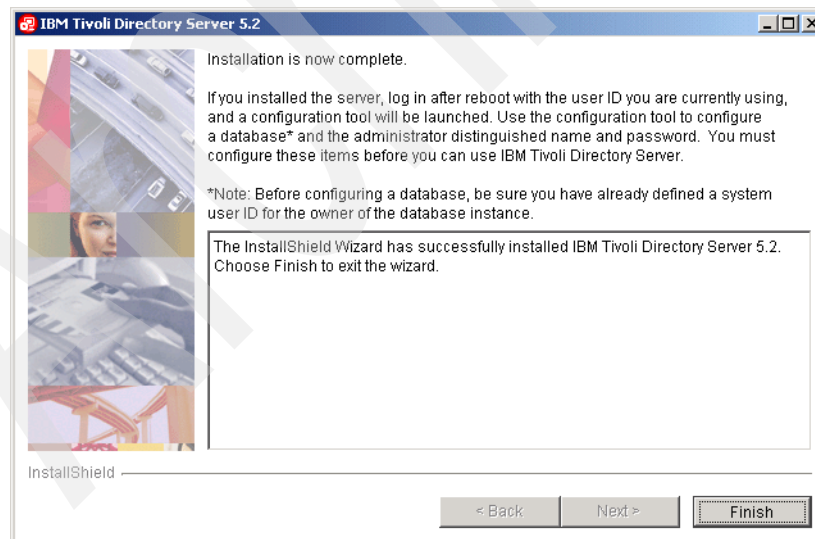


Figure 5-35 *Installation complete window*

16. You need to reboot after the installation of IBM Tivoli Directory Server on Windows, as shown in Figure 5-36. Select **Yes** and click **Next** to reboot your system and finish the installation.

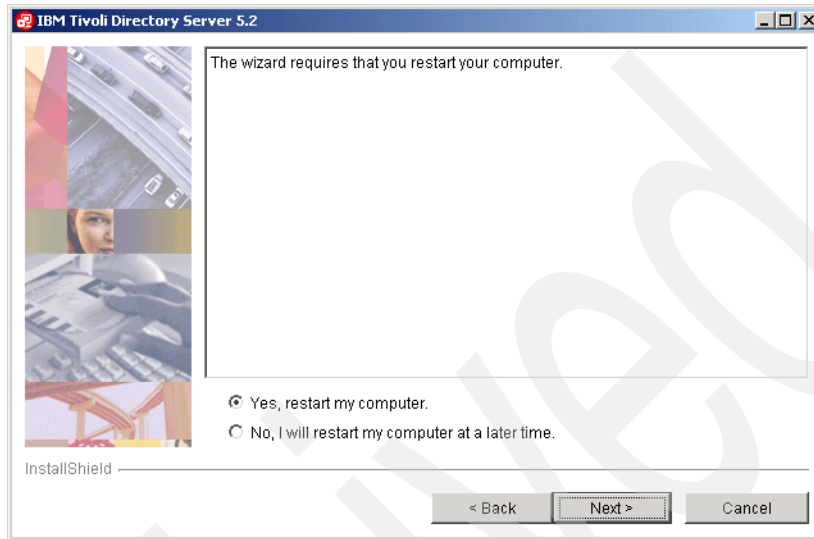


Figure 5-36 Reboot window

After rebooting, the LDAP configuration window opens. We cover the configuration next.

5.2.4 Configuration steps after IBM Tivoli Directory Server installation

The IBM Tivoli Directory Server Configuration Tool window, as shown in Figure 5-37 on page 246, opens automatically when you reboot the system after the IBM Tivoli Directory Server installation.

In this section, we use the configuration tool to create the LDAP administrator and the database on DB2 and to populate the database with users' information.

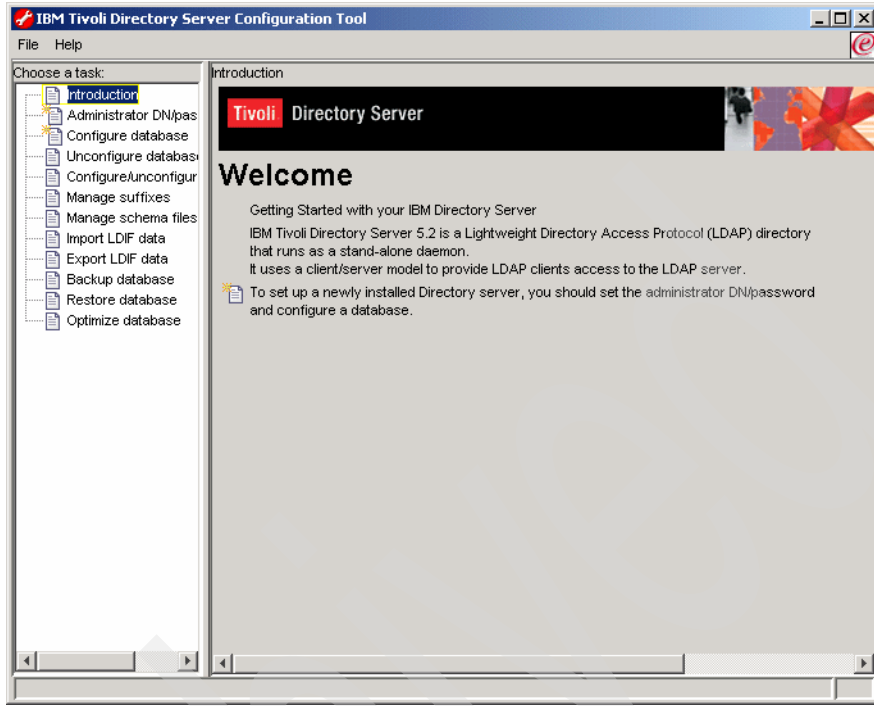


Figure 5-37 Tivoli Directory Server Configuration Tool window

Creating the LDAP administrator

The first thing to do in the LDAP configuration tool is to create the administrator. Here, we create the administrator `cn=root,dc=ibm,dc=com`, performing the following steps:

1. In the left menu of Figure 5-37, select **Administrator DN/password**.
2. Figure 5-38 on page 247 shows the Administration DN/password window. Enter the Administrator `cn=root,dc=abbc,dc=com`, set a password for it, and click **OK**.

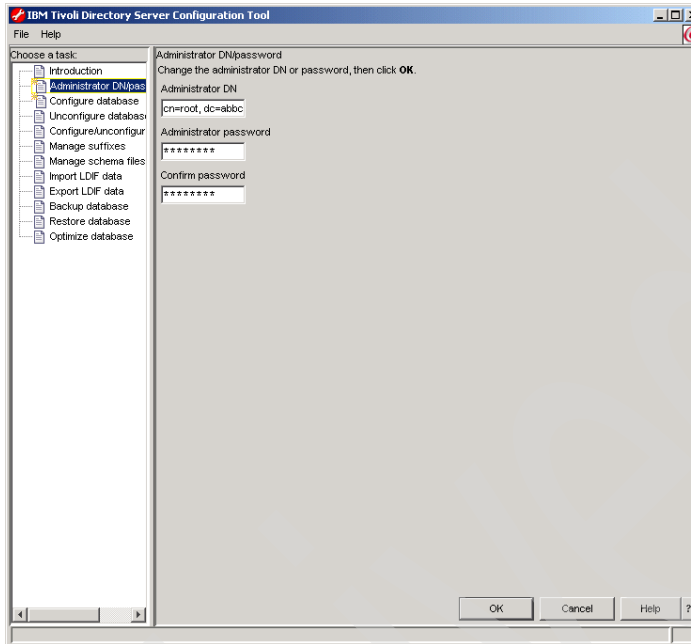


Figure 5-38 Creating an LDAP administrator

3. Figure 5-39 shows the confirmation of the administrator creation. Click **OK**.



Figure 5-39 Confirmation window

4. You return to the Welcome page, as shown in Figure 5-37 on page 246. Next, we create the suffix `dc=abbc,dc=com`. To do that, select **Manage suffixes** from the left menu.
5. Figure 5-40 on page 248 shows the Manage suffixes window. Enter the new suffix `dc=abbc,dc=com`. Click **Add** and then **Close**.

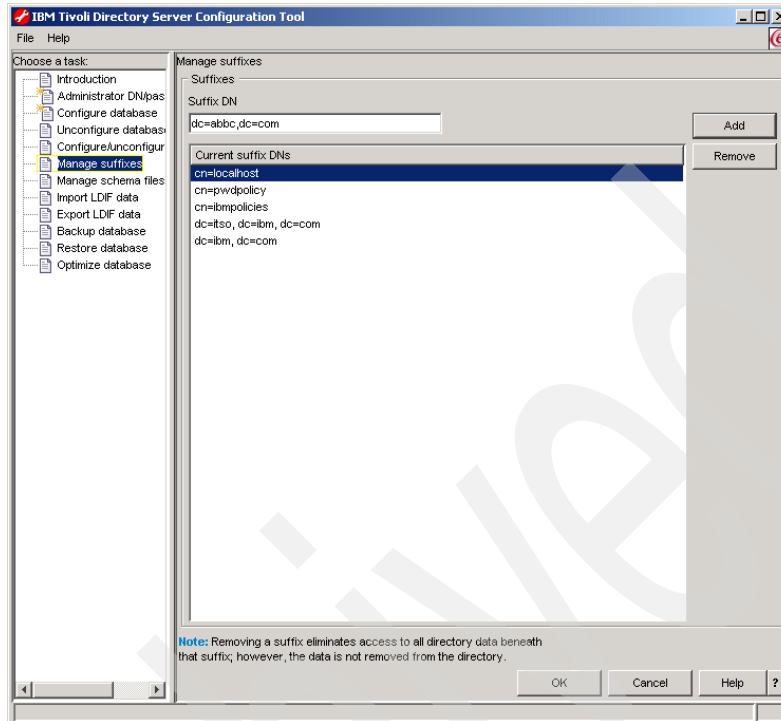


Figure 5-40 Manage suffixes window

You will return to the Welcome window, shown in the Figure 5-37 on page 246. Next, you need to create the database, as described in the next section.

Creating the LDAP database

After creating the LDAP administrator, you need to create a database to store the LDAP data. To do this, perform the following steps:

1. In the Welcome window, shown in Figure 5-37 on page 246, select **Configure database**.

2. In Figure 5-41, select **Create a new database** and click **Next** to continue.

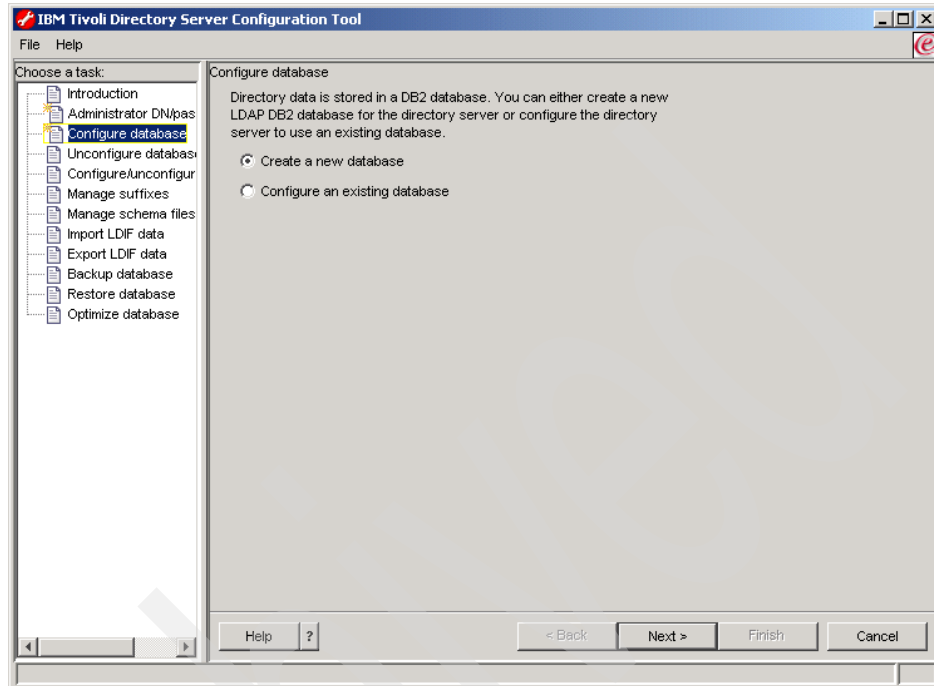


Figure 5-41 Configure database window

3. In Figure 5-42, enter the DB2 user and password to create the database and click **Next** to continue.

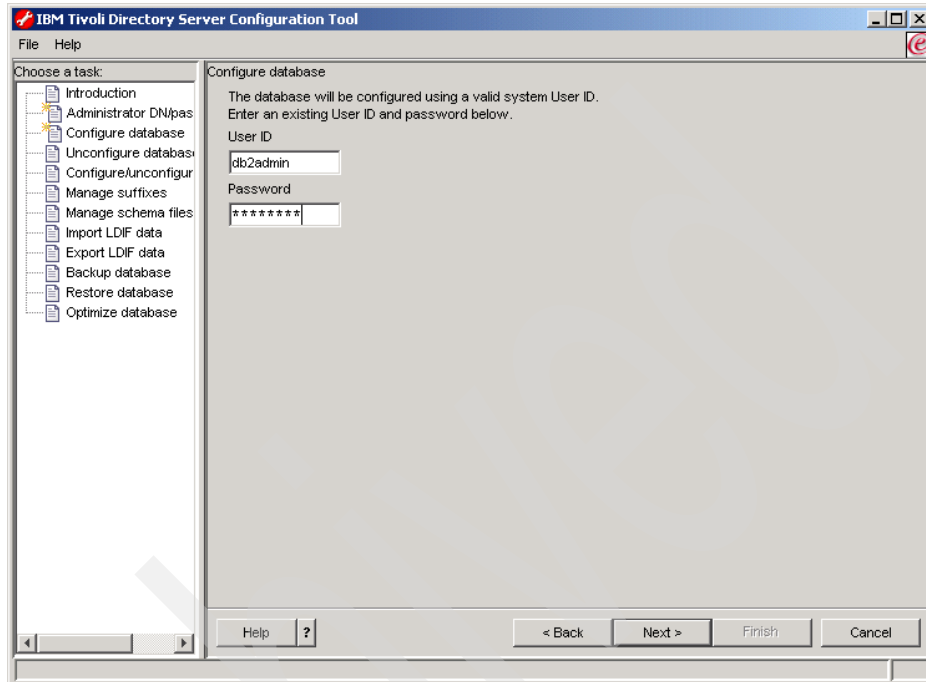


Figure 5-42 DB2 user

4. In Figure 5-43, enter the database name that will be created and click **Next** to continue. In our environment, the database name is `itcmldap`.

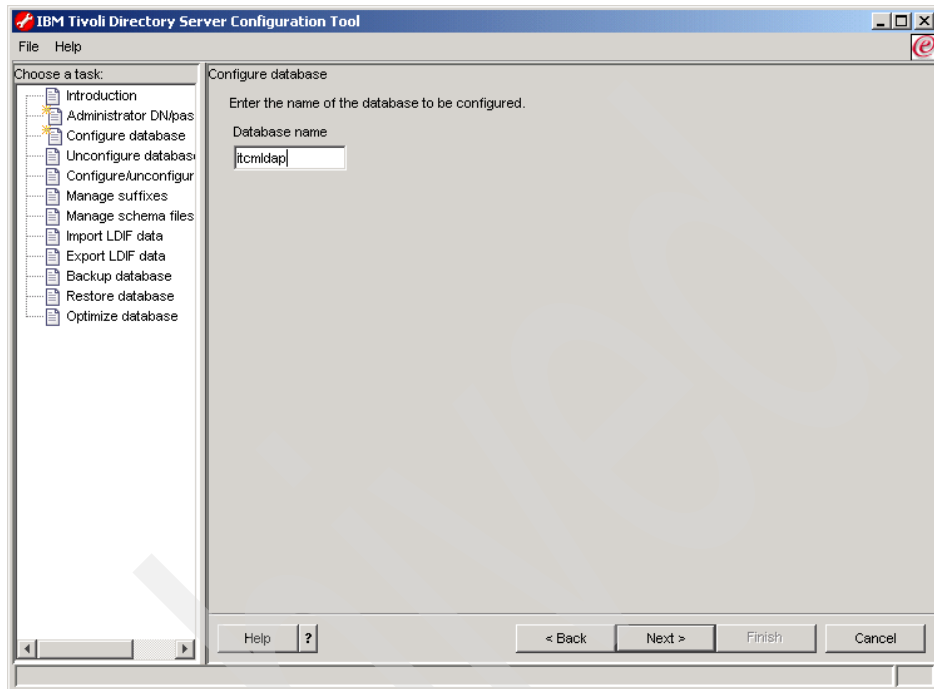


Figure 5-43 Database name

5. In Figure 5-44, select the type of DB2 database that you want to create and click **Next** to continue. In our environment, we create a DB2 database.

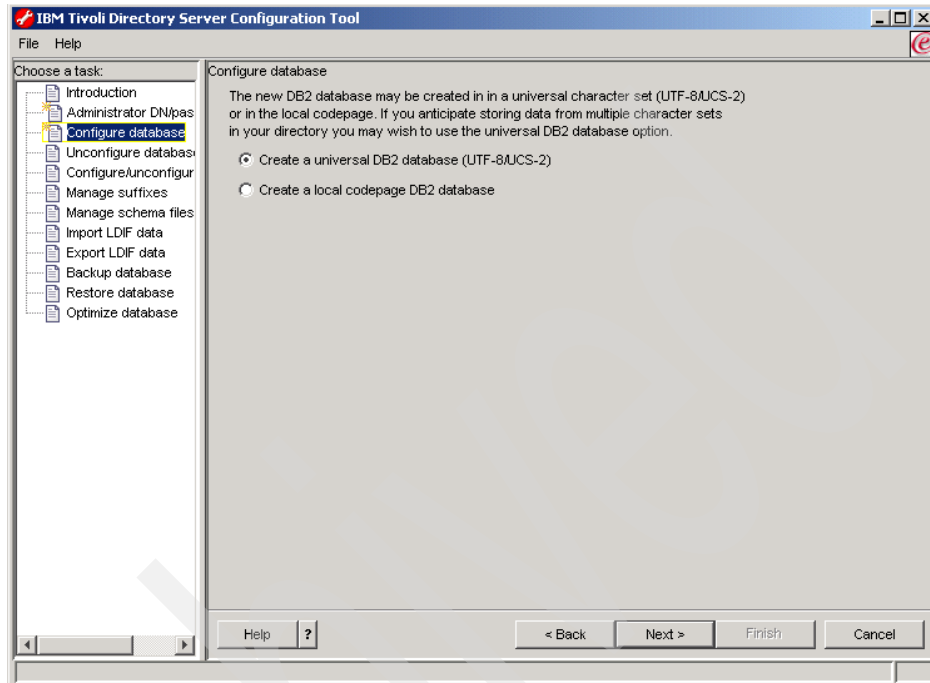


Figure 5-44 Type of DB2 database

- In Figure 5-45, select the drive where the database will be created and click **Next** to continue.

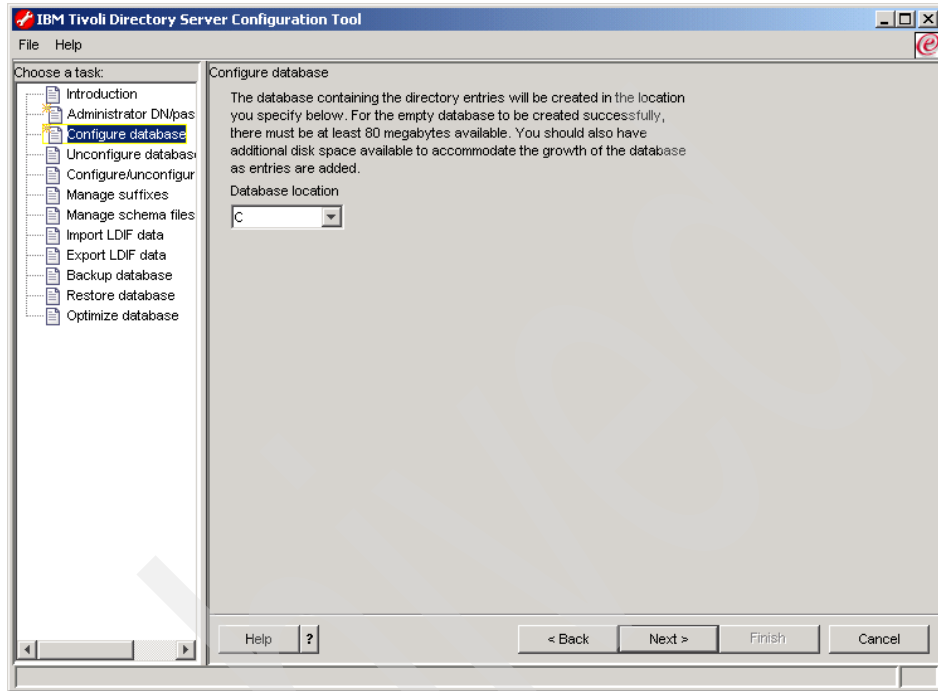


Figure 5-45 Selecting the drive of the database location

7. Figure 5-46 shows the actions that will be performed. Click **Finish** to create the database.

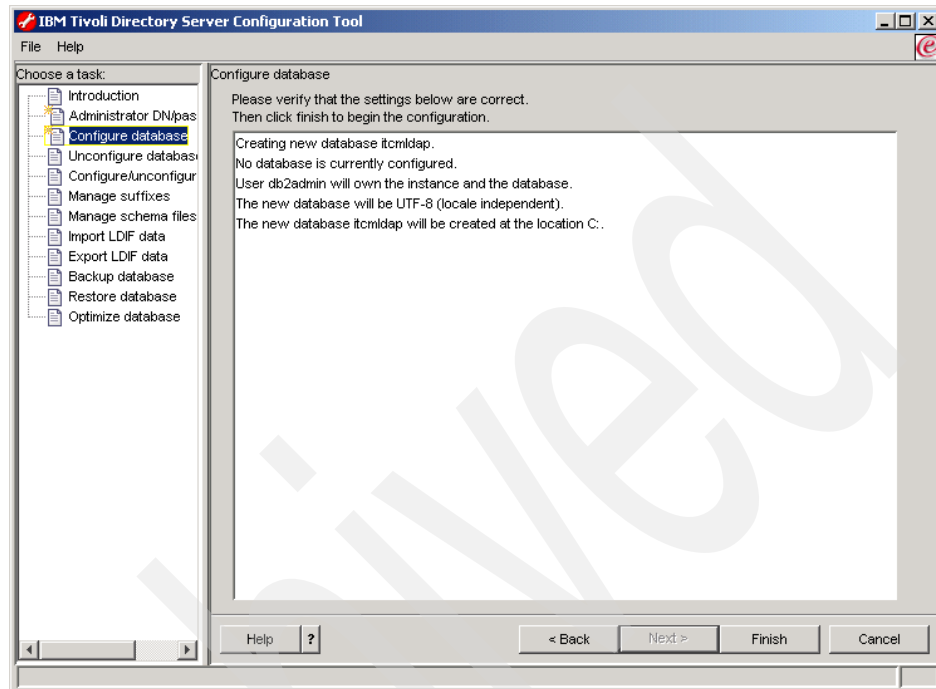


Figure 5-46 LDAP: Actions to be done

8. Figure 5-47 on page 255 shows the status of the database creation. Click **Close** to close the window and return to the Welcome page.

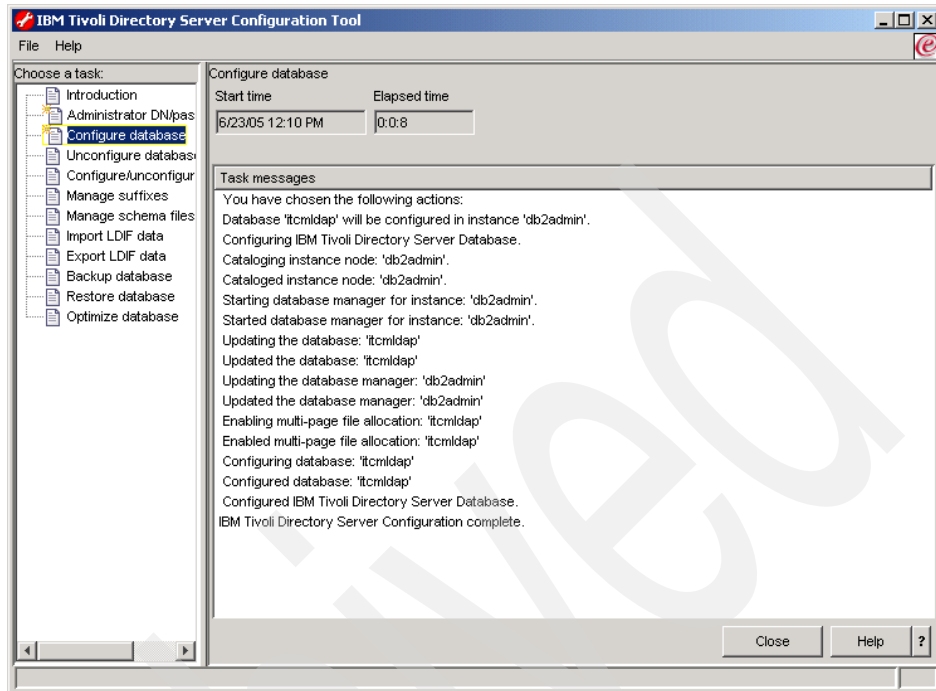


Figure 5-47 LDAP: Status of the database creation

Next, we populate the LDAP server with user information, as described in the following section.

Populating the LDAP database

This section shows the last step that must be done in the IBM Tivoli Directory Server Configuration Tool.

The easiest way to populate the LDAP database is by importing an LDIF file. Example 5-1 shows the LDIF file used to populate the LDAP database described in this book.

Example 5-1 The sample.ldif file

```
# IBM Directory Server sample LDIF file
#
# The suffixes "dc=abbc, dc=com" should be defined before attempting
# to load this data.

version: 1

dn: dc=abbc, dc=com
```

```
objectclass: top
objectclass: domain
dc:abbc
```

```
dn: ou=people, dc=abbc, dc=com
ou: people
objectclass: organizationalUnit
```

```
dn: cn=Vasfi Gucer, ou=people, dc=abbc, dc=com
objectclass: ePerson
objectclass: inetOrgPerson
cn:Vasfi Gucer
sn:Gucer
givenName:Vasfi
initials:VG
title:Technical Leader
uid:vgucer
o:abbc
ou:abbcus
userpassword:password
mail:vgucer@abbc.com
```

```
dn: cn=Sanver Ceylan, ou=people, dc=abbc, dc=com
objectclass: ePerson
objectclass: inetOrgPerson
cn:Sanver Ceylan
sn:Ceylan
givenName:Sanver
initials:SC
title:Technical Leader
uid:sceylan
o:abbc
ou:abbcus
userpassword:password
mail:sceylan@abbc.com
```

```
dn: cn=Luciano Peetz, ou=people, dc=abbc, dc=com
objectclass: ePerson
objectclass: inetOrgPerson
cn:Luciano Peetz
sn:Peetz
givenName:Luciano
initials:LP
title:IT Specialist
uid:lpeetz
o:abbc
ou:abbcbr
userpassword:password
mail:lpeetz@abbc.com
```

After creating the LDIF file, perform the following steps to import it:

1. In left menu of the Welcome page, shown in Figure 5-37 on page 246, select the **Import LDIF data**.
2. To import the LDIF sample file, browse for the LDIF sample file, select **Standard import**, and click **Import**, as shown in Figure 5-48.

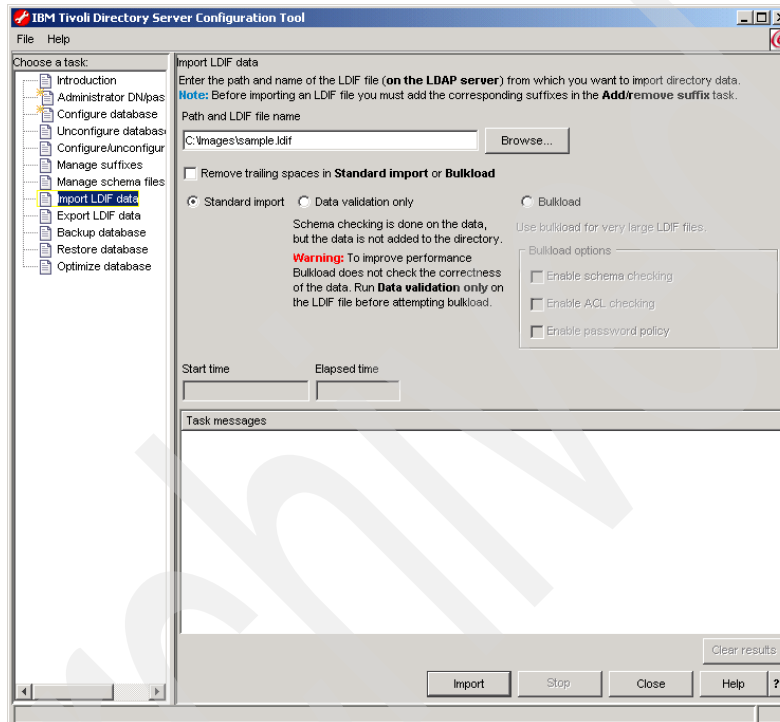


Figure 5-48 Importing LDIF data

3. Figure 5-49 on page 258 shows the status of the import operation. Click **Clear results** and **Close** to return to the Welcome page.

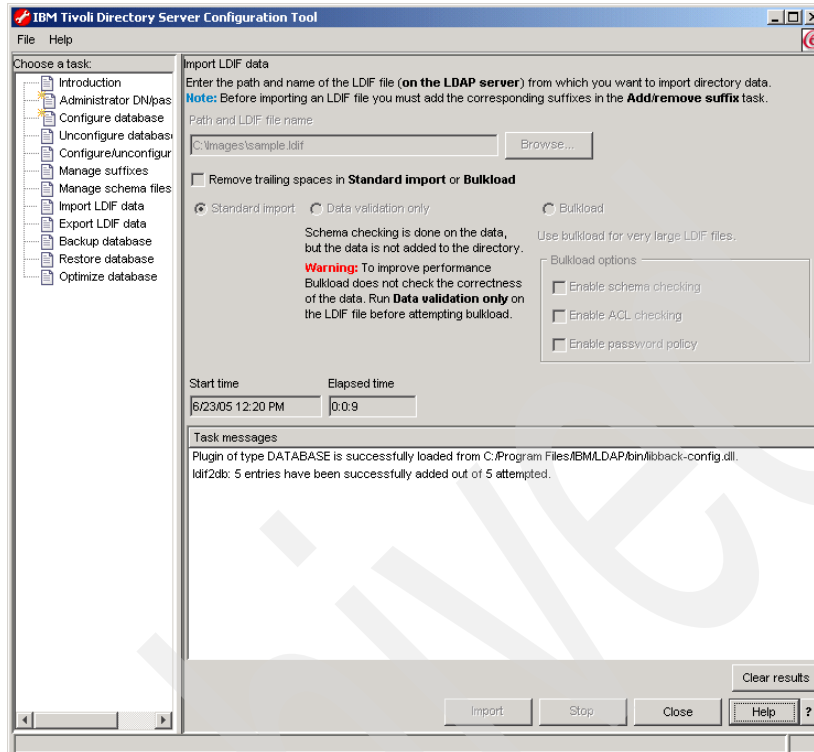


Figure 5-49 Importing LDIF data status window

Now, you can close the IBM Tivoli Directory Server Configuration Tool and start IBM Tivoli Directory Server Version 5.2, as shown in Figure 5-50.

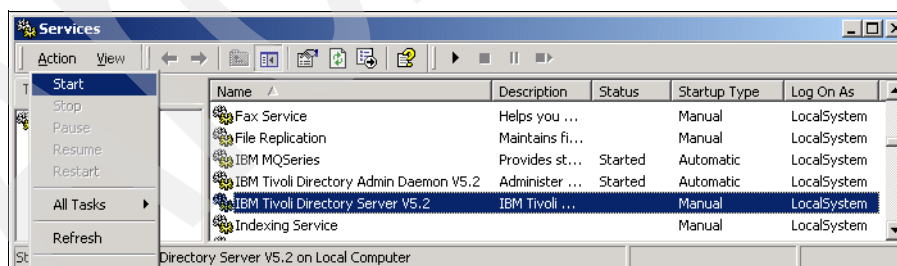


Figure 5-50 Starting IBM Tivoli Directory Server Version 5.2

Your LDAP server is installed and configured.

Note: You need to stop IBM Tivoli Directory Server before opening the Configuration Tool. You cannot open the IBM Tivoli Directory Server Configuration Tool while IBM Tivoli Directory Server is active.

5.3 IBM WebSphere Application Server installation

This section describes the prerequisites and the steps to install WebSphere Application Server Version 5.1 on an AIX 5L machine.

5.3.1 Before installing WebSphere Application Server V5.1

You must complete the following steps before installing WebSphere Application Server V5.1 on your AIX 5L machine. For more information about the prerequisites, go to the following Web site:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1/index.jsp>

Perform the following steps:

1. Install the following AIX 5L filesets:
 - X11.fnt.ucs.ttf
 - X11.fnt.ucs.ttf_KR
 - X11.fnt.ucs.ttf_TW
 - X11.fnt.ucs.ttf_CN
2. Create the file system /usr/WebSphere to store the WebSphere bin files:

```
crfs -v jfs -g rootvg -m'/usr/WebSphere' -p'rw' -a size='700M' -a frag='4096' -a nbpi='4096' -a ag='8' -A'yes'
```
3. Create the file system /usr/IBMHttpServer to store the HTTP Server bin files:

```
crfs -v jfs -g rootvg -m'/usr/IBMHttpServer' -p'rw' -a size='30M' -a frag='4096' -a nbpi='4096' -a ag='8' -A'yes'
```
4. Create the AIX 5L groups mqm and mqbrkrs, as shown in Example 5-2.

Example 5-2 Creating the mqm and mqbrkrs administrative groups

```
mkgroup -a mqm  
mkgroup -a mqbrkrs
```

5. Include the groups mqm and mqbrkrs to the root user.
6. Make sure that the port 9090 is not in use. In general, this port is used by the websm service on AIX 5L; disable this service to make it available.
7. Create the user mqm, as shown in Example 5-3 on page 260.

Example 5-3 Creating the mqm user

```
# mkuser pgrp='mqm' mqm
# passwd mqm
Changing password for "mqm"
mqm's New password:*****
Enter the new password again:*****
# pwdadm -c mqm
```

8. Include the group mqm to the mqm user.

Now, your environment is ready to start the WebSphere Application Server V5.1 installation.

5.3.2 Installing WebSphere Application Server V5.1

Perform the following steps to install WebSphere Application Server Version 5.1:

1. Mount the *WebSphere Base Installation CD* and run the `1aunchpad.sh` script.
2. The window shown in Figure 5-51 opens. Select **Install the product** to start the installation.

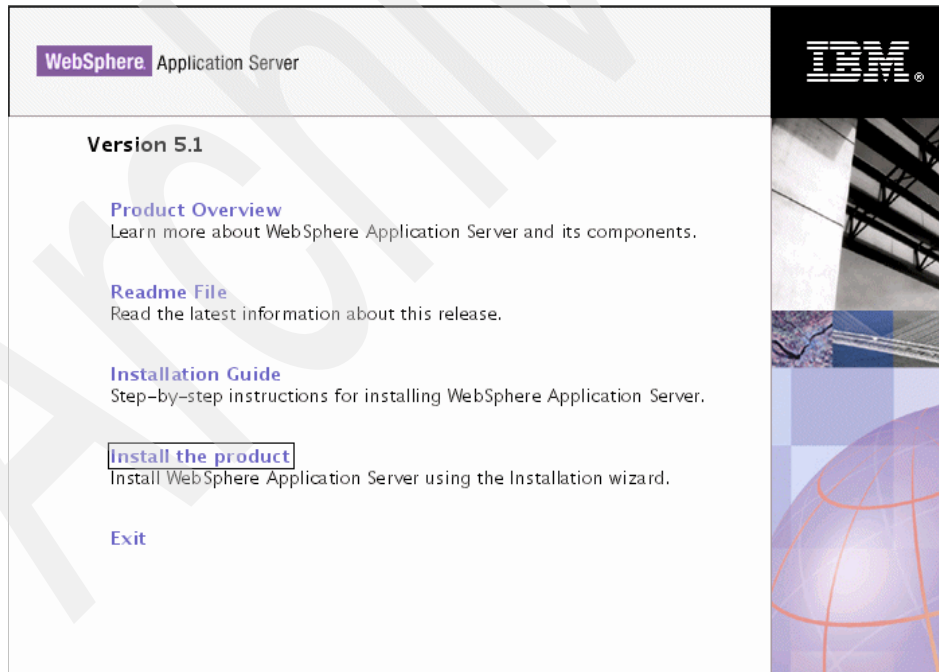


Figure 5-51 WebSphere Application Server V5.1: Installation menu

3. Click **Next** to continue, as shown in Figure 5-52.

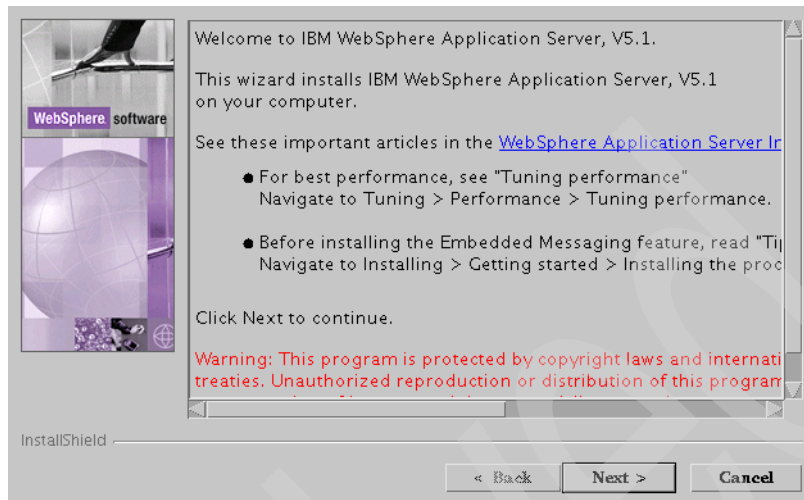


Figure 5-52 WebSphere Application Server V5.1: Welcome window

4. Read and accept the license agreement and click **Next** to continue, as shown in Figure 5-53.

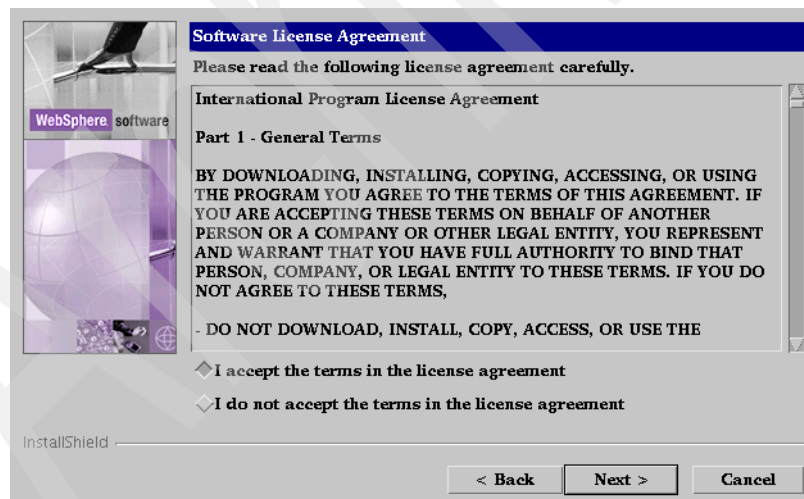


Figure 5-53 WebSphere Application Server V5.1: License Agreement

5. The window shown in Figure 5-54 opens only if you have any prerequisites missing. Fix any needed prerequisites and click **Next** to continue.

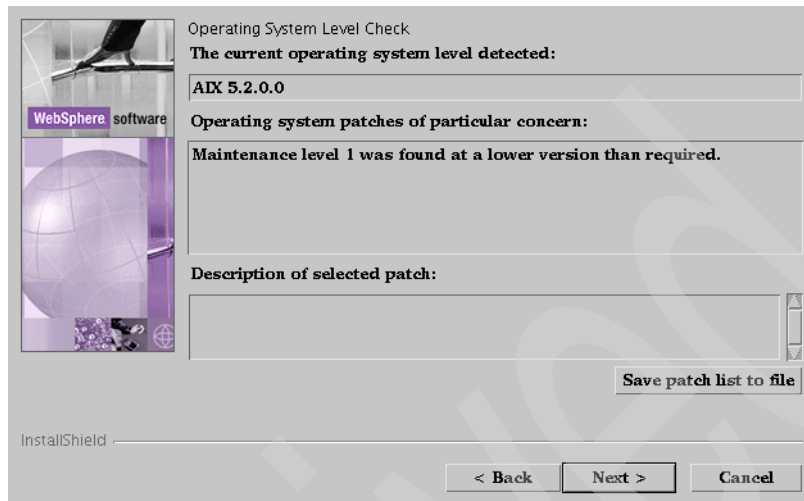


Figure 5-54 WebSphere Application Server V5.1: Prerequisites

6. Select the **Full** installation and click **Next** to continue, as shown in Figure 5-55.

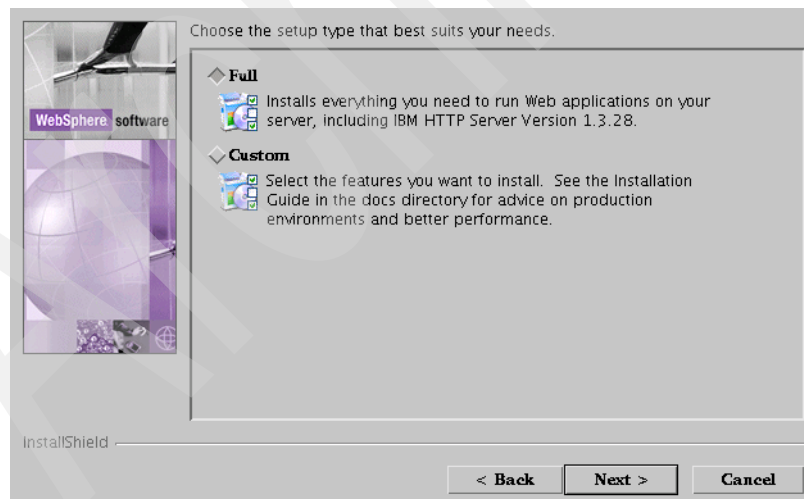


Figure 5-55 WebSphere Application Server V5.1: Full installation

7. In Figure 5-56 on page 263, enter the destination directory for the installation and click **Next** to continue.

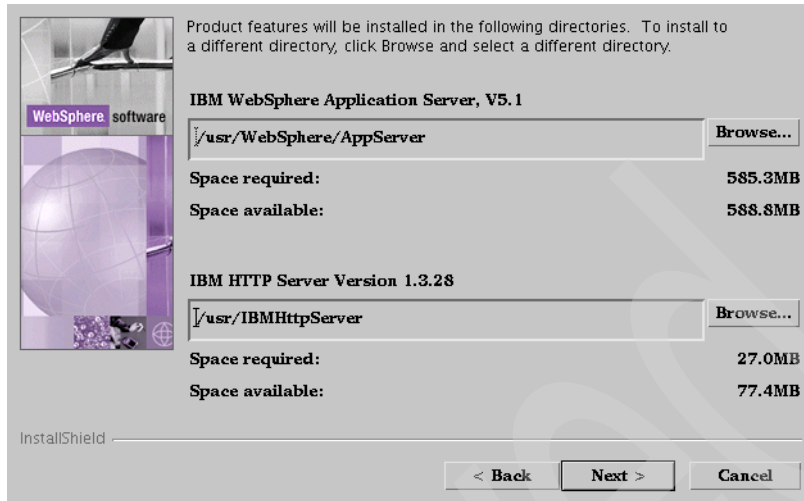


Figure 5-56 WebSphere Application Server V5.1: Destination directories

8. Figure 5-57 shows the node information. Click **Next** to continue.

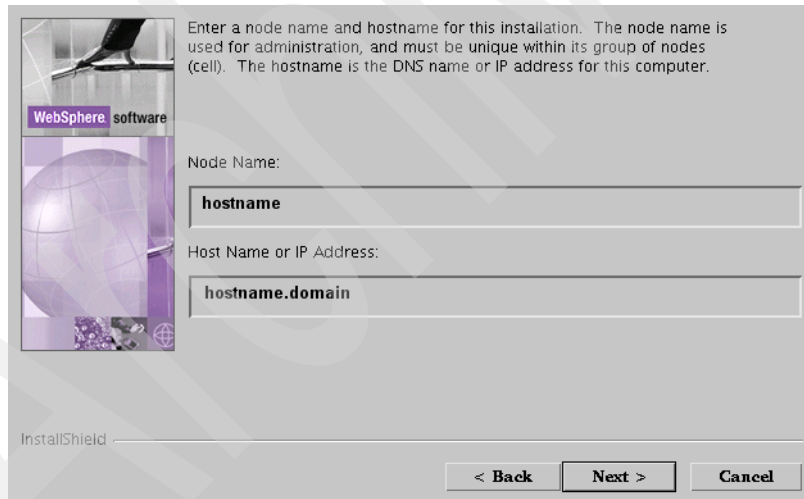


Figure 5-57 WebSphere Application Server V5.1: Node information

9. The window in Figure 5-58 shows the actions that will be performed during the installation. Click **Next** to continue.

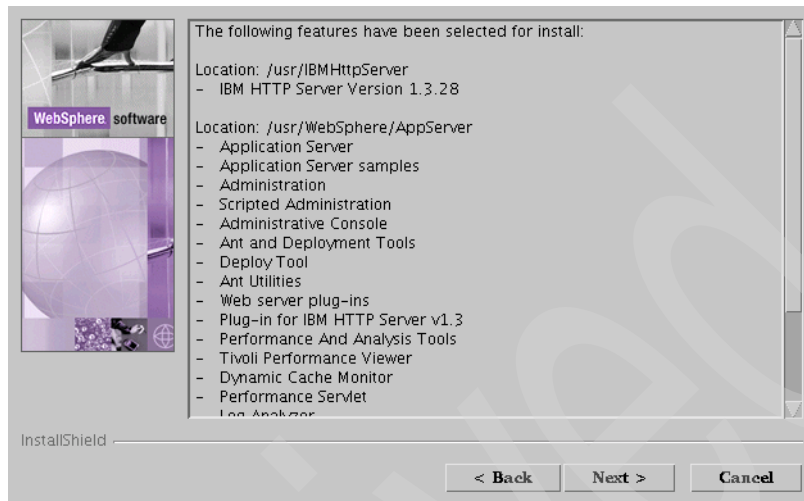


Figure 5-58 WebSphere Application Server V5.1: Actions to be performed

10. After the installation, you will be asked to register the product, as shown in Figure 5-59. Click **Next** to continue.

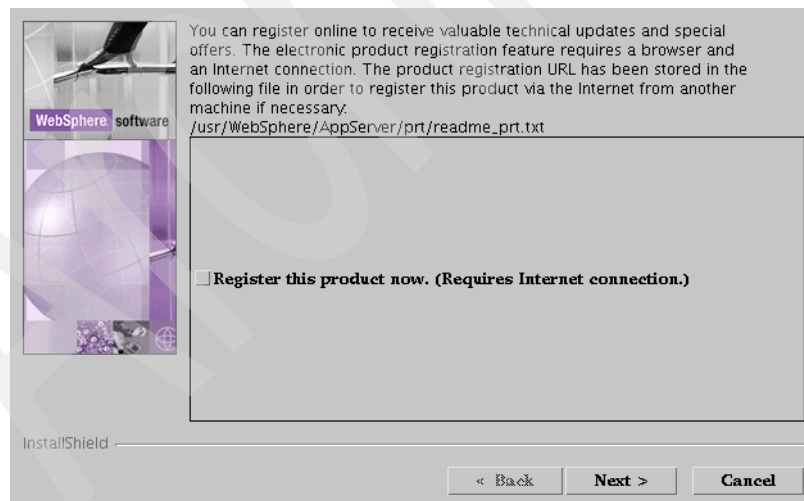


Figure 5-59 WebSphere Application Server V5.1: Register window

11. Click **Finish** to terminate the installation, as shown in Figure 5-60 on page 265.

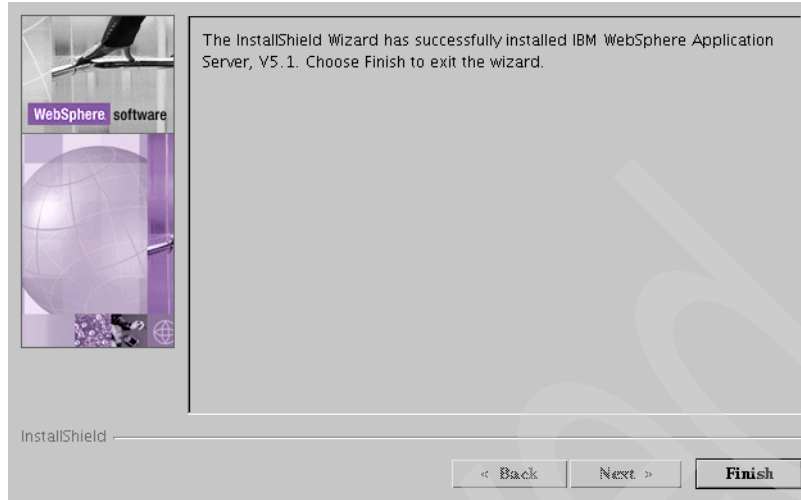


Figure 5-60 WebSphere Application Server V5.1: Exit installation window

12. Select **Start the Server**, and after it starts, click **Exit** to close the window, as shown in Figure 5-61.

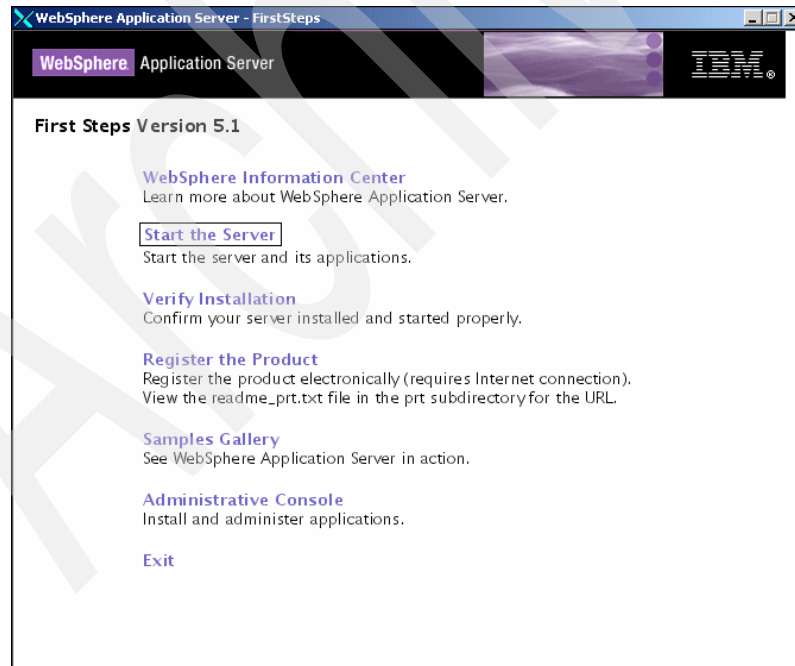


Figure 5-61 WebSphere Application Server V5.1: After installation window

13. After starting the server, you can test the installation by opening a Web browser and typing the following URL, as shown in Figure 5-62:

`http://hostname:9080/snoop`

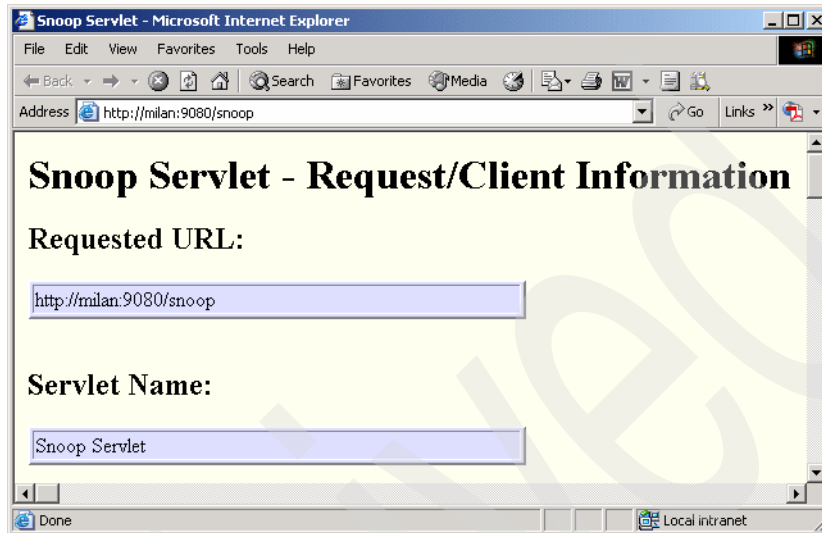


Figure 5-62 WebSphere Application Server V5.1: Testing the installation

The WebSphere Application Server V5.1 is installed and running. Now, you need to install WebSphere Application Server V5.1 Fix Pack 1. You can download Fix Pack 1 from the following FTP site:

<ftp://service.software.ibm.com/software/websphere/appserv/support/fixpacks/was51/cumulative/cf5101>

5.4 Tivoli Access Manager and Access Manager WebSEAL installation

This section describes the steps to install and configure IBM Tivoli Access Manager Version 5.1 and IBM Tivoli Access Manager WebSEAL Version 5.1 on an AIX 5L machine.

The installation and configuration order is important to have a successful installation. Perform the following steps:

1. Install Global Security Kit (GSKit), performing the following steps:
 - a. Log in as root.
 - b. Mount the *IBM Tivoli Access Manager Base Version 5.1* CD.

- c. Enter the following command to install the 32-bit runtime package:

```
installp -acgXd /cdrom/usr/sys/inst.images gskta.rte
```

Note: If you are installing GSKit on an IBM Tivoli Directory Server system, both the 32-bit and 64-bit runtime packages are required. To install the 64-bit package, enter the following command:

```
installp -acgXd cd_mount_point/usr/sys/inst.images gsksa.rte
```

After you install GSKit, no configuration is necessary.

2. Install IBM Tivoli Directory Client, performing the following steps:

- a. Log in as root.

- b. Mount the CD.

- c. Enter the following command to install the LDAP package:

```
installp -acgXd /cdrom/usr/sys/inst.images ldap.client  
ldap.max_crypto_client
```

After you install IBM Tivoli Directory Client, no configuration is necessary.

3. Install IBM Tivoli Access Manager policy server, performing the following steps:

- a. Log in as root.

- b. Mount the *IBM Tivoli Access Manager Base Version 5.1* CD.

- c. Enter the following command to install the IBM Tivoli Access Manager Runtime package:

```
installp -acgXd /cdrom/usr/sys/inst.images PD.RTE
```

- d. Enter the following command to install the IBM Tivoli Access Manager policy server package:

```
installp -acgXd /cdrom/usr/sys/inst.images PD.Mgr
```

Note: When installing the policy server, you must install the Access Manager Runtime first. However, you must not configure this component until the policy server is installed.

- e. Copy the `secschema.def` file from the common directory located in the Base Installation CD to a temporary directory in the LDAP server and run the **ldapmodify** command, as shown in the Example 5-4 on page 268:

```
ldapmodify -v -h ldap_hostname -p ldap_port -D ldap_user -w  
ldap_password -f secschema.def
```

Example 5-4 Updating the LDAP schema

```
ldapmodify -v -h nice -p 389 -D cn=root,dc=abbc,dc=com -w ibm4root -f  
secschema.def
```

4. Install IBM Tivoli Access Manager WebSEAL, performing the following steps:
 - a. Log in as root.
 - b. Mount the *IBM Tivoli Access Manager Web Security Version 5.1 CD*.
 - c. Install GSKit, if it is not installed.
 - d. Install IBM Tivoli Directory Client, if it is not installed.
 - e. Install the IBM Tivoli Access Manager Runtime package, if it is not installed.
 - f. Enter the following command to install the IBM Tivoli Access Manager Web Security Runtime package:

```
installp -acgXd /cdrom/usr/sys/inst.images PDWeb.RTE
```
 - g. Enter the following command to install the IBM Tivoli Access Manager WebSEAL server package:

```
installp -acgXd /cdrom/usr/sys/inst.images PDWeb.Web
```
5. Configure the policy server and WebSEAL:
 - a. You must log in as root and run the **pdconfig** command to configure the policy server and WebSEAL.
 - b. Figure 5-63 shows the pdconfig menu. Select **1** and press Enter to configure the installed packages.

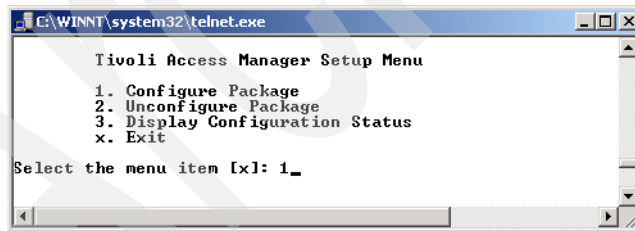


Figure 5-63 Tivoli Access Manager: pdconfig menu

- c. Select **1** and press Enter to configure the Access Manager Runtime, as shown in Figure 5-64 on page 269.

```
C:\WINNT\system32\telnet.exe

Tivoli Access Manager Configuration Menu

1. Access Manager Runtime Configuration
2. Access Manager Policy Server Configuration
3. Access Manager WebSEAL Configuration
x. Return to the Tivoli Access Manager Setup Menu

Select the menu item [x]: 1
Tivoli Common Directory logging is already configured.
You may choose to use the configured location for log files, or use an applicati
on specific log directory.

Do you want to use Tivoli Common Directory logging <y/n> [No]: n

Log files for this application will be created in directory: /var/PolicyDirector
/log

1. LDAP
Registry [1]: 1
LDAP server host name: nice
LDAP server port [389]: 389
The package has been configured successfully.

Press Enter to continue.
```

Figure 5-64 Tivoli Access Manager: Configuring Tivoli Access Manager Runtime

- d. Once configured, the package will be removed from the configuration menu. Select **1** and press Enter to configure the Access Manager policy server, as shown in Figure 5-65.

```
C:\WINNT\system32\telnet.exe

Tivoli Access Manager Configuration Menu

1. Access Manager Policy Server Configuration
2. Access Manager WebSEAL Configuration
x. Return to the Tivoli Access Manager Setup Menu

Select the menu item [x]: 1
LDAP administrator ID [cn=root]: cn=root,dc=abbc,dc=com
LDAP administrator password:*****
Do you want to enable SSL between the
Tivoli Access Manager policy server and the LDAP server <y/n> [Yes]? n

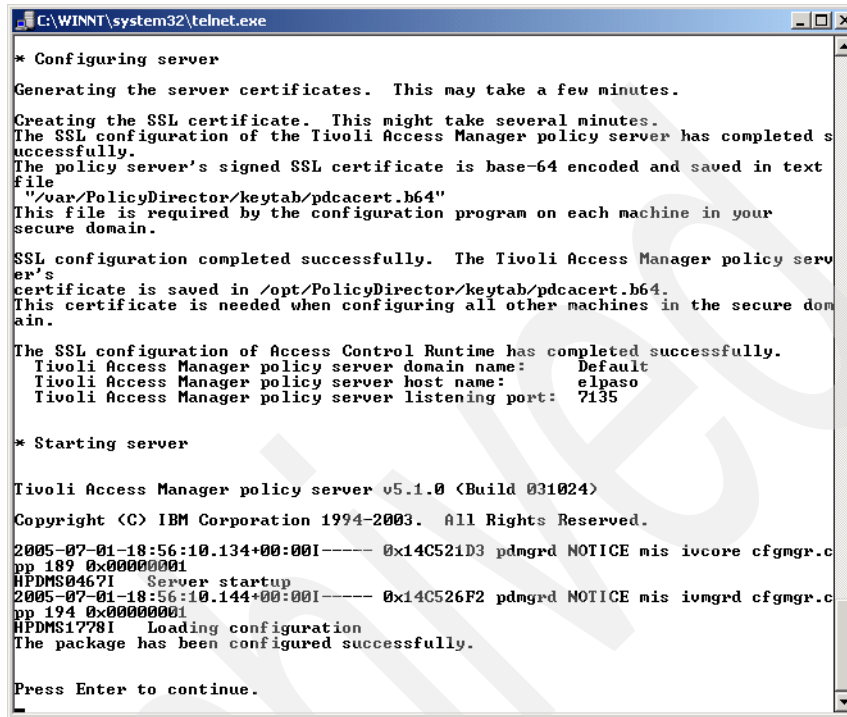
Provide a password for the Access Manager administrator account.
The administrator login name is sec_master and cannot be changed.

Tivoli Access Manager administrator password:
Confirm password:*****

Policy server SSL port [7135]:
SSL certificate lifecycle [365]:
```

Figure 5-65 Tivoli Access Manager: Configuring the policy server

- e. After configuring the Access Manager policy server, the service automatically starts, as shown in Figure 5-66.



```
C:\WINNT\system32\telnet.exe

* Configuring server
Generating the server certificates. This may take a few minutes.
Creating the SSL certificate. This might take several minutes.
The SSL configuration of the Tivoli Access Manager policy server has completed successfully.
The policy server's signed SSL certificate is base-64 encoded and saved in text file
"/var/PolicyDirector/keytab/pdcacert.b64"
This file is required by the configuration program on each machine in your secure domain.

SSL configuration completed successfully. The Tivoli Access Manager policy server's
certificate is saved in /opt/PolicyDirector/keytab/pdcacert.b64.
This certificate is needed when configuring all other machines in the secure domain.

The SSL configuration of Access Control Runtime has completed successfully.
Tivoli Access Manager policy server domain name: Default
Tivoli Access Manager policy server host name: elpaso
Tivoli Access Manager policy server listening port: 7135

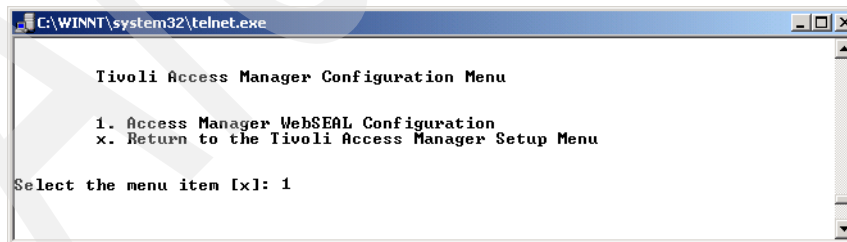
* Starting server

Tivoli Access Manager policy server v5.1.0 <Build 031024>
Copyright <C> IBM Corporation 1994-2003. All Rights Reserved.
2005-07-01-18:56:10.134+00:001----- 0x14C521D3 pdmgrp NOTICE mis ivcore cfgmgr.c
pp 189 0x00000001
HPDMS04671 Server startup
2005-07-01-18:56:10.144+00:001----- 0x14C526F2 pdmgrp NOTICE mis ivmgrd cfgmgr.c
pp 194 0x00000001
HPDMS17781 Loading configuration
The package has been configured successfully.

Press Enter to continue.
```

Figure 5-66 Tivoli Access Manager: Starting the policy server

- f. Select 1 and press Enter to configure Access Manager WebSEAL, as shown in Figure 5-67.



```
C:\WINNT\system32\telnet.exe

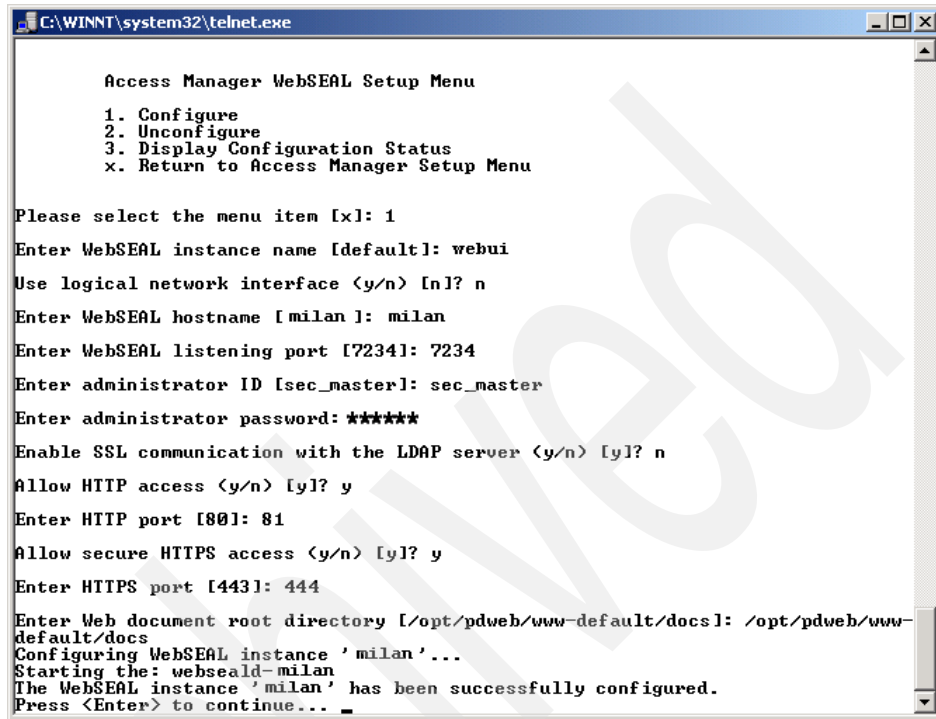
Tivoli Access Manager Configuration Menu

1. Access Manager WebSEAL Configuration
x. Return to the Tivoli Access Manager Setup Menu

Select the menu item [x]: 1
```

Figure 5-67 Tivoli Access Manager: Configuring WebSEAL

- g. Select **1** and press Enter to configure a new instance, as shown in Figure 5-68.



```
C:\WINNT\system32\telnet.exe

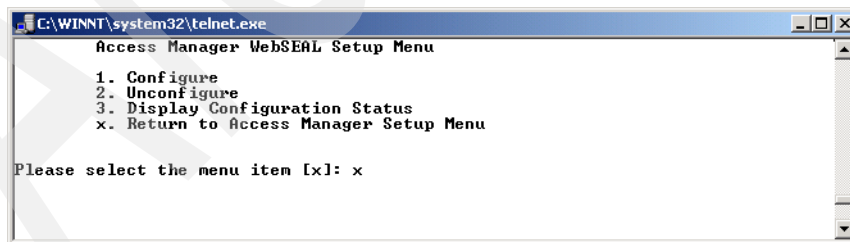
Access Manager WebSEAL Setup Menu

1. Configure
2. Unconfigure
3. Display Configuration Status
x. Return to Access Manager Setup Menu

Please select the menu item [x]: 1
Enter WebSEAL instance name [default]: webui
Use logical network interface <y/n> [n]? n
Enter WebSEAL hostname [milan]: milan
Enter WebSEAL listening port [7234]: 7234
Enter administrator ID [sec_master]: sec_master
Enter administrator password: *****
Enable SSL communication with the LDAP server <y/n> [y]? n
Allow HTTP access <y/n> [y]? y
Enter HTTP port [80]: 81
Allow secure HTTPS access <y/n> [y]? y
Enter HTTPS port [443]: 444
Enter Web document root directory [/opt/pdweb/www-default/docs]: /opt/pdweb/www-
default/docs
Configuring WebSEAL instance 'milan'...
Starting the: webseald-milan
The WebSEAL instance 'milan' has been successfully configured.
Press <Enter> to continue... _
```

Figure 5-68 Tivoli Access Manager: Configuring a WebSEAL instance

- h. Select **x** and press Enter to leave the Access Manager WebSEAL setup menu, as shown in Figure 5-69.



```
C:\WINNT\system32\telnet.exe

Access Manager WebSEAL Setup Menu

1. Configure
2. Unconfigure
3. Display Configuration Status
x. Return to Access Manager Setup Menu

Please select the menu item [x]: x
```

Figure 5-69 Tivoli Access Manager: Leaving WebSEAL Setup

- i. Select **x** and press Enter to leave the Access Manager configuration menu, as shown in Figure 5-70 on page 272.

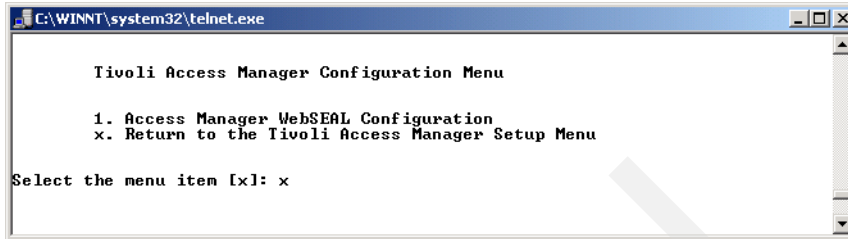


Figure 5-70 Tivoli Access Manager: Leaving Configuration Menu

- j. To verify that all the components have been configured, select **3** and press Enter in the Tivoli Access Manager setup menu, as shown in Figure 5-71.

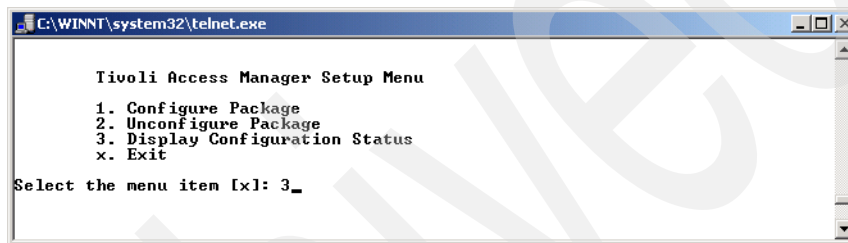


Figure 5-71 Tivoli Access Manager: Selecting Display Configuration

- k. Figure 5-72 shows all the components with a Configured status of Yes. Press Enter to return to the Tivoli Access Manager setup menu and then select **x** to leave the pdconfig utility.

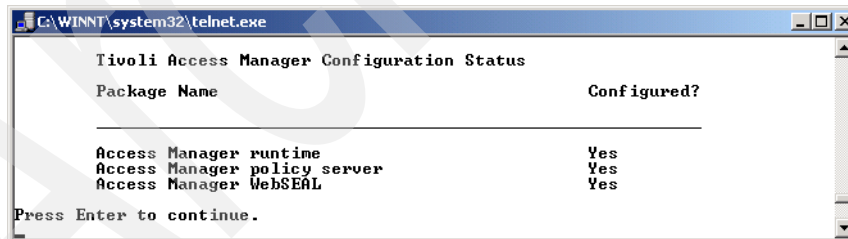


Figure 5-72 Tivoli Access Manager: Display configuration

6. Install the IBM Tivoli Access Manager Java Runtime, performing the following steps:
 - a. Log in as root.
 - b. Mount the *IBM Tivoli Access Manager Base Version 5.1 CD*.

- c. Run the AIX 5L command `ls1pp -1 |grep Java` and check if Java Version 1.3.1.5 or later is installed. If yes, skip this item. If not, run the following command:

```
installp -acgXd /cdrom/usr/sys/inst.images Java131.rte
```

- d. Enter the following command to install the IBM Tivoli Access Manager Java Runtime package:

```
installp -acgXd /cdrom/usr/sys/inst.images PDJ.rte
```

- e. Change the PATH variable to include the WebSphere Java home directory:

```
export PATH=/usr/WebSphere/AppServer/java/jre/bin:$PATH
```

- f. Go to the `/opt/PolicyDirector/sbin` directory and run the `pdjrtecfg` command to configure the Access Manager Java Runtime, as shown in the Example 5-5. Do not use `pdconfig` to configure it.

```
./pdjrtecfg -action config -host policy_server_host -java_home  
/usr/WebSphere/AppServer/java/jre
```

Example 5-5 Running the pdjrtecfg command

```
#!/opt/PolicyDirector/sbin/pdjrtecfg -action config -host milan -java_home  
/usr/WebSphere/AppServer/java/jre
```

To test the Tivoli Access Manager and Tivoli Access Manager WebSEAL installation, open a Web browser and enter:

```
https://webseal_localhost
```

Select **Yes** in the secure notification window and log in with the `sec_master` user and its password. The WebSEAL page opens, as shown in Figure 5-73 on page 274.

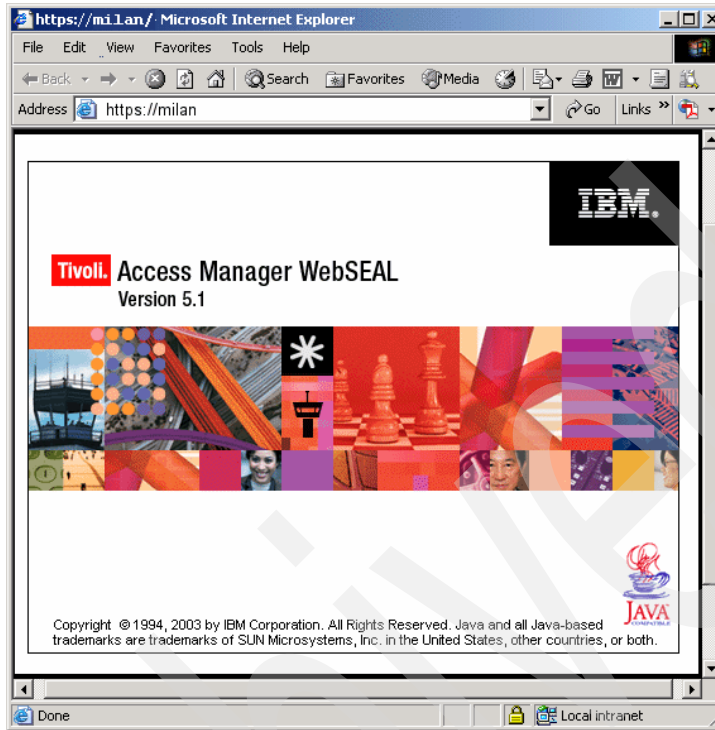


Figure 5-73 Tivoli Access Manager: WebSEAL window

At this point, IBM Tivoli Access Manager and IBM Tivoli Access Manager WebSEAL are configured and running.



Enterprise Directory integration

In this chapter, we explain what how to configure the Enterprise Directory integration. This feature provides the link between your Tivoli server and the LDAP server.

We discuss the following topics:

- ▶ Enterprise Directory integration
- ▶ Directory Query Library and Directory Query

6.1 Enterprise Directory integration

In this section, we explain how to configure the Tivoli server to access information stored in a directory server.

In our environment, we installed IBM Tivoli Directory Server Version 5.2 on a Windows system called nice. We describe the installation in Chapter 5, “Extra components” on page 217.

6.1.1 Exchanging data with a directory server

We must use a `DirectoryContext` object to exchange data with a directory server. A `DirectoryContext` object is comparable to a RIM object, after the `DirectoryContext` object contains the settings needed to access the directory server, just like the RIM object that contains the settings to connect to an RDBMS.

The installation of the Directory Query Facility creates one default `DirectoryContext` object named `directory`, even if no configuration was done during the installation. Other `DirectoryContext` objects can be created later.

Note: Tivoli Resource Manager uses the `directory` `DirectoryContext` object. It is hard coded and cannot be changed.

6.1.2 Working with `DirectoryContext` objects

`DirectoryContext` objects are manipulated using the command line interface. There is no GUI available to create, configure, remove, and manage these objects.

Creating a `DirectoryContext` object

The `DirectoryContext` object is used to connect the directory server.

The `wcrtdirectx` command is used to create a Directory Query context.

The syntax is:

```
wcrtdirectx [-i] -s server -u user -c naming_context [-f provider] [-p port] [-P  
ssl_port] [-S y|n] [-k keystore_path] directory_ctx_name
```

Where:

-i Specifies that the password must be read from standard input.

- s** Specifies the server.
- u** Specifies the directory server administrator or a user with equivalent authority.
- c** Specifies the naming context to use for the search.
- f** Specifies the Java class name that identifies the directory access protocol. The default is "com.sun.jndi.ldap.LdapCtxFactory".
- p** Specifies a server port for a non-secure connection. If not specified, the default value 389 is used.
- P** Specifies a server port for a secure (SSL) connection. If not specified, the default value 636 is used.
- S** Enables the Secure Sockets Layer (SSL) between the directory server and the Tivoli region. Y specifies enable, n specifies do not enable. If not specified, SSL is disabled.
- k** Specifies the path of the keystore containing the certificates used during an SSL connection. If you specify this option, you must also specify the `keystore_path` password.

dirquery_ctx_name Specifies the name of the new Directory Query context.

Example 6-1 shows the creation of the DirectoryContext object `people`. This new object will contact the LDAP server named `nice`, using the LDAP administrator user `cn=root,dc=abbc,dc=com`, and search for the naming context `ou=people,dc=abbc,dc=com`. The password is the LDAP administrator password.

Example 6-1 Creating the DirectoryContext object people

```
wcrtmdirctx -s nice -u cn=root,dc=abbc,dc=com -c ou=people,dc=abbc,dc=com people
Directory password:*****
Retype new password:*****
```

Configuring a DirectoryContext object

After creating a DirectoryContext object, you can change its configuration or its password if needed. These changes can be necessary if any change are made in the directory server.

Because we did not perform any LDAP configuration during the server installation, the DirectoryContext object named `directory` was created only with the default settings, as shown in Example 6-2 on page 278.

Example 6-2 Displaying the contents of the directory DirectoryContext object

```
# wgetdirctx directory
Server:
Port:          389
SSL Port:     636
User:
Provider:     com.sun.jndi.ldap.LdapCtxFactory
Naming context:
Keystore path:
SSL enabled:  FALSE
```

Because Tivoli Resource Manager uses the DirectoryContext object, we need to configure it to connect to the LDAP server installed on the nice machine and search for the naming context ou=people,dc=abbc,dc=com. To do this, run the following command, and after that, see the changes in Example 6-3:

```
wsetdirctx -s nice -u cn=root,dc=abbc,dc=com -c ou=people,dc=abbc,dc=com
directory
```

The directory DirectoryContext object has no password, so you need to run the **wsetdirctxpw** command to input the LDAP administrator password. Because the object has no password, just press Enter when it asks for the old password.

```
wsetdirctxpw directory
```

Now, you can display the contents of the DirectoryContext object again, as shown in Example 6-3.

Example 6-3 Displaying the contents of the directory DirectoryContext object changed

```
# wgetdirctx directory
Server:      nice
Port:       389
SSL Port:   636
User:       cn=root,dc=abbc,dc=com
Provider:   com.sun.jndi.ldap.LdapCtxFactory
Naming context: ou=people,dc=abbc,dc=com
Keystore path:
SSL enabled:  FALSE
```

Removing a DirectoryContext object

A DirectoryContext object can be removed from the Tivoli environment by running the **wdel** command.

In our case, there is no reason to have two DirectoryContext objects configured to the same directory server and searching for the same naming context, so we can remove the people DirectoryContext object:

```
wdel @DirectoryContext:people
```

You can run the **wlookup** command to check how many DirectoryContext objects are configured in your environment:

```
wlookup -ar DirectoryContext
```

Managing a DirectoryContext object

After creating and configuring the DirectoryContext object, we must associate the endpoints registered in the Tivoli region with their owners. It is not possible to associate a user with more than one endpoint, and one endpoint cannot have more than one user associated to it.

You can use the **wmanagedir** command to create, remove, or modify the association between the endpoint and the user on the directory server.

Example 6-4 shows the **wmanagedir** command creating the associations for the Luciano, Vasfi, and Sanver users to their endpoints.

Example 6-4 Running the wmanagedir command to create the associations

```
# wmanagedir -a "cn=Luciano Peetz,ou=people,dc=abbc,dc=com" -l kwc12z
directory
DISQD0051I The requested operation completed successfully.
# wmanagedir -a "cn=Vasfi Gucer,ou=people,dc=abbc,dc=com" -l lizbon directory
DISQD0051I The requested operation completed successfully.
# wmanagedir -a "cn=Sanver Ceylan,ou=people,dc=abbc,dc=com" -l madrid directory
DISQD0051I The requested operation completed successfully.
```

After that, the Enterprise Directory integration is configured. To have access to the directory server user data, create a Directory Query Library and a Directory Query. We describe this configuration next.

6.2 Directory Query Library and Directory Query

Directory Query Libraries and Directory Queries are similar to Inventory Query Libraries and Inventory Queries. A Directory Query Library is a collection of Directory Queries. A Directory Query enables you to find information about users or endpoints defined in the Enterprise Directory.

Query Directories can be used to retrieve any available data from the Enterprise Directory or they can be used to select subscribers. The subscribers queries can

be used to specify the targets for an Activity Plan Monitor activity or to select the subscribers for a Change Manager reference model.

Before you can create a query directory, you must create a Directory Query Library. A Directory Query Library is used to group similar query directories.

Directory Query Libraries can be created from the GUI or by using the **wcrtdirquerylib** command. Directory Query Libraries are created within a policy region. The administrator creating the library or the queries requires the super or the senior authorization role on the policy region, and the policy region must have the managed resources Directory Query Library and Directory Query.

Perform the following steps to create a Directory Query Library named LDAP_QUERY and the directory queries ABBC_Users, ABBC_US_Users, and ABBC_BR_Users. We use the Tivoli desktop to perform these tasks. If you prefer to use the command line, you need to use the **wcrtdirquerylib** and **wcrtdirquery** commands. Refer to the *Tivoli Management Framework Reference Manual, Version 4.1.1*, SC32-0806, for the syntax of these commands.

1. From the Tivoli desktop open the policy region where you will create the Directory Query Library. In our environment, we create the Directory Query Library in the rome-region policy region, as shown in Figure 6-1 on page 281.

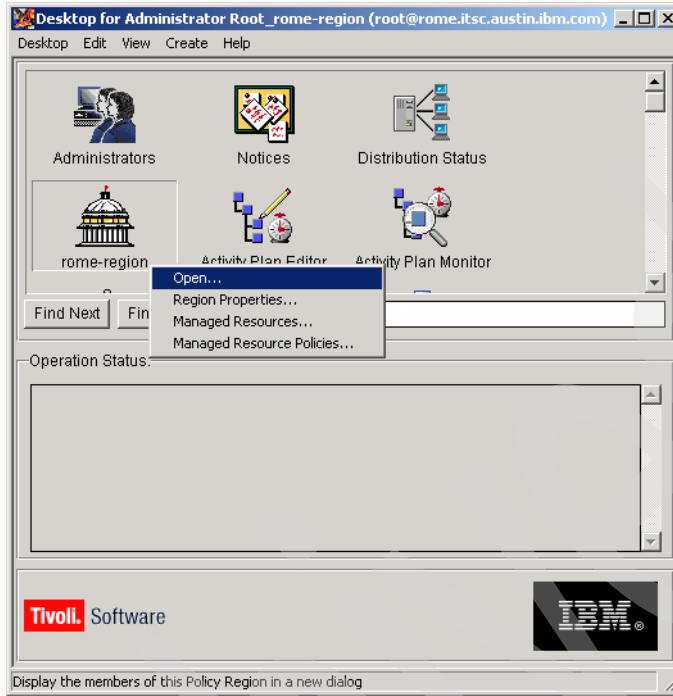


Figure 6-1 Directory Query: Opening the policy region

2. Select **Create** → **Directory Query Library**, as shown in the Figure 6-2.

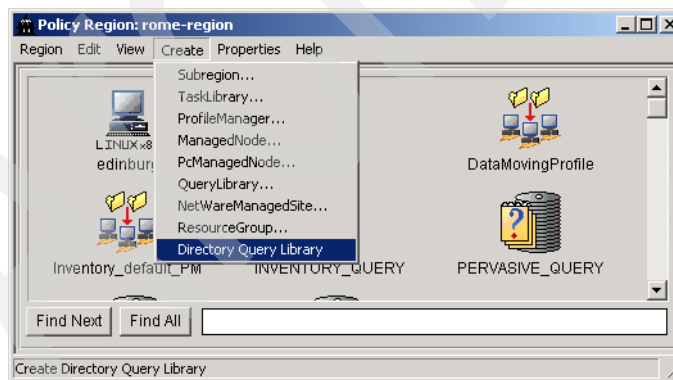


Figure 6-2 Directory Query: Creating a Directory Query Library

3. In Figure 6-3, enter the name LDAP_QUERY and click **Create & Close**.

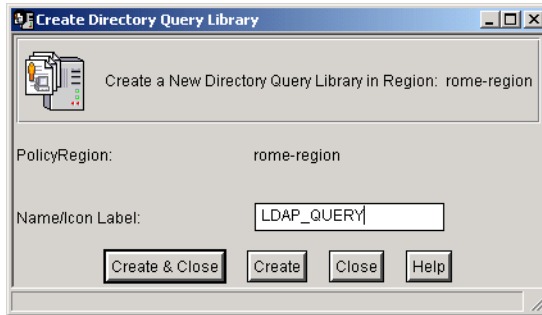


Figure 6-3 Directory Query: Creating LDAP_QUERY Directory Query Library

4. Right-click **LDAP_QUERY** and select the item **Create Directory Query**, as shown in the Figure 6-4.

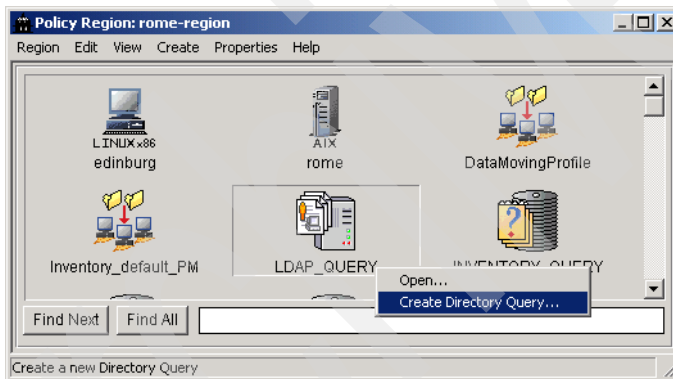


Figure 6-4 Directory Query: Creating the Directory Query

5. Enter the Directory Query Name ABBC_Users, select **directory** as the Directory Context, and enter o=abbc in the Search String.

6. Add the Attributes `tmeObjectId`, `tmeObjectLabel`, `cn`, `o`, and `ou` and click **Create**, as shown in Figure 6-5.

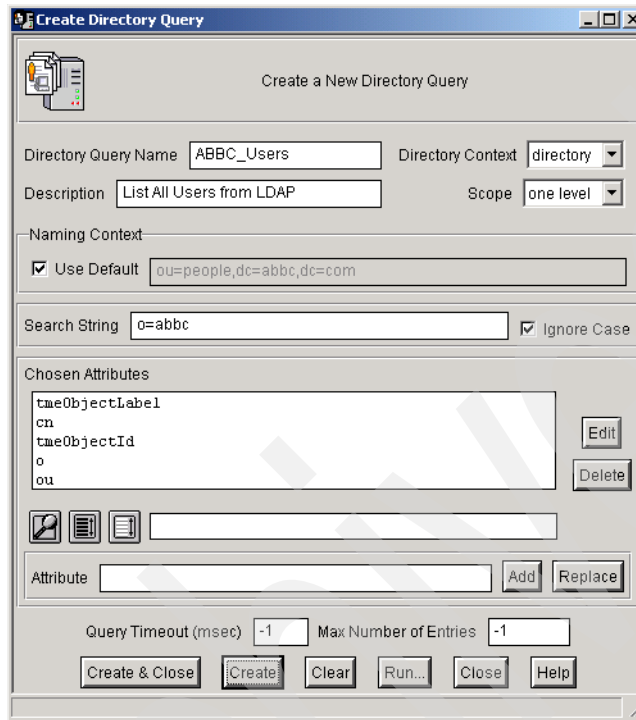


Figure 6-5 Directory Query: Creating the ABBC_Users Directory Query

- It is not necessary to clear the window to create the Directory Query ABBC_US_Users. Change the Directory Query Name to ABBC_US_Users, the Description, and the Search String to ou=abbcus, as shown in Figure 6-6. Click **Create**.

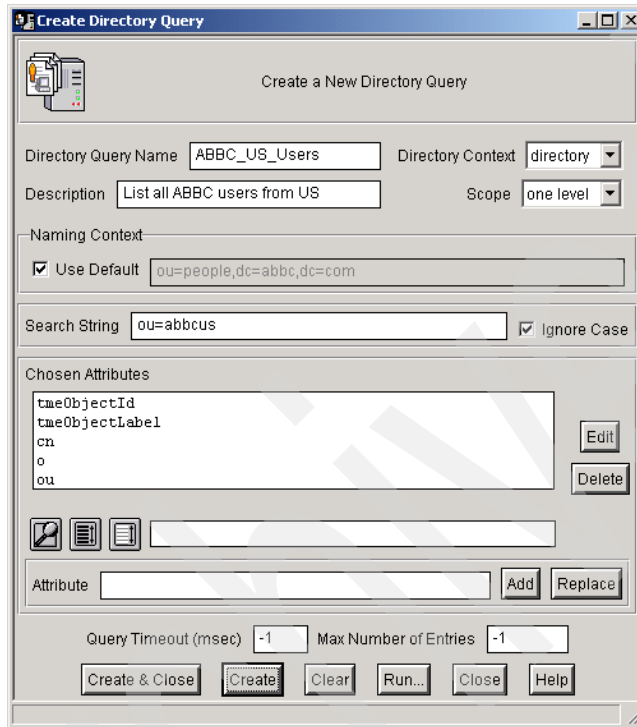


Figure 6-6 Directory Query: Creating the ABBC_US_Users Directory Query

8. It is not necessary to clear the window to create the Directory Query ABBC_BR_Users. Change the Directory Query Name to ABBC_BR_Users, the Description, and the Search String to ou=abbcbr, as shown in Figure 6-7. Click **Create & Close**.

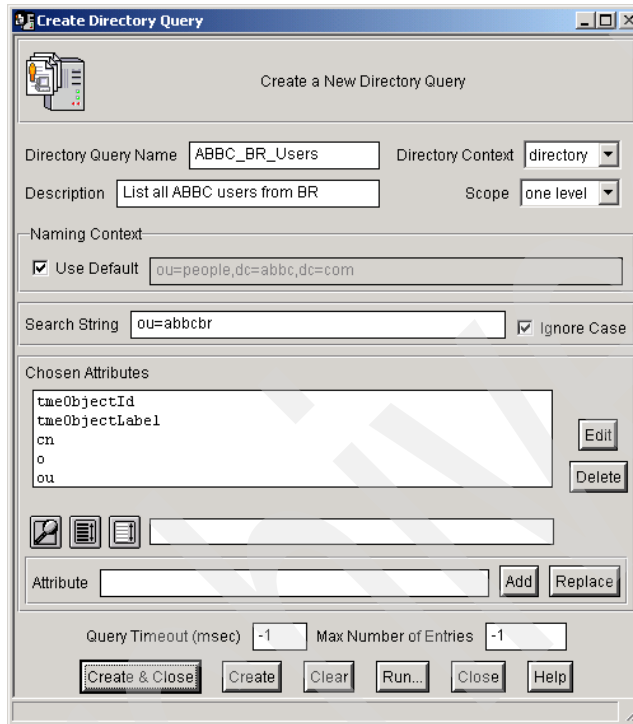


Figure 6-7 Directory Query: Creating the ABBC_BR_Users Directory Query

9. Now, you can run the directory queries created from GUI, as shown in Figure 6-8, or by the command line, as shown in Example 6-5.

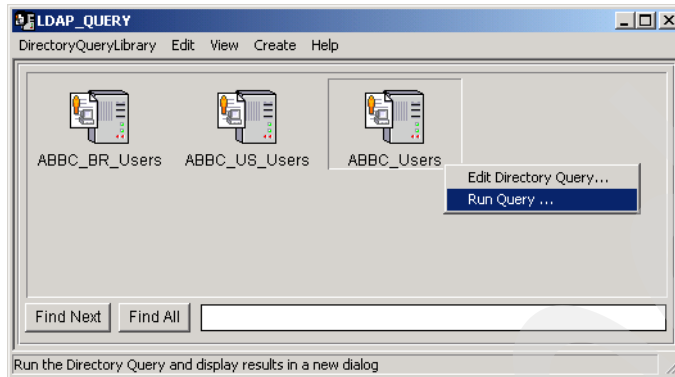


Figure 6-8 Directory Query: Running the Directory Query

Example 6-5 Running the `wrundirquery` command

```
# wrundirquery ABBC_Users
Directory Query Name: ABBC_Users
Number of Entries : 3

ou:abbcus
tmeObjectId:1254603259.4.522+#TMF_Endpoint::Endpoint#
tmeObjectLabel:lizbon
o:abbc
cn:Vasfi Gucer

ou:abbcus
tmeObjectId:1254603259.5.522+#TMF_Endpoint::Endpoint#
tmeObjectLabel:madrid
o:abbc
cn:Sanver Ceylan

ou:abbcbr
tmeObjectId:1254603259.2.522+#TMF_Endpoint::Endpoint#
tmeObjectLabel:kwc12z
o:abbc
cn:Luciano Peetz

# wrundirquery ABBC_US_Users
Directory Query Name: ABBC_US_Users
Number of Entries : 2

ou:abbcus
```

```
tmeObjectId:1254603259.4.522+#TMF_Endpoint::Endpoint#  
tmeObjectLabel:lizbon  
o:abbc  
cn:Vasfi Gucer
```

```
ou:abbcus  
tmeObjectId:1254603259.5.522+#TMF_Endpoint::Endpoint#  
tmeObjectLabel:madrid  
o:abbc  
cn:Sanver Ceylan
```

```
# wrundirquery ABBC_BR_Users  
Directory Query Name: ABBC_BR_Users  
Number of Entries : 1
```

```
ou:abbcbr  
tmeObjectId:1254603259.2.522+#TMF_Endpoint::Endpoint#  
tmeObjectLabel:kwc12z  
o:abbc  
cn:Luciano Peetz
```

The Enterprise Directory integration is configured and running.

Archived



IBM Tivoli Web Gateway for Web User Interface

This chapter explains how to install and configure the IBM Tivoli Web Gateway for Web User Interface and also how to configure the prerequisites of the product.

We discuss the following topics:

- ▶ System requirements
- ▶ Software requirements
- ▶ Installing IBM Tivoli Web Gateway
- ▶ Preparing to use the Web UI

7.1 System requirements

The Tivoli Web Gateway installation requires 500 MB of free disk space for the product installation, 120 MB of free space on /tmp directory for the installation wizard, 1 GB of memory, and 2 GB of paging space on the AIX 5L machine.

Note: The installation process can take about three hours if your AIX 5L machine does not have the memory and paging space required.

7.2 Software requirements

Before installing IBM Tivoli Web Gateway, you must install and configure prerequisite software. We list the prerequisite software here and describe the installation instructions in Chapter 5, “Extra components” on page 217. In this chapter, we cover the configuration steps.

The following prerequisite software must be installed on the same machine with IBM Tivoli Web Gateway:

- ▶ IBM DB2 UDB V8.1 Server with Fix Pack 6
- ▶ IBM WebSphere Application Server V5.1.1
- ▶ IBM Tivoli Access Manager WebSEAL V5.1

The following prerequisite software must be installed in the environment:

- ▶ LDAP server
- ▶ IBM Tivoli Access Manager V5.1

7.2.1 Configuring IBM DB2 UDB V8.1 server

Before installing IBM Tivoli Configuration Manager Web Gateway, you need to install IBM DB2 UDB V8.1 with Fix Pack 6 locally and prepare it to store the database and tables that will be created during the Web Gateway installation.

We describe the installation procedure of DB2 UDB in Chapter 5, “Extra components” on page 217. Perform the following steps to configure the RDBMS server to work with IBM Tivoli Configuration Manager Web Gateway:

1. Log in to the RDBMS server as the root user.

2. Create the db2grp1 group:

```
mkgroup -A db2grp1
```

3. Create the db2fgrp1 group:

```
mkgroup -A db2fgrp1
```

4. Create the db2inst1 user:

```
mkuser pgrp='db2grp1' home='/home/db2inst1' db2inst1
```

5. Create the db2fenc1 user:

```
mkuser pgrp='db2fgrp1' db2fenc1
```

6. Set the password for the users db2inst1 and db2fenc1, as described in Example 7-1.

Example 7-1 Setting the password for the users db2inst1 and db2fenc1

```
# passwd db2inst1
Changing password for "db2inst1"
db2inst1's New password:*****
Enter the new password again:*****
# pwadm -c db2inst1

# passwd db2fenc1
Changing password for "db2fenc1"
db2fenc1's New password:*****
Enter the new password again:*****
# pwadm -c db2fenc1
```

7. Create the instance dn2inst1:

```
/usr/opt/db2_08_01/instance/db2icrt -u db2fenc1 db2inst1
```

8. Check if the following lines are included in the /etc/services file. If the port number of the services are different, change them for the following ports:

```
DB2_db2inst1      50000/tcp
DB2_db2inst1_1    50001/tcp
DB2_db2inst1_2    50002/tcp
DB2_db2inst1_END  50003/tcp
```

9. Log in to the RDBMS as the db2inst1 user, the owner of the instance.
10. Set the value DB2_db2inst1 to the SVCENAME variable using the following command:

```
db2 update dbm cfg using SVCENAME DB2_db2inst1
```

11. Set the DB2COMM variable to tcpip using the following command:

```
db2set DB2COMM=tcpip
```

12. Start the db2service:

```
db2start
```

13. The users dmsuser and dmsadmin have to be created on the RDBMS server, as shown in Example 7-2 on page 292.

Example 7-2 Creating the users dmsuser and dmsadmin

```
# mkuser pgrp='db2grp1' dmsuser
# passwd dmsuser
Changing password for "dmsuser"
dmsuser's New password:*****
Enter the new password again:*****
# pwdadm -c dmsuser

# mkuser pgrp='db2grp1' dmsadmin
# passwd dmsadmin
Changing password for "dmsadmin"
dmsadmin's New password:*****
Enter the new password again:*****
# pwdadm -c dmsadmin
```

14. After creating the users dmsadmin and dmsuser, edit the .profile of both users and include the following lines:

```
# The following three lines have been added by UDB DB2.
if [ -f /home/db2inst1/sqllib/db2profile ]; then
    . /home/db2inst1/sqllib/db2profile
fi
```

Now, your RDBMS server is configured.

7.2.2 Configuring IBM WebSphere Application Server

IBM WebSphere Application Server must be configured prior the Web Gateway installation. To do that, perform the following steps:

1. Edit the file `$WAS_HOME/properties/soap.client.props`, and set the `com.ibm.SOAP.requestTimeout` parameter to 540.
2. Run the command `$WAS_HOME/bin/stopServer.sh server1`.
3. Run the command `$WAS_HOME/bin/startServer.sh server1`.

The IBM WebSphere Application Server is now configured.

7.2.3 Configuring IBM Tivoli Access Manager and IBM Tivoli Access Manager WebSEAL

Before installing the Web Gateway, you need to create SSL keystores and configure the IBM Tivoli Access Manager and IBM Tivoli Access Manager WebSEAL components. We describe the installation procedure for these components in Chapter 5, "Extra components" on page 217.

Creating the SSL keystores

Run the following command to create the SSL keystores. The command must be run in a single line (it is shown as three lines for readability).

```
java com.tivoli.mts.SvrSslCfg appl_nm security_pwd policy_srv_hostnm  
auth_srvr_hostnm policy_srvr_port auth_srv_port config_file keystores_file  
operation
```

Where:

appl_nm	The name of Access Manager application to create and associate with the SSL communication. The application name must be unique.
security_pwd	The password associated with the sec_master user.
policy_srv_hostnm	The name of the system where the Access Manager policy server process, ivmgrd, is running.
auth_srv_hostnm	The name of the system where the Access Manager authorization server process, ivacl, is running.
policy_srv_port	The port used for SSL communication with the policy server. The default is 7135.
auth_srv_port	The port used for SSL communication with the authorization server. The default is 7136.
config_file	The URL to the configuration file. The URL must use the file:/// format. The default is java_home/PdPerm.properties.
keystores_file	The URL to the keystores file. The URL must use the file:/// format. The default is java_home/PdPerm.ks. The PdPerm.properties and PdPerm.ks files must be in the same directory.
operation	The operation to perform. Valid operations are create, replace, and unconfig.

Note: To successfully run this command, ensure that WAS_HOME/java/jre/bin is the first entry for the PATH environment variable. On AIX 5L, for example, enter the following command:

```
export PATH=/usr/WebSphere/AppServer/java/jre/bin:$PATH
```

Example 7-3 on page 294 is an example of the command to create the SSL configuration file and keystores.

Example 7-3 Creating SSL keystores

```
#export PATH=/usr/WebSphere/AppServer/java/jre/bin:$PATH
#java com.tivoli.mts.SvrSslCfg webgw secmaster milan milan 7135 7136
file:///usr/WebSphere/AppServer/java/jre/PolicyDirector/PdPerm.properties
file:///usr/WebSphere/AppServer/java/jre/PolicyDirector/PdPerm.ks create
```

When the command runs successfully, check to make sure that the PdPerm.properties file has been created.

To protect the Web objects using WebSEAL, you also need to perform the following tasks on the WebSEAL server:

- ▶ Modify Web Access to use forms authentication.
- ▶ Configure the query_contents file for WebSEAL.
- ▶ Create a junction for the WebSEAL server.

Note: If you are using WebSEAL to protect both Web objects for the Web UI and enrollment URLs for the resource managed by Tivoli Resource Manager, you need to create and maintain two WebSEAL instances. This is due to the different authentication requirements. Web Access uses forms authentication, and Resource Manager uses basic authentication. Each instance will have its own junction point and webseald-instance.conf file.

Configuring Web Access to use forms authentication

To configure the Web Access to use forms authentication, locate the webseald-instance.conf file and change the following settings:

- ▶ In the Basic Authentication section, make sure that you have ba-auth=none.
- ▶ In the Forms section, make sure that there is forms-auth=https.
- ▶ In the Sharing Sessions section, check to make sure that there is use-same-session=yes.

You must restart the WebSEAL server for the changes to take effect, running the following command:

```
pdweb_start restart
```

Configuring the query_contents file for WebSEAL

Configure query_contents on the Web Gateway Server using the instructions in *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1*, SC32-1359. The basic steps, by operating system, are as follows:

1. Copy the query_contents.sh file from the WebSEAL server to the /usr/IBMHttpServer/cgi_bin directory. It is located in the /opt/pdweb/www-instance/docs/cgi-bin directory.
2. Rename query_contents.sh to query_contents:

```
mv query_contents.sh query_contents
```
3. Change access control to the query_contents file by entering:

```
chmod 755 query_contents
```
4. Edit the query_contents file to ensure that the DOCROOTDIR variable, on line 83, is as follows:

```
DOCROOTDIR='pwd `./../htdocs/en_US
```
5. Edit the query_contents file to add the following entry after line 77:

```
SERVER_SOFTWARE=Apache
```
6. Restart IBM HTTP Server, and test query_contents by entering the following URL into a Web browser. The result of this URL should be a 100 return code, followed by a listing of the files and directories in the htdocs subdirectory.

```
http://<WebGateway_hostname>/cgi-bin/query_contents?dirlist=/
```

Creating a junction point

You need to obtain the host name of the WebSEAL server before creating the junction. To find the host name of the WebSEAL server, open a command prompt and run the following commands:

```
pdadmin -a sec_master -p password  
server list
```

To create the junction point, run the following command using the WebSEAL server name from the previous command:

```
server task WebSEAL_SRV_nm create -j -c all -t tcp -h Web_GW_host -p  
Web_GW_http_port junction_point
```

Where:

- | | |
|-----------------------|---|
| WebSEAL_SRV_nm | The WebSEAL server name from the pdadmin server list command |
| Web_GW_host | The host name where WebSphere Application Server was installed |

Web_GW_http_port The HTTP port where IBM HTTP Server is running
junction_point The name of the junction point

Type **exit** to leave the pdadmin command line.

Example 7-4 shows how to create the junction point named /software.

Example 7-4 Creating the junction point named software

```
# pdadmin -a sec_master -p secmaster
pdadmin sec_master> server list
    web-milan.itsc.austin.ibm.com
    web-webseald-milan
pdadmin sec_master> server task web-webseald-milan create -j -c all -t tcp -h
milan.itsc.austin.ibm.com -p 80 /software
Created junction at /software
pdadmin sec_master> exit
```

7.3 Installing IBM Tivoli Web Gateway

This section will describe the steps to install the IBM Tivoli Configuration Manager Web Gateway component using the InstallShield wizard.

This installation program installs:

- ▶ The Web Gateway components (database and server) to perform device management
- ▶ The Web Interface component to perform configuration management operations from a Web browser

Perform the following steps to install Web Gateway:

1. Mount the CD *IBM Tivoli Configuration Manager Web Gateway Version 4.2.2* and run the **setup_aix.bin** command to start the InstallShield wizard.
2. The window shown in Figure 7-1 opens. Select the InstallShield language and click **OK** to continue.

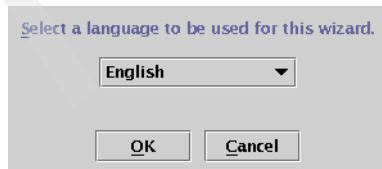


Figure 7-1 Web UI: Installation language

3. The Welcome window opens, as shown in Figure 7-2. Click **Next** to continue.

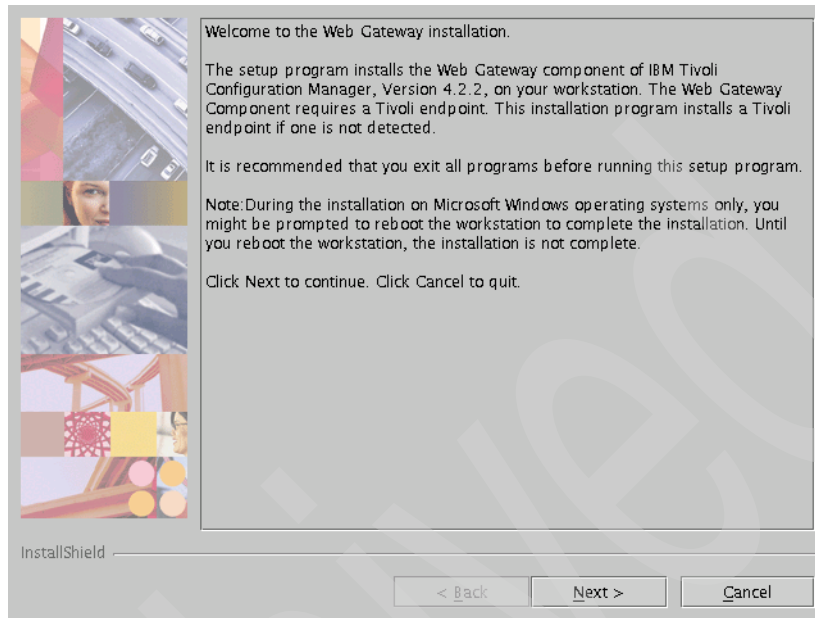


Figure 7-2 Web UI: Welcome window

4. Read the and accept the license agreement, as shown in Figure 7-3, and click **Next** to continue.

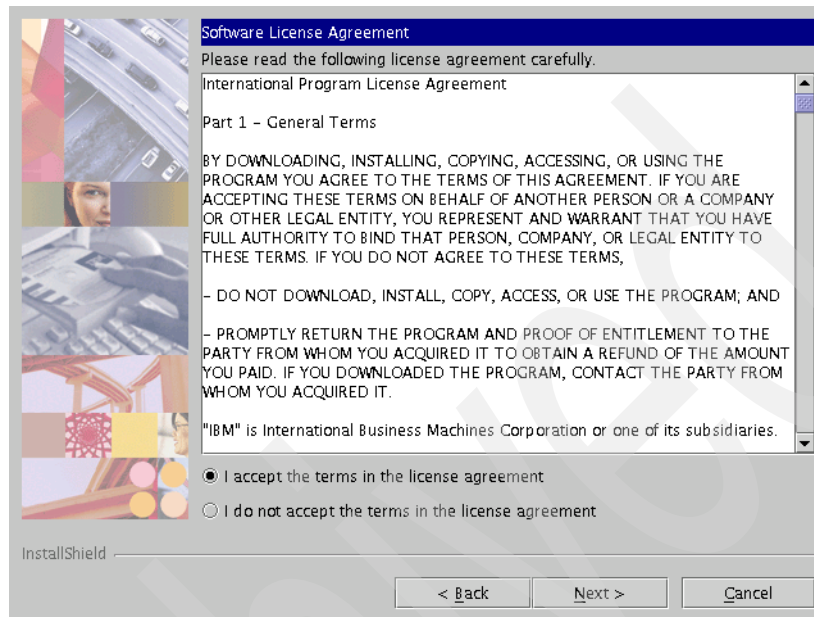


Figure 7-3 Web UI: License Agreement window

5. Select the Tivoli Web Gateway components that you want to install and click **Next** to continue. In our environment, we decided to install all Tivoli Web Gateway components, as shown in Figure 7-4.



Figure 7-4 Web UI: Select components to install

6. Choose the destination directory for the installation and click **Next** to continue, as shown in Figure 7-5.

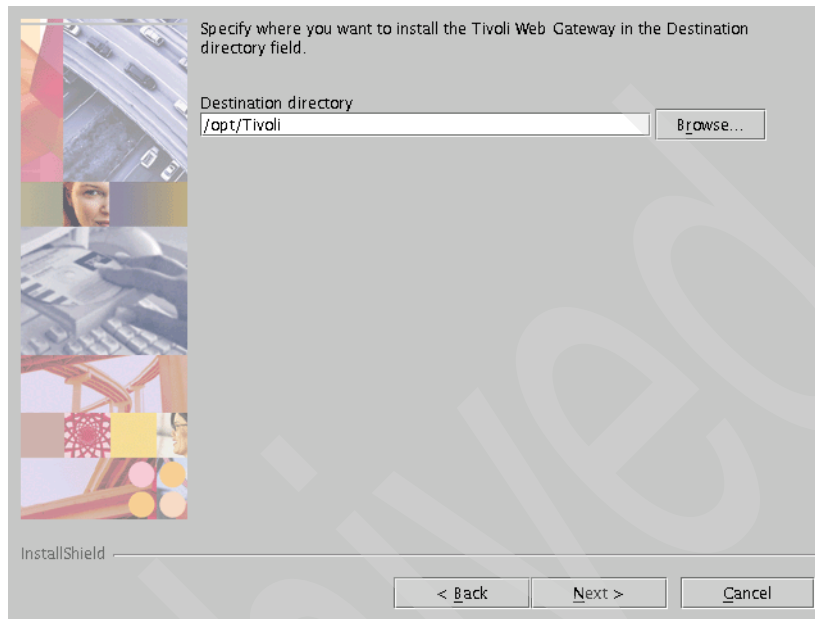


Figure 7-5 Web UI: Destination directory

7. Complete the fields with the database information and click **Next** to continue, as shown in Figure 7-6.

Specify configuration information for the Web Gateway database.

IBM DB2 home directory
/home/db2inst1/sqllib

DB2 administrator name DB2 administrator password

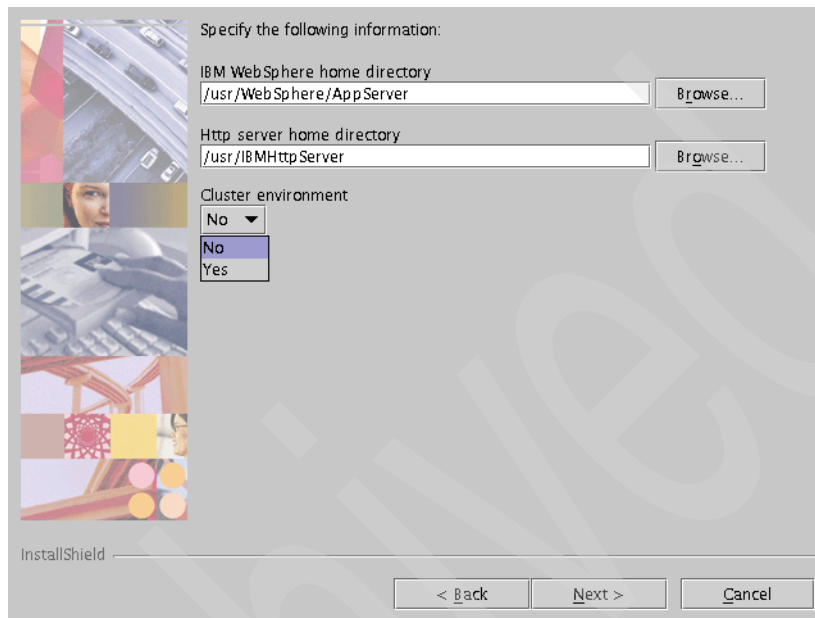
Database user Database user password

DB2 instance name

InstallShield

Figure 7-6 Web UI: Database information

- Complete the fields with the WebSphere and HTTP Server information. Select **No** for the Cluster environment and click **Next** to continue, as shown in Figure 7-7.

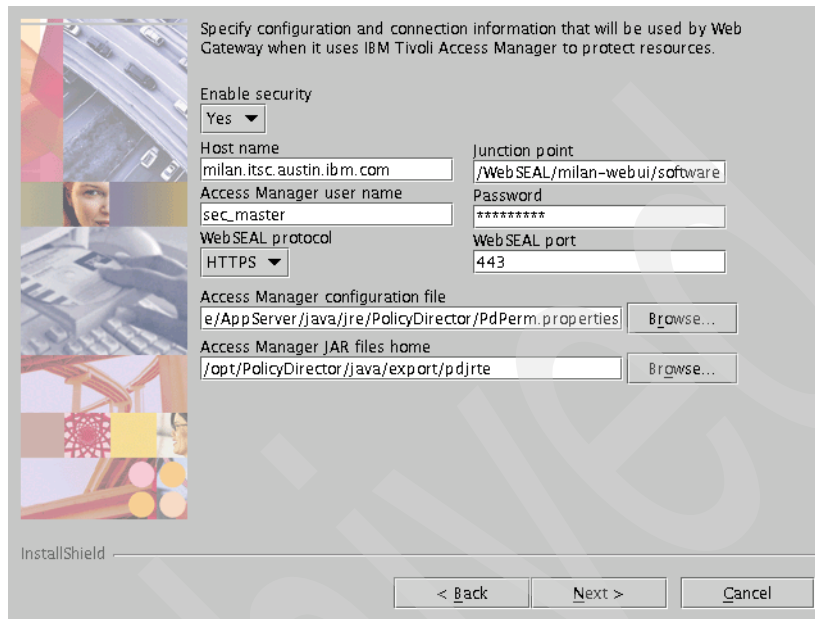


The screenshot shows a configuration window titled "Specify the following information:". It contains two text input fields with "Browse..." buttons: "IBM WebSphere home directory" with the value "/usr/WebSphere/AppServer" and "Http server home directory" with the value "/usr/IBMHttpServer". Below these is a "Cluster environment" section with a dropdown menu showing "No" selected. At the bottom are three buttons: "< Back", "Next >", and "Cancel".

Figure 7-7 Web UI: WebSphere information

Note: The cluster environment is used when you are installing a second Tivoli Web Gateway server to connect the same Tivoli Web Gateway database.

9. Enable the security option, complete the fields with the WebSEAL information, and click **Next** to continue, as shown in Figure 7-8.



Specify configuration and connection information that will be used by Web Gateway when it uses IBM Tivoli Access Manager to protect resources.

Enable security
Yes ▾

Host name
milan.itsc.austin.ibm.com

Access Manager user name
sec_master

WebSEAL protocol
HTTPS ▾

Junction point
/WebSEAL/milan-webui/software

Password

WebSEAL port
443

Access Manager configuration file
e/AppServer/java/jre/PolicyDirector/PdPerm.properties

Access Manager JAR files home
/opt/PolicyDirector/java/export/pdjrte

InstallShield _____

< Back Next > Cancel

Figure 7-8 Web UI: WebSEAL information

Note: Web UI is only supported with security enabled. The security is optional only for Resource Manager.

10. Figure 7-9 shows the actions that will be performed during the installation. Click **Next** to continue.

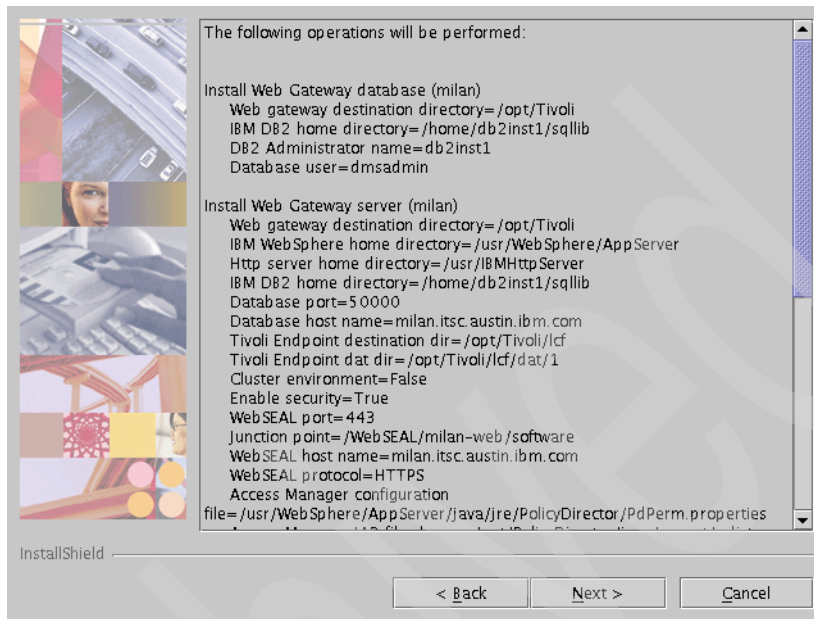


Figure 7-9 Web UI: Actions to be performed during the installation

11. Figure 7-10 shows the installation progress. No action is needed.

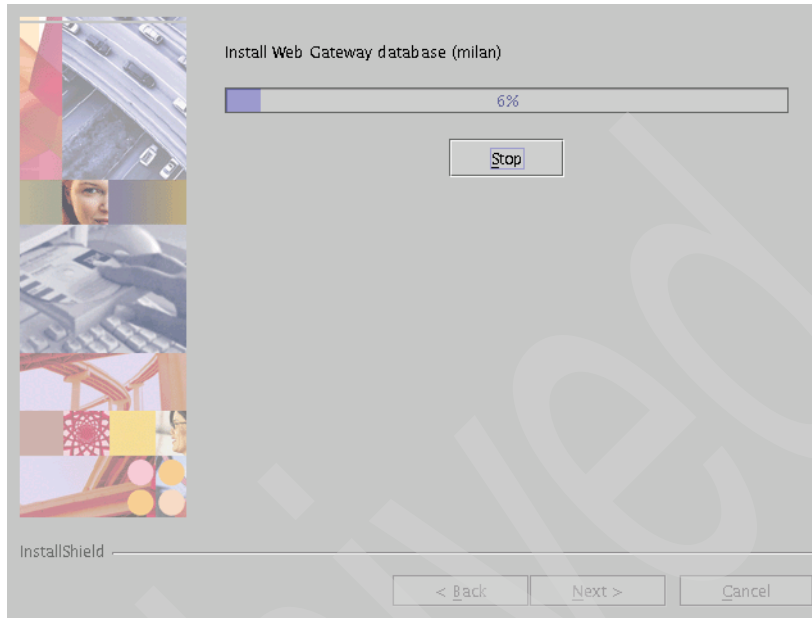


Figure 7-10 Web UI: Installation progress

12. Figure 7-11 on page 306 shows the installation summary. Click **Finish** to terminate the installation.

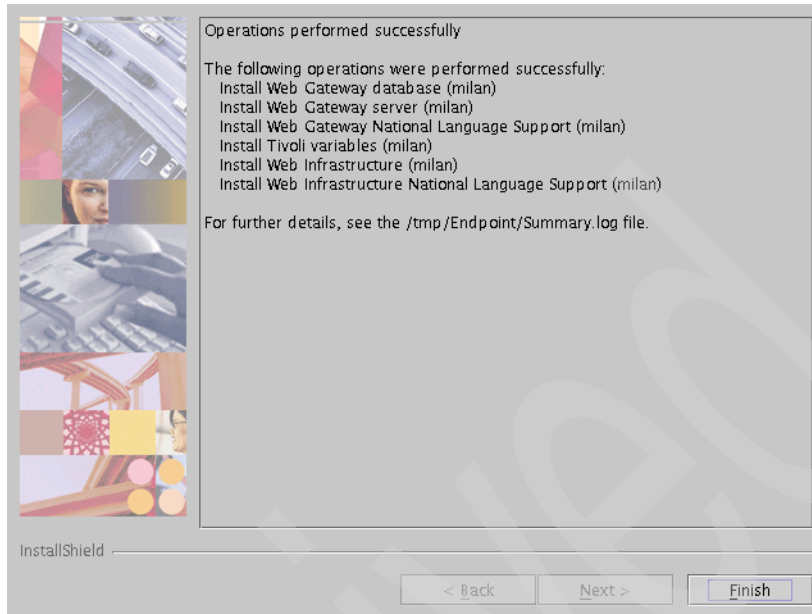


Figure 7-11 Web UI: Installation summary

13. You need to reboot the server to complete the installation. After the reboot, make sure that the IBM HTTP Server and WebSphere applications are up and running.

7.4 Preparing to use the Web UI

Before you start to use the Web Interface, you must perform the following tasks:

1. Assign the WebUI_Admin role.
2. Create users in WebSEAL who will be using the Web UI.
3. Publish inventory and software package Web objects to the Web Gateway.

7.4.1 Assigning the WebUI_Admin role

To publish and unpublish Web objects, you must have the WebUI_Admin role. This is a TMR role, and you must assign it to the root administrator and any other Tivoli administrator that will be performing Web UI functions. This can be done through the Tivoli Desktop by performing the following steps:

1. Open the **Administrators** icon.

2. Right-click the **Administrator** icon.
3. Select **Edit TMR Roles** to assign the WebUI_Admin role to that administrator.

7.4.2 Creating Web UI users

Web UI users need to log in and be authenticated by the WebSEAL server before they can access the Web UI and the Web objects. You must create users in WebSEAL for end users who will be using the Web UI. Example 7-5 shows how to create a user in WebSEAL.

Example 7-5 Web UI: Creating a WebSEAL user

```
pdadmin -a sec_master -p password
user create user1 cn=user1,dc=abbc,dc=com user1 user1 password
user modify user1 account-valid yes
user modify user1 password-valid yes
```

7.4.3 Publishing profiles to the Web Gateway

To publish a software package, inventory scan, or reference model as a Web object, you can use the **wweb** command. The **wweb** command provides access control to users or a group of users with access rights to one or more Web objects.

The published profiles and reference models are stored on the Web server and reflected in the DB2 database. This information is located in the directory C:\IBMHTTP Server\htdocs\twg\user. Here, the CCM directory is for reference models, InventoryWeb is for inventory scans, and SoftwareDistributionWeb is for the software package. You must ensure that there is adequate disk space, because the SoftwareDistributionWeb directory will contain all the published packages.

The format of the **wweb** command is as follows:

```
wweb {-publish|-unpublish} -p publicName -v version {-i all | -i interp[-i
interp]...} -w app_srv_ep_label [-w app_serv_tma]...[-c connspeed] {-u all |-u
user[-u user] ... } [-U file]... [-f|-n] @profile
```

Where:

- publish|-unpublish** Publish or unpublish the profile. Required.
- p publicName** The public name of the published Web object. Required. Use a meaningful name such as /Microsoft/Word, because this will be the name displayed on the Web UI.

-v version	The version of the published Web object. Required. As an extension to the public name example, give it a valid version number such as 2000. The Web UI would display Microsoft as the category with a public name of Word at Version 2000.
-i interp / all	The interps on which the Web object works (-publish operation only).
-w app_srv_ep_label	The labels of the Tivoli managed agents that have the Web Gateway component installed. Required.
-c connspeed	The minimum connection speed suggested for the client to download the profile (-publish operation only).
-u all -u user	Users allowed to access the Web object from the Web, where a user is a WebSEAL account name.
-U file	A file containing the list of users.
-f	Force the Web Gateway depot to load (when publishing) or unload (when unpublishing).
-n	Force the Web Gateway depot to not load (when publishing) or to unload (when unpublishing).
@profile	The name of the profile to be published. The at sign (@) is required.

Example 7-6 shows how to publish and unpublish the software package named example_win.1.0.

Example 7-6 Web UI: wweb command

```
#wweb -publish -p /Windows/example1 -v 1.0 -i all -w milan -u sec_master -u
user1 -f @example_win.1.0
#wweb -unpublish -p /Windows/example1 -v 1.0 -w milan -f @example_win.1.0
```

7.4.4 Starting the Web UI

To start the Web UI, open a browser and enter the following Web address:

<https://hostname/junction/WebConsole/com.tivoli.WebUI.servlets.ConsoleMain>

Where:

hostname	The host name of the machine where WebSEAL has been installed and configured
junction	The name of the junction that has been configured in WebSEAL

In our environment, we used the following command to open the Web UI:

```
https://milan:444/software/WebConsole/com.tivoli.WebUI.servlets.ConsoleMain
```

Note: The port 444 was informed because the instance created on WebSEAL is listening to port 444 for https and not the default port 443.

The Java and JavaScript™ options must be enabled in the Web browser.

The browser Java plug-in must be at a minimal 1.4 version. The latest version can be downloaded from:

<http://java.sun.com/j2se/>

For more information about IBM Tivoli Configuration Manager Web Gateway, see:

- ▶ Chapters 13 and 14 of *IBM Tivoli Configuration Manager Version 4.2.2: User's Guide for Deployment Services*, SC23-4710
- ▶ Chapter 6 of *IBM Tivoli Configuration Manager Version 4.2.2: Planning and Installation Guide*, GC23-4702
- ▶ Chapter 4 of *All About IBM Tivoli Configuration Manager Version 4.2*, SG24-6612

Archived

Tuning and best practices

This chapter contains advanced topics about analyzing and tuning your IBM Tivoli Configuration Manager environment. In this chapter, we assume that readers have in-depth knowledge of Tivoli Configuration Manager features and terminology.

All scripts discussed in this chapter can be found in Appendix B, “Scripts referenced in the book” on page 359. You can also download these scripts from the ITSO FTP site. For download instructions, refer to Appendix C, “Additional material” on page 387.

We discuss the following topics in this chapter:

- ▶ Analyzing your Tivoli Configuration Manager environment
- ▶ Tivoli Management Framework considerations
- ▶ Tivoli Configuration Manager Software Distribution
- ▶ Activity Planner
- ▶ Change Manager
- ▶ Inventory tuning
- ▶ Resource Manager
- ▶ Pristine Manager
- ▶ Tivoli Configuration Manager V4.2.3

8.1 Analyzing your Tivoli Configuration Manager environment

The IBM Tivoli Configuration Manager architecture should enforce a common behavior among all of the management components. During a Tivoli Configuration Manager deployment, you can have two scenarios: fresh install or upgrade. The topics covered in this chapter apply to both scenarios.

8.1.1 Applying fix packs or implementing a new version

In 1.4.2, “Tivoli Configuration Manager V4.2.x product-based releases” on page 14, we summarize each Tivoli Configuration Manager version and the fix pack that was available at the time the newer version was released. For example, patches or fix packs that might apply to Tivoli Configuration Manager V4.2.1 might not be included in Tivoli Configuration Manager V4.2.2 or later, because Tivoli Configuration Manager V4.2.2 was released when Tivoli Configuration Manager V4.2.1 Fix Pack FP01 was just available. Consider this when deciding on whether to upgrade to a later version or to apply just a fix pack. We recommend that you upgrade your environment gradually whenever possible. When upgrading to a later version, consider whether new features or architecture changes in your infrastructure can fulfill your production requirements. It is key to read carefully the *Release Notes* of each Tivoli Configuration Manager version and the *readme* files of each patch or fix pack released.

8.1.2 Upgrading: Gradual or everything at the same time

It is important to notice that deploying the components of each IBM Tivoli Configuration Manager depends greatly on the architecture you are planning to install. For example, you might install the Activity Planner component only and might delay the upgrade of other components.

Important: Special notice should be given to the components that can download methods on your Software Distribution or Inventory targets. Method download can impact your whole installation, because during the first software package profile or inventory profile distributions, the method download will be triggered through the Tivoli Management Framework dependency mechanism. Recovery to a previous state of Tivoli Configuration Manager is much more stressful and complex in this case and it might leave your software distribution targets in an unrecoverable or inconsistent state. We recommend deploying Tivoli Configuration Manager components that do not impact the targets directly. The roll out of the Tivoli Configuration Manager components that can impact your installation should be performed in a scheduled way.

The following components can be installed and allow a smooth rollback independent of your targets or endpoints:

- ▶ Activity Planner
- ▶ Change Manager
- ▶ Resource Manager
- ▶ Pristine Manager
- ▶ Web Gateway

8.1.3 Assessing your Tivoli Configuration Manager environment: Assessment tool

The assessment tool is a Perl program, `assess.pl`, that obtains a snapshot of configuration information about a Tivoli management region (TMR) and formats that data into a set of linked browser (.html) files. Any HTML-3 capable Web browser can then be used to view this information. Environments having multiple, interconnected TMRs can be assessed as well by running the tool on each Tivoli server and gathering the results into a single location.

The `assess.pl` program is a Perl V4 script, intended for use in capturing and assessing significant architectural information related to an installed Tivoli environment. It is compatible with all currently-supported Tivoli product versions.

This is an extremely useful tool that creates a very user-friendly picture of your Tivoli deployment and evaluates the deployment for known configuration issues and common problem symptoms. In the case of problems for which you need to contact IBM Support for assistance, this tool's output can possibly shorten the normal data-gathering phase of problem resolution. It was designed to gather the data that a support engineer needs to get a clear picture of your environment. This is not to say that additional data will not be required, for example, trace data or log files. But standard information, such as product and patch levels and operating system types, is automatically gathered and presented.

Using such a tool plus the information provided in the previous section regarding IBM Tivoli Configuration Manager versions and fix packs can help give you a clear picture on the actions you need to perform in your production environment.

Note: We recommend that you carefully read the *Tivoli Field Guide: Tivoli Assessment Tool (assess.pl) Guidelines for the use of the Assess tool and analysis of results obtained*, which describes the tool installation and functionality in detail, available at:

http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html

8.1.4 Tivoli Configuration Manager upgrade: SD_CMSTATUS_VIEW

The view `SD_CMSTATUS_VIEW` is defined in both Tivoli Configuration Manager V4.1 and Tivoli Configuration Manager V4.2.x to query the table `COMPUTER` and the table where the software package information, endpoint, and the package status is stored.

Consider the case where you need to upgrade from Tivoli Configuration Manager V4.1 to Tivoli Configuration Manager V4.2 or later. Upgrading to Tivoli Configuration Manager V4.2.x implies a change in the Inventory schema. The table `SD_CM_STATUS` in Tivoli Configuration Manager V4.1 changes to `SD_INST` in Tivoli Configuration Manager V4.2, so you would need to plan to move the data from `SD_CM_STATUS` to `SD_INST`. In addition, `SD_INST` has a foreign key constraint on the table `COMPUTER`. In Tivoli Configuration Manager V4.1, we were able to insert an endpoint into the table `SD_CM_STATUS` while that endpoint GUID key was not in the table `COMPUTER`. However, in Tivoli Configuration Manager V4.2.x, the foreign key constraint requires the endpoint to be recorded in the table `COMPUTER`.

8.1.5 Hub-spoke configuration

In a hub-spoke architecture, we recommend that you install the Activity Planner, Change Manager, and Pristine Manager in the hub Tivoli server. Therefore, you would be able to upgrade those components without impacting the rest of your infrastructure.

Consider also having a separate source host in your hub Tivoli server to load your software packages. It has been proven in stress tests that a possible improvement in performance can be obtained by installing a source host on each spoke Tivoli server. However, that configuration would require you to automate your package import process to cover up all the spoke Tivoli servers.

8.1.6 Tivoli Web Gateway upgrade or migration

The Tivoli Configuration Manager V4.2.1 Web Gateway component is different from the Tivoli Configuration Manager V4.2.2 and Tivoli Configuration Manager V4.2.3 Web Gateway component architecture. In fact, if you need to upgrade Tivoli Configuration Manager V4.2.1 Web Gateway, you need to set up a parallel infrastructure or a set of machines for your incoming Tivoli Configuration Manager V4.2.2 Web Gateway using DB2 UDB V8.1 Fix Pack 6 and WebSphere Application Server V5.x. Note that you need to upgrade the Tivoli Configuration Manager Resource Manager and Enterprise Directory Query components in your V4.2.1 Tivoli server.

8.2 Tivoli Management Framework considerations

IBM Tivoli Management Framework is the software infrastructure for many Tivoli Management products. Tivoli Management Framework provides services that are used by the installed Tivoli Enterprise products such as IBM Tivoli Configuration Manager. Tivoli Management Framework is used to manage targets in the enterprise environment, thus its stability is paramount.

We now discuss the factors and issues that affect the stability and performance of your customer's management solutions. Consider the recommendations given to improve your framework stability so that you can avoid scenarios that have been found in the field and might render your framework and management products unusable.

8.2.1 Tuning the operating system: Threads

A Tivoli management region will exhibit any of a variety of symptoms (oserv crashes, **wchkdb** hangs, **wupdate** hangs, fork failures, endpoint manager hangs, or the gateway does not respond) that equal “there aren't enough Tivoli threads configured to handle the workload.” The correct setting for `rpc_max_threads` parameter is especially important when running commands that contact all your managed nodes, such as **wchkdb** or **wgateway**.

The number of remote procedure call (RPC) threads is limited by the number of OS file descriptors. The Tivoli “`rpc_max_threads`” parameter sets the maximum number of Tivoli threads (processes or spawned off child subprocesses) that can be active at one time. In a very busy environment, allowing more `rpc_max_threads` is better. The default is 250. We recommend, for tuning purposes, that you make this value a multiple of what the computer considers 1000, that is, 1024 or 2048 (or at least 512). However, the operating system also has limits on the maximum number of processes or threads that can be running at one time. Therefore, this number must also be tuned.

It makes no sense to set the Tivoli `rpc_max_threads` value to 2048 if the operating system allows a total of only 1024 threads for everything including the Tivoli processes. Sometimes, the command `odadmin set_rpc_max_threads` seems to be unable to set the maximum number of threads to the desired value, and will, at best, only increase it to some lower number. Other symptoms are:

- ▶ Error messages in the `oservlog` such as the following message (although it might be a different number than 250):

```
Apr 23 15:42:44: #RPC Request rejected outstanding threads: 250
```

- ▶ If you figure out that this is a thread resource issue, you need to increase the number of Tivoli threads permitted. But sometimes, you run into the situation where trying to set the `rpc_max_threads` value to some *higher limit* does *not* work:

```
prompt# odadmin get_rpc_max_threads
Maximum number of RPC threads is 250
prompt# odadmin set_rpc_max_threads 2048
prompt# odadmin get_rpc_max_threads
Maximum number of RPC threads is 1014
```

If you encounter this situation, we recommend that you perform an analysis of all your `rpc_thread` settings and an overall analysis of number of the endpoints per gateway and the total number of endpoints being supported by each Tivoli server.

Note: In “gw_tuning.pl” on page 360, we provide a script called `gw_tuning.pl`. Based on best practices, this script detects wrong configurations and recommends the necessary settings of a typical Tivoli server configuration, such as `oserv` threads, job settings per gateway based on the number of endpoints, and endpoint manager threads.

rpc_max_threads controlled by number of file descriptors

In this section, we explain a number of operating system parameters that influence the behavior of your Tivoli environment.

(Parameter) A: [`rpc_max_threads` parameter] is controlled by (parameter) B: [the max # of file descriptors specified by the OS].

On most UNIX-based operating systems:

- ▶ B: [maximum # of file descriptors] is related to C: [max # of processes allowed per user, sometimes called `max_uproc`].
- ▶ C: [max # processes allowed per user `max_uproc`] is based on D:[max # procs allowed on machine, sometimes called `max_proc`]. (These are multiplied by a number and divided by a different number. The multiplying and dividing numbers depend on the operating system).
- ▶ D: [max # procs allowed on machine `max_proc`] is based on E:[max # users] * <some OS-dependent integer> + 64) (where the 64 is the typical default number of “fd” or “file descriptors” each user gets to start with).

Therefore, A (Tivoli `rpc_max_threads` parameter) is based on B (UNIX OS parameter for max fd). We need to increase B. B is based on C (UNIX OS parameter for max user procs `max_uproc`). But C is based on D (UNIX OS parameter for max system procs `max_proc`), and D is based on E (UNIX OS parameter for max # users). Increasing E (max # users), increases D (max

system procs `max_proc`). This also increases C (max user procs `max_uproc`), which in turn, increases B (max # file descriptors). This allows us to increase A (Tivoli `max_rpc_threads`). C (max user processes `max_uproc`) probably needs to be more than 2000 to get the Tivoli `rpc_max_threads` parameter over 2000 (up to 2048).

Type `ulimit -a` to see what OS parameter configurations you have. The output should be similar to that shown in Example 8-1.

Example 8-1 Command output

```
# ulimit -a
time(seconds) unlimited
file(blocks) unlimited
data(kbytes) unlimited
stack(kbytes) unlimited
memory(kbytes) unlimited
coredump(blocks) 2048
nofiles(descriptors) 2000
```

Whether it is listed in the “ulimit” output or not, it is the “nofiles” (number of file descriptors) parameter that you need to increase to be able to set up the `rpc_max_threads` within the Tivoli environment.

8.2.2 Tuning Tivoli threads: Protecting the `oserv`

We highly recommend that child process RPC threads are not greater than 60% of `oserv` RPC threads.

Important: In the case of a Tivoli server, you can have the `oserv` threads, endpoint manager threads, and gateway threads. In such a case, the endpoint manager threads plus the gateway threads should not be greater than 60% of the TMR `oserv` threads. We recommend that you avoid creating a gateway in your Tivoli server, so the `oserv` plus the endpoint manager have more threads available.

Endpoint manager (`max_epmgr_rpc_threads`)

Too many endpoint calls can exhaust this resource quickly. For example, avoid or minimize:

- ▶ Logins (isolation, initial, orphaned)
- ▶ Migrations
- ▶ Long-running policy scripts and w-commands

The endpoint manager `max_epmgr_rpc_threads` variable is restricted or impacted by the availability of `oserv` RPC threads; therefore, exercise some caution when changing this. (See the `wepmgr` command in *Tivoli Management Framework Reference Manual, Version 4.1.1, SC32-0806*.)

8.2.3 Gateway tuning

Each gateway method uses an RPC thread until its type is determined. Each MDist 2 session uses an RPC thread (`rpt <-> app`, `rpt <-> rpt`) that is restricted by the availability of `oserv` RPC threads. The `tmf_dispatch` method use a gateway RPC thread and an `oserv` RPC thread. Other factor that might influence your Tivoli environment is that the `reconnect_thread` is single threaded; therefore, a high volume of TCP connections can overwhelm it. This thread needs to accept() connections before the TCP backlog is full. Symptoms can include:

- ▶ Tivoli Configuration Manager components might trigger fail-over mechanisms due to delays.
- ▶ The Tivoli management agent/endpoint does not record an error because upcall eventually works.
- ▶ A low number of jobs running and waiting in job queue.

Number of endpoints

On each gateway the `JOBQ` threads (`pthreads`)/`max_concurrent_jobs` or TCP backlog not being properly set might result in results in requests being queued or in isolation logins that will impact the endpoint manager, thus the overall Tivoli infrastructure.

We recommend that you configure your gateways to handle the expected workload.

Configure the following settings:

- ▶ Increase the gateway `max_concurrent_jobs` > number of endpoints.
- ▶ `rpc_maxthreads = (max_concurrent_jobs+max_concurrent_logins)*15%`.
- ▶ Use `logstatus` to monitor workload (set `logstatus_interval = 300`):
 - 750-1000 endpoints per gateway (Windows)
 - 1500-2000 endpoints per gateway (UNIX)

We have seen production environments with managed endpoint numbers above these. However, be cautious when dealing with mixed environments where you might find massive logins, machines that are heavily monitored, or gateways that need to perform large software distributions or inventory scans simultaneously.

Number of TCP connections

During a TCP connection request flood, the TCP backlog might get full so that connection attempts are never seen by the gateway.

You can get the value by running (where <GW0ID> is the object ID returned by **wgateway** command):

```
# idlattr -tg <GW0ID> tcp_backlog short
```

You can set it by using:

```
# idlattr -ts <GW0ID> tcp_backlog short <new_value>
```

TCP backlog: The tcpBackLog directive controls the TCP “backlog queue” when listening for connections in stand-alone mode. It has no effect on servers in inetd mode. When a TCP connection is established by the TCP/IP stack inside the kernel, there is a short period of time between the actual establishment of the connection and the acceptance of the connection by a user-space program. The duration of this latency period is widely variable and can depend on several factors (hardware, system load, and so on). During this period, TCP connections cannot be accepted, because the port that was previously “listening” has become busy with the new connection. Under heavy connection load, this can result in occasional (or even frequent!) “connection refused” messages returned to the incoming client, even when there is a service available to handle requests. To eliminate this problem, most modern TCP/IP stacks implement a “backlog queue,” which is simply a preallocation of resources necessary to handle backlog-size connections during the latency period. The larger the backlog queue, the more connections can be established in a very short time period. The trade-off, of course, is kernel memory or other kernel resources. Generally, it is not necessary to use a tcpBackLog directive unless you intend to service a large number of hosts or have a consistently heavy system load. If you begin to notice or hear of “connection refused” messages from remote clients, try setting a slightly higher value to this directive.

8.2.4 Network tuning parameters

You might also need to tune your networking configuration on your Tivoli servers and gateways. The networking configuration especially should be tuned for the Tivoli infrastructure servers that need to protect against network attacks.

AIX 5L

Disable some various forms of denial of service attacks. While this is not going to net a performance boost, it will keep people from sapping your performance and availability later:

```
/usr/sbin/no -o clean_partial_conns=1
```

The following command specifies whether or not SYN (synchronize the sequence number) attacks are being avoided. If on, it randomly removes partial connections to make room for new non-attack connections.

```
# Socket queue defense against SYN attacks/usr/sbin/no -o clean_partial_conns=1
/usr/sbin/no -a or no -o somaxconn (Outstanding Connection requests)
/usr/sbin/no -o somaxconn=NewValue (default = 1024)
```

This change takes effect immediately. The change is effective until the next reboot. To make the change permanent, add a **no** command to **/etc/rc.net**.

Solaris

For Sun™ Solaris™, use:

```
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q 1024
```

The q queue holds sockets awaiting an accept() call from the application:

```
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 2048
```

The q0 queue contains half-open sockets.

HP-UX

For HP-UX, use:

```
/usr/sbin/ndd -set tcp_syn_rcvd_max 1024
/usr/sbin/ndd -set tcp_conn_request_max 200
```

8.2.5 Endpoint login storms

Typically in a heavy loaded or not properly configured Tivoli environment, we find some indicators, such as:

- ▶ Gateways queue logins/upcalls.
- ▶ All endpoints login at the same time, for example, 8:00 a.m. on Monday morning.
- ▶ Gateway unavailable, down: Results in an Isolation login to an alternate gateway and is unmanageable until it completes the login.
- ▶ Busy endpoint manager: Results in queued requests.
- ▶ Software Distribution or Inventory delays.

- ▶ Software Distribution or Inventory retries.
- ▶ Endpoints become isolated.
- ▶ The gateway takes too long to respond.
- ▶ Management of servers versus the management of desktops: Usually, desktops are rebooted daily and will perform a normal login unless their gateway cannot handle the workload. If you need to avoid login storms, you can redirect or migrate your desktop endpoints to a different Tivoli server.

These are typical symptoms of a poorly configured Tivoli environment or lack of resources to support the workload. Usually, the endpoint manager fails under the load. All upcall methods will now fail, and all endpoints can now become isolated as isolated login requests have to be handled by the endpoint manager and each gateway forwards endpoints to it. The `max_allow`, `max_sgp`, and `max_after` parameters do not solve the problem. You might also notice that running the command `odstat -av | grep encrypted` will indicate that the method `endpoint_login_encrypted` accumulates at the endpoint manager. The problem is even worse if one gateway fails or is unavailable.

In this scenario, the Tivoli environment is unmanageable, and it might take hours to recover, so you might have now escalated the problem. Sometimes, manual intervention to recover the scenario is needed.

You might also need to check whether:

- ▶ The endpoint `udp_interval/login_timeout` is too short.
- ▶ The `Login_policy` script is too complex, takes too long to run.
- ▶ The problem is related to endpoint manager RPC thread usage. If it is, increase the number of `max_epmgr_rpc_threads`. (You need to increase `oserv's rpc_max_threads` so that it is greater than the `max_epmgr_rpc_threads`. Also, make sure that OS file descriptor limit is greater than the `rpc_max_threads`.)

Tip: Before reaching a login storm, we recommend that you monitor or “survey” the behavior of your endpoints. In “find_rogue” on page 370, we provide the script called find_rogue.pl that performs an analysis of the Tivoli server endpoint manager to provide “rogue” endpoints that should be investigated before reaching any login storm or crisis scenario. The script find_rogue.pl generates a report analyzing the following topics:

- ▶ Endpoints ordinarily make only *one* initial login attempt: Multiple attempts, therefore, represent a potential problem, the script records *only* multiple attempts.
- ▶ Migration legions occur when an endpoint is moved to a different gateway. Ordinarily, only one login attempt should be needed. Multiple attempts from one endpoint need to be investigated. *Only* multiple attempts are included in the script report.
- ▶ Endpoints perform an *isolation* login when they cannot contact their primary gateway. This activity is reported.
- ▶ Orphaned endpoints seem to be having a problem as evidenced by multiple isolation login attempts reported by the script.
- ▶ A dispatcher number is unique to one host. Multiple IP addresses attempting to use the same dispatcher are reported by the script.

As a reference for you, the following list provides the thread usage information for each type of endpoint login:

- ▶ Migration:
 - Avoid migrating endpoints to specific gateways.
 - Each migration command uses a total of 12 epmgr threads.
 - Use gateway clouds/farms where possible.
- ▶ Isolation:
 - Isolation logins use seven epmgr threads.
 - Impatient endpoints exacerbate the problem.
 - Make udp_interval/login_interval reasonable (default).
- ▶ Orphaned:
 - Orphaned logins use seven epmgr threads.
- ▶ Initial:
 - Initial logins use eight epmgr threads.
 - Do not be too aggressive with the rollout of new endpoints.

- ▶ Thread usage of other operations:
 - The CLI `wep status` uses six `epmgr` threads for security.
 - The CLI `wde1ep` uses seven threads, plus one thread for each profile manager.

8.2.6 Protecting the endpoint manager

In this section, we review two scenarios that can help to protect the endpoint manager from excessive logins.

Tivoli Configuration Manager Framework level earlier than V4.1.1

In this case, we need to protect the endpoint manager by disabling endpoint initial, isolated, or orphaned logins through the `allow_install_policy` and `select_gateway_policy` parameters. Under a crisis scenario, such as a login storm where the endpoint manager is being affected, we provide a temporary solution that might help to alleviate the problem of too many isolated logins.

The script `protect_epmgr.sh` performs the following steps:

1. Detects the number of endpoint login encrypted methods active.
2. If it is over a certain number (60), it loads `allow_install_policy` and `select_gateway_policy` that return “1” (exit 1) and rejects endpoint logins (initial, isolated, orphaned).
3. When the number of encrypted logins is under a specific number (30), the original `allow_install_policy` and `select_gateway_policy` are set back.

Using this approach enables the endpoint manager to process the current endpoint logins being forwarded by the gateways. The `protect_epmgr.sh` script is in the “`protect_epmgr.sh`” on page 374.

Tivoli Configuration Manager Framework V4.1.1

We recommend that you install Tivoli Management Framework Version 4.1.1. The new features available to manage the endpoint manager also protect it from heavy login storms.

If you use these features, the endpoint manager will inform the gateway when it is busy and will dedicate its resources to manage the current workload.

Endpoint manager “busy” actions are:

- ▶ All “login threads” are in use.
- ▶ The gateway sends a “-1” timeout to the endpoint.

- ▶ The endpoint adjusts its login_timeout by a random amount: A value between n and 2n.
- ▶ The gateway maintains a queue of login requests.
- ▶ The gateway tells endpoint how long a login will take.
- ▶ The gateway sends this timeout value to the endpoint.

Note:

To set the maximum concurrent endpoint logins in Framework V4.1.1:

- ▶ Endpoint manager Framework V4.1.1:
 - Command line:
wepmgr login_limit <NUM>
 - The default is 80% of max_epmgr_rpc_threads.
 - “0” disallows logins.
- ▶ Gateway:
 - Command line:
wgateway <gwlabel> max_concurrent_logins <NUM>
 - The default is 100 (minimum).
 - The maximum is 500.

8.2.7 Gateway debug level not verbose

In the field, we have verified that setting the gateway debug level to more than 2 (harmless) can impact the performance of your gateways adversely. This might lead to isolated endpoints or endpoints failing over to the alternate gateway.

When possible, we recommend that you keep this setting (using `wgateway xxx set_debug_level`) at the minimum level.

8.2.8 Slow links: Method download

In your pristine installation, when possible, include the methods that are downloaded through the Framework dependency mechanism. In order to check the dependencies of each product, you need to:

1. Capture the dependency OID using the `wlookup -ar DependencyMgr` command.
2. Run the `wdepset -v @DependencyMgr` command to view the dependency set.

Tip: We provide the script `metodos.pl` to capture the methods downloaded to an endpoint. In order to capture the methods correctly, the following steps are performed:

1. The endpoint logs in to a gateway that has the proper Tivoli Configuration Manager product levels.
2. Distribute a profile (software package, inventory) to the endpoint. This process will automatically download all the necessary methods to the endpoint.
3. Run the `metodos.pl` script on the machine. This script reads the file `index.V5` and generates a ZIP file containing all the binaries that can be preloaded in your endpoint pristine installation under the directory where the endpoint was installed.
4. Stop the endpoint and unzip the ZIP file generated by this tool.

In Framework V4.1.1, a new feature enables you to control the download of methods through MDist 2 to keep the network from becoming overloaded.

8.2.9 Monitor disk/file system space and transaction log size

To avoid `odb.bdb` corruption and `oserv` instability, we recommend that you monitor the `$DBDIR` the size of `odb.log`.

Transaction log size: `Odb.log` size does not change

The size of the `odb.log` should fluctuate, growing and shrinking, sometimes even reaching to zero (although you might not be quite fast enough to see it at that size, when, for example, you run an `1s` command on UNIX workstations), and growing again, as requests, (transactions) are requested, submitted, and handled. There is no one size fits all number for the size of the `odb.log`. Really busy environments will show much greater `odb.log` sizes reached than quieter environments.

What you want to look for is an `odb.log` that does *not* change size (which means transactions are hung), or one that just seems to grow continuously and is “too big” (relative to what you are used to seeing).

Sometimes, patience will cure the problem, and the system will slowly process all transactions that are queued up. But if the environment is busy, things might get bogged down and just keep having more piled up. In that case, you might have to take explicit actions to unclog things and clear hung transactions. You might see transactions locks when running the `tmstat` command. Avoid stopping the `oserv` process if `odb.log` is too large.

8.2.10 Framework V4.1.1: Health check parameters

In Framework V4.1.1, the gateways periodically check the health of their assigned endpoints. To check the endpoint health, the gateway creates an endpoint health job queue, which spawns 20 job threads at a time. Therefore, as many as 20 endpoints are health checked concurrently per gateway. Currently, the session timeout set for these threads is 10 seconds so that gateway performance is not impacted. This timeout value might cause problems if the endpoint is too busy or is connected on a slow link and does not respond back in 10 seconds. This results in the endpoint status being set to unreachable. The current endpoint health check implementation also controls downcalls to the endpoint based on endpoint health. If endpoint status is unreachable (for example, due to intermittent networking problems or a busy endpoint), the gateway does not attempt to downcall.

To address this, two gateway attributes, which can be set with the **wgateway** command, have been added in 4.1.1-TMF-0010 to control the endpoint health check session timeout and to control downcall behavior based on endpoint status:

- ▶ **epcheck_sess_timeout**: Specifies the session timeout used for the endpoint health check session. This can be set to values in the range from 10 (default) to 600 seconds. This can also be set to 0 to cause the endpoint health check session to use the current gateway session_timeout.
- ▶ **downcall_limit**: Specifies whether the gateway checks endpoint status before initiating a downcall to the endpoint. If downcall_limit is set to 0 (default), the gateway will attempt downcalls even if the endpoint status is unavailable or unreachable. That is, the gateway will not block any downcall based on endpoint status. If downcall_limit is greater than 0, the gateway will block downcalls based on endpoint status after the “downcall_limit” number of downcall failures.

Important:

- ▶ In the base Framework V4.1.1 code, it is not possible to turn off the health check. The normal way of setting the interval is `wgateway <gw> epcheck_interval <value>`. The value must be within the range of 300-14400. The following `idlattr` can be used to set it to a larger value, but it is still not possible to deactivate it altogether by setting it to 0:

```
idlattr -ts <Gateway-0ID> epcheck_interval long <value>
```

You have to restart the gateway for this value to take effect when using the `idlattr` method of setting it. The value can be set to 86400 (1 day) or 604800 (1 week), for example.

- ▶ We recommend setting the value of `epcheck_interval` to a larger value, because it might impact your distributions. In addition to slow links, you might need to increase the value of `epcheck_sess_timeout` from 10 to a larger value depending on the available bandwidth on your network.

If you do not apply those settings, you might find the scenario shown in Example 8-2 was captured from a gateway log.

Example 8-2 Gateway log

```
2005/01/22 20:59:13 6 2059e9b8: idmap: user ($root_user,w32-ix86) ->
BuiltinNTAdministrator
2005/01/22 20:59:13 6 2059e9b8: idmap: group ($root_group,w32-ix86) -> root
2005/01/22 20:59:13 8 2059e9b8: check_ephealth: od=1257
2005/01/22 20:59:13 8 2059e9b8: check_ephealth: skip: od=1257 status=3
2005/01/22 20:59:13 9 2059e9b8: mdist: execute_method_on_endpoint error,
reason -
2005/01/22 20:59:13 2 2059e9b8: mdist: Job (send) exception -
receiver(1800000002.1257.522+) message(
2005/01/22 20:59:13 9 2059e9b8: mdist: sending_job_run ended:
id=[18556553961106423629:1800000002.1257.522+].
2005/01/22 20:59:13 8 2059e9b8: mdist: processing send job completion.
2005/01/22 20:59:13 6 2059e9b8: mdist: Job UNAVAILABLE - receiver
(1800000002.1257.522+).
```

We can easily detect that the software distribution through MDist 2 “`sending_job_run`” finds the endpoint unavailable due to the endpoint health check performed and the send method is completed.

Note: Endpoint health status error codes are:

- ▶ 0: File permission error, cannot create temporary file.
- ▶ 1: Insufficient disk space in temporary directory.
- ▶ 2: File permission error, cannot create/update method cache.
- ▶ 3: Insufficient disk space in LCF_CACHEDIR.
- ▶ 4: Cannot generate token for tmersrvd account.
- ▶ 5: Cannot generate token for built-in administrator account.
- ▶ 6: Cannot spawn a process.

8.2.11 Framework V4.1.1: Endpoint method download control

In previous versions of Framework, method/dependency downloads were not bandwidth controlled, and method/dependency downloads might flood the slow links. A workaround was to preload the cache directory of the endpoint. Method/dependency downloads can now use the MDist 2 bandwidth, as follows:

```
wgateway <gwlabel> mcache_bwcontrol TRUE
```

The default is FALSE.

Even though you have this feature available, there are scenarios where you might need to preload your methods in order to speed up your distributions.

8.2.12 Framework 4.1.1-TMF-0010: TCP backlog

In Framework V4.1.1-TMF-0010, the TCP backlog is tunable through the command line, as follows:

```
wgateway <gwlabel> tcp_backlog <value>
```

This value can be set in the range of 5-65535.

8.2.13 Framework 4.1.1-TMF-0010: Updated STATUS DATA in gatelog

The status of several indicators are displayed under STATUS DATA in the gatelog, such as:

```
STATUS DATA: jobqq=0 jobqr=3 redyq=0 gwmethods=0 loginq=0 loginr=0 ephq=0  
ephr=0 sendq=0 recvq=1 high=0 med=0 low=0
```


The following list describes the parameters involved:

- ▶ THREAD USAGE
 - jobqq: JOBQ threads in queue
 - jobqr: JOBQ threads running
 - redyq: Reader_threads in queue
 - gwmethods: Gateway methods/RPC threads running
- ▶ LOGIN QUEUE
 - loginq: Logins requests in login queue
 - loginr: Logins running
- ▶ HEALTH CHECK
 - ephq: Endpoint health checks in queue
 - ephr: Endpoint health checks running (maximum of 20)
- ▶ MDist 2
 - sendq: Jobs in send queue on gateway
 - recvq: Jobs in receive queue on gateway
 - high: Number of high sessions in use on gateway
 - med: Number of medium sessions in use on gateway
 - low: Number of low sessions in use on gateway

8.3 Tivoli Configuration Manager Software Distribution

In this section, we review the tunable parameters in the MDist 2 component and the notification manager that might improve or avoid bottlenecks in your software distribution performance.

8.3.1 Adjusting your Software Distribution configuration

Software distributions are submitted to the source host, which in turn, obtains the routing information from the repeater manager in the Tivoli server through the distribution manager. (The distribution manager also runs in the Tivoli server.) Software Distribution reports are stored in the Tivoli server through the distribution manager. The Software Distribution notification manager running in the Tivoli server reads the distribution manager report queue and periodically updates the Inventory RDBMS. Therefore, we advise that you adjust your

Software Distribution and MDist 2 parameters according to your software distribution needs. Usually, you will need to track the following information:

- ▶ How many targets you will need to distribute software to simultaneously
- ▶ The overall number of targets to which you distribute
- ▶ The maximum number of targets to which you distribute software through the same gateway

Be aware that other factors might also slow down the performance of your software distributions, such as:

- ▶ Massive endpoint logins
- ▶ Endpoint upcalls due to other activities such as monitoring
- ▶ Endpoint downcalls due to Inventory scans, and so on

8.3.2 Monitoring your Software Distribution report flow

Typically, you will notice a symptom such as the report processing gets stuck or incoming reports get stuck for unknown or not obvious reasons. The inbound reports flow back using the same path, that is, the repeater hierarchy, as is taken for the outbound distribution. Therefore, the reports flow back through the source host to the TMR, which affects the available sessions and report thread settings.

When the Software Distribution server stops processing incoming reports, perform the following checks to understand the reason:

1. Execute the `odstat -c` command on the Tivoli server to verify that there are no child methods in `rwait` state for the method `run_nm`, which implements the Software Distribution notification manager component.

Important: Only one `run_nm` method can be active; therefore, if a child method gets stuck, the system is not able to process incoming reports.

2. Execute the `odstat -c` command on the Tivoli server to verify that there are no child methods in `rwait` state for method `mdist2_results`, which is invoked by the source host resulting from an incoming report that is executed on the Tivoli server.

Important: If a child method is stuck, other `mdist2_result` methods can still execute, up to the maximum value of the `report_threads_limit` defined in the `swdis.ini` file, or through the `wswdcfg -s` command on the TMR Software Distribution server.

3. Verify that each repeater involved with the reports (Software Distribution results) are active and available by executing the `wping <ManagedNode_Name>` command as identified in `wmdist -I <Source_Host_MN>` command; the OID of the MN/GWY-RPTR is displayed in the “Internal Id:” field. If operating in an interconnected environment, try the `wping` command from both the hub and the spoke Tivoli regions.

8.3.3 Software Distribution high, medium, and low session tuning

The important thing when setting these parameters is getting the balance right. Reducing `notify_interval` will cause MDist 2 to return results more frequently; this triggers the `mdist_result` method to be run on the Tivoli server. However, the number of concurrent `mdist2_result` threads is controlled by the `report_threads_limit`, and if this limit is reached, the `wmdist -I <repeater>` command can show results jobs hanging, waiting to run the method. If you increase the `report_threads_limit` parameter, more of the `mdist2_result` threads can run, and these threads will process the results and create files in the `swdis/work/message` directory for the Software Distribution `notification_manager` (`run_nm`) method to process.

Important: Note that the `mdist2_result` methods (multi-threaded) and the `run_nm` method (single threaded) are handled by the same single process (`swdmgr`) on the Tivoli server. So, the more `mdist2_result` threads there are, the less processing time notification manager gets. Then, you get a build up of files waiting to be processed in the `$BINDIR/./swdis/work/message` directory. This must be tailored to the work profile for each individual environment, but it is possible that increasing an individual value can reduce performance with regards to Software Distribution results.

If you suspect that the source host is out of ports due to a large number of distributions, raise the maximum sessions parameter in MDist 2.

The `report_thread_limit` parameter identifies the maximum number of threads the Software Distribution server can use for processing reports. Consider that each report flowing from the repeater to Software Distribution will use one of the sessions defined in the `max_sessions` parameter (of the available connections). Thus, having the `report_threads_limit` equal to or greater than the sum of the `max_sessions` parameters (for example, 5 high + 10 medium + 40 low = 55, then `report_thread_limit` should be set to 60 or greater) should prevent any report to be rejected, because all threads are in use. If there are sufficient threads to handle all the incoming reports from repeaters, it is also necessary to analyze the flow of messages. Set the `report_thread_limit` parameter by using the `wswdcfg -s report_thread_limit=60` command.

Important: Consider increasing the thread limiter (`report_thread_limit`) to greater than the sum of all high, medium, and low sessions defined in the source host. When sessions are exhausted, the Software Distribution manager throws an exception and messages in the queue go into INTERRUPTED state. The reason for this is that distributions and reporting use MDist 2 sessions and the `report_thread_limit` parameter controls the number of `mdist2_result` methods that run from the source host on your Tivoli server.

8.3.4 Notification manager report processing timeout

From the Software Distribution perspective, if you observe that message files are growing in the Tivoli server's `$BINDIR/./swdis/work/` directory structure, try to reduce the value of the `nm_restart_timeout` parameter (default is 180 seconds), which is the timeout used by notification manager to check for new messages. Use the `wswdcfg -s nm_restart_timeout=90` command to set it. You can also monitor the messages in the notification manager's queue by using the `wmsgbrowse -a` command.

Note that reducing the notification manager restart timeout might increase the database insertion in the Inventory RDBMS.

8.3.5 Database: Too many locks on the SD_CM_STATUS table

The following parameters were added in 4.1-COU-0104 to customize the number of transactions after which a commit operation is performed:

- ▶ `report_transaction_timeout` (default 180 seconds)
Influences the database write operation only during the reporting phase.
- ▶ `max_num_transactions` (default 200)
Influences the database write operation both during the submit and reporting phases.

A commit operation is performed every time the minimum value among the two is reached. So, values of these two tuning parameters have to be chosen, making a trade off between locks on the DB and performance. To minimize locks on the `SD_CM_STATUS` table (or `SD_INST` table for Software Distribution V4.2.x) of the Inventory configuration repository database and historical database during Tivoli Configuration Manager product tests, the following values have been set:

- ▶ `max_num_transactions=5`
- ▶ `report_transaction_timeout=180`

With these settings, we successfully tested the distribution of a software package to about 5000 endpoints.

These parameters can be set using the `wswdcfg -s` command or including them in the `swdis.ini` file in your Tivoli server.

8.3.6 Protect the source host: `notify_interval`, `conn_retry_interval`

In order to protect the source host from being flooded by reports, we recommend that you adjust the following two MDist 2 parameters using the `wmdist -s` command:

▶ `conn_retry_interval`

Increase the `conn_retry_interval` value on the source host to prevent the TMR from being flooded with `mdist2_result` method requests. If there are too many, they will be rejected, because the limit in the `report_threads_limit` threshold has been reached. We recommend that you start with a value of 600 seconds for `conn_retry_interval`.

▶ `notify_interval`

From the Framework MDist 2 perspective, on the repeater, the messages are buffered and sent when the `notify_interval` timeout expires. Therefore, if you observe that too few messages are coming back from the repeaters, try to decrease the `notify_interval` (default is 30 minutes). If an MDist 2 distribution transaction is completed, the report flows back using the same path, that is, the repeater hierarchy, as it did for the outbound distribution. Therefore, it might happen that a distribution report or result reaches the notification manager queue *before* the `notify_interval` expires for each gateway or repeater. Such a scenario can happen with distributions of small software packages or high-bandwidth links.

Important: We recommend that you do not set the MDist 2 parameter `notify_interval` to too short a time. Approximately 10 minutes is an acceptable trade-off. Consider setting slightly different `notify_interval` values per gateway so that reports are not sent up at the same time.

If one endpoint out of the multiple endpoints involved in a single MDist 2 distribution is in an interrupted or unavailable status, the distribution is not completed. In such a case, the distribution report will not be sent to the notification manager until the `notify_interval` expires at the gateway, where the report is bundled for all the endpoints that completed the distribution except for the failing one.

8.3.7 Distribution deadline and retry_ep_cutoff

If the endpoint remains in an unavailable or interrupted status, the gateway will try to contact the endpoint every `conn_retry_interval` until the `retry_ep_cutoff` is reached. In this case, the distribution continues to be in the MDist 2 queue, but it is ignored by the distribution manager until the endpoint logs in again or the distribution reaches the deadline.

8.3.8 Machine hangs while running a script or during reboot

There might be a situation (for example, a commit with automatic reboot or reboot if necessary) when the machine does not respond to the Windows shutdown command. In other scenarios, a script can be in a loop that does not timeout. In those cases, the distribution process `spd_eng` in the endpoint waits for the script to exit or the reboot to be performed (`spd_eng`). In that scenario, the following behavior is expected:

- ▶ The `execute_timeout` or `send_timeout` expires, so the endpoint status becomes interrupted.
- ▶ The gateway contacts the endpoint every `conn_retry_interval`, and a second `spd_eng` is triggered in the endpoint; however, it exits as the original `spd_eng` is still running.
- ▶ The gateway will keep contacting the endpoint until `retry_ep_cutoff` is reached. After that, the distribution is kept in interrupted status until the distribution deadline is reached or the endpoints logs in again.
- ▶ However, because the gateway `retry_ep_cutoff` was expired, the process `spd_eng` cannot return the result of the distribution to the gateway, because the endpoint now is disconnected.
- ▶ If the endpoint is rebooted or the script finished after the `retry_ep_cutoff` expires, the local catalog is properly updated, *but* the table `SD_INST` will not be updated in your Inventory database.

Tip: Increase the value of `retry_ep_cutoff` to a number that ensures that a reboot will be performed or a script times out. For example, you might need to increase it to 8 hours so that the Software Distribution operators notice the machine is hung and it should be rebooted before those 8 hours expire.

8.3.9 Force reboot type during commit operations

The reboot using the commit operation can be forced using the variable `__COMMIT_REBOOT_FORCED__=YES` in `swdis.var` file on target endpoint.

The file `swdis.var` can also contain the variable `__COMMIT_REBOOT_TIMEOUT__=XXX`, where `XXX` represents the seconds to specify the timeout.

The file `swdis.var` path is set up in the endpoint `swdis.ini` file. The fix for this APAR was included in 4.2.1-SWD-0020LA and Tivoli Configuration Manager V4.2.1 Fix Pack FP02.

Commit with reboot if necessary (`wcommtsp -o`)

There might be the case where, before running a commit with a reboot, a set of actions must be performed (for example, shut down a database or stop a service). In the case of a commit with reboot if necessary, we cannot perform any action before the reboot because we cannot detect whether the reboot would take place or not. Therefore, there is no mechanism to ensure that the action takes place *only* when the reboot would be triggered when a commit with reboot if necessary is triggered. This feature will be supported in IBM Tivoli Configuration Manager V4.2.3 FP01.

8.3.10 Software Distribution architecture hints and tips

The following list provides additional recommendations for the Software Distribution component gathered from field experience:

- ▶ When possible, set up your database server and notification manager in different machines.
- ▶ Set up the log host on the same machine of notification manager, but ensure that file systems are different.
- ▶ When possible, ensure that source host is in a separate box than notification manager (Tivoli server).
- ▶ Disable Software Distribution error reporting on MDist 2 RDBMS, Tivoli Enterprise Console integration, signature integration, and database integrations if not needed.
- ▶ Avoid pass-through remote lookup.
- ▶ Reorganize the `COMPUTER` and `SD_INST` table indexes periodically.
- ▶ Use the lenient option when possible for your software packages.
- ▶ Limit your user program's stdout/stderr size in your software package.
- ▶ If you run scripts on any action, remember to define a timeout.
- ▶ Disable versioning and dependency checking at submission when not needed.

- ▶ Have a queue indicator for notification manager. When the queue grows too large, verify the tuning parameters specified in the previous sections and take corrective actions.
- ▶ Have the local source host defined at the spoke Tivoli servers when possible.

8.3.11 Catalog recovery: SPE disconnected commands

We strongly recommend that you load the disconnected commands included with the Software Package Editor (SPE) onto your production endpoints. This will help you debug problems. For example, you can run the `wd1ssp` command to list all the software packages and their status on the local endpoint catalog. If not, you need to run the `wsyncsp` command to be able to track the result of your distributions on your local endpoint catalog through the `sd_inst` Inventory table.

Note that the `epsp.cat` file gathers all the software package status information. You can use this file to add, delete, or simulate a package installation using the disconnected command line. Consider this when having to install dummy packages in order to replicate a production catalog.

Tip: We included an *as-is* script, `gen_cat.sh`, to be able to add or recreate a catalog with a number of dummy entries in “`gen_cat.sh`” on page 383.

8.3.12 Servers versus desktops

We strongly recommend that you split the management of those machines that perform massive daily logins to a gateway from those that do not. Be aware that the factors that can influence your gateway resource exhaustion might be increased when the gateway needs to dedicate resources to manage endpoint logins during morning time frames.

8.4 Activity Planner

Activity Planner enables you to define a group of activities in an activity plan, submit the plan to be executed, and monitor the execution of the plan. Operations include Tivoli Management Framework task library tasks, Software Distribution, Inventory, and Pristine Manager, if the corresponding plug-ins are installed. In this section, we discuss some best practices and tuning parameters that were introduced in Tivoli Configuration Manager V4.2 Fix Pack FP02 and Tivoli Configuration Manager V4.2.1 GA. Additional performance enhancements were introduced by Tivoli Configuration Manager V4.2.1 Fix Pack FP01.

8.4.1 Plan Monitor submission tuning: Caching endpoint information

Activity Planner Monitor validates the endpoints when an activity plan is submitted by using:

- ▶ Name registry: A cache is used if available.
- ▶ Endpoint manager.
- ▶ Endpoints are validated with blocks of 500 elements. If one of the endpoints is not valid, the validation is switched to one endpoint search.

There are consequences in performance for the overall submission time. Remote lookup for endpoints at submission time might cause a general slowdown of the process. Therefore, the following parameters were introduced to enable endpoint caching:

- ▶ `cache_local_target_info`

If you enable this parameter, targets are evaluated only once at plan submission. This parameter applies only to plans in which the targets are specified at plan level and are resolved at plan submission. (The default is yes.)

- ▶ `cache_global_target_info`

This cache is unique in the Activity Planner engine and its data is shared between all active plans. Enable this cache if you have many plans in which targets are specified at the activity level or resolved at activity execution, because the local-target-cache cannot be used in these cases. (The default is no.)

- ▶ `TMF_object_cache`

This contains the relationship between the endpoint object ID and endpoint label. It is activated when the number of targets is greater than *pre_loading_tmf_object_threshold* (minimum number of targets for an activity required to enable the TMF-objects-cache, keeping the target label and the target object ID). If activated, all the targets in the environment are stored in cache, as a result of `get_all` invocation. This does not reflect updates of endpoint data (for example, if an endpoint is deleted). The cache is refreshed after `global_cache_refresh_timeout` (default is 120 minutes).

Note: Cache usage improves the submission phase, but it consumes memory. We recommend that you use the cache when submissions to a large number of endpoints occur or the endpoints might be in separate TMRs, so remote lookup might delay the whole process. Use the target resolution at the activity level only if needed. The targets are validated at each activity execution by the Planner Monitor Executer process. The target resolution at plan submission is resolved only once at the submission of the plan.

Tivoli Configuration Manager V4.2.1 Fix Pack FP01 introduced a new parameter, `retrieve_gateways_info =yes (no)`, in `apm.ini`. At submission time, the retrieve of gateway information (`get_endpoints_by_label`) can be skipped. This might improve your submission plan processing time when your plan consists of thousands of endpoint targets, but you will not have the information about the gateways available (`wmonpln -wi plan_id -a activity -g`).

8.4.2 Reporting performance tuning: Skipping notification manager

As we have seen, the notification manager is single thread and processes the reports from the queue on the Tivoli server `$BINDIR/./swdis/work/messages`.

We strongly recommend that you use the command (not Activity Planner) `wswdcfg -s notify_ext_directly{*}=a` to enable it.

Notification manager might be very slow to release a conditioned activity, and the Activity Planner Monitor handler process might be overloaded if many reports are received. This parameter enables you to skip notification manager at the report phase and skip the state validation for the next submission on the server (if you specify `a`). It can be set on the plan base (in the example, the parameters apply to all plans). Therefore, the next distribution for the conditioned target starts as soon as the conditioning one completes. You will notice that the status for a specific endpoint and package might not be in the table `SD_INST`. However, Software Distribution will check the status of the package in the local endpoint catalog for the next conditioned activity to proceed.

8.4.3 Plan performance tuning: Number of activities

Tivoli Configuration Manager V4.2.1 Fix Pack 01 introduced a redesign of the Activity Planner handler report queue. The requests are grouped and processed by activity. The order of processing depends on the arrival time of the first request of each subqueue. Now, `repqueue.dat` is 248 bytes long if no activity is queued. Therefore, access to RDBMS is less frequent, but involves more rows. If the `APMHandler` is overloaded, `repqueue.dat` can grow to MBs, and the processing of the file is faster.

Tips: We recommend that you follow these guidelines for your plan build and submission:

- ▶ Plan your distributions to have the least amount of fragmentation as possible, because the fragmentation grows with the number of conditioned activities. Keep the number of conditioned activities under control if the conditions are specified by target (successful target, ST, or complete target, CT).
- ▶ Delegate the complexity of the plan at the software package level (whenever possible).
- ▶ Keep the number of the targets under control (especially for conditioned plans). We recommend that you maintain an average of 5000 endpoints x 5 activities (CT or ST).
- ▶ When you submit a plan with conditioned activities (ST or CT), only the first activity generates a distribution to all the targets; the other ones use the multistart mechanism.

8.4.4 Activity Planner Monitor: Executer threads

With the introduction of Tivoli Configuration Manager V4.2.1, the executer process is now multithreaded. The executer thread is responsible to start the software distribution and task. The `executer_max_threads` parameter defines the maximum number of threads that have to be used to execute external activities. The thread closes after `executer_max_idle_time`. Possible values are 0 (pool disabled), -1 (not limited), and any finite number. We do not recommend that you extend the value over 5.

8.4.5 Activity Planner Editor: Plan layout and large number of activities

When a plan has a large number of activities, activities might not fit in your Activity Planner Editor panel. In such a case, you might need to redesign your plan to be moved onto different X and Y coordinates.

Tip: We provide a script, `coord.sh`, that will redraw the plan based on the X and Y coordinates of the plan. The new coordinates are calculated and updated in the table. We provide the script in “`coord.sh`” on page 379.

8.4.6 Deleting plans: Cancel pending

You might need to delete a plan that was already submitted. For example, a plan might remain in Cancel Pending status for a long time. In that case, we provide a script, `del_plan.sh`, that requires a file containing the list of plans to be removed generated by `wlstpln -l` command. We provide this script in “`del_plan.sh`” on page 381.

8.5 Change Manager

Be aware that the Change Manager status must be enabled (see the output for the `wswdmgr -s` command), because the differencing mechanism is computed based on the `SD_INST` table contents.

8.5.1 Assign priority: Order of activities using Successful Target

Tivoli Configuration Manager V4.2.1 FP01 and later enables you to assign priorities in your reference model. By default, the generated activities are not conditioned. “Assign a Priority” enables you to assign an order of installation that will generate Successful Target, ST, condition. Configuration elements with priority 0 are excluded from conditioning mechanism. You can specify different conditioning branches by using different values for the column “Priority Set.”

Important: Note that these priorities are not related to the MDist 2 priority related to the high, medium, and low sessions. Take care in assigning a priority to an Inventory scan element in the middle of the chain, because conditioning by all is generated (instead of by target).

8.5.2 Adding states to the desired state

The `stable.xml` file is located under the `$BINDIR/TME/CCM/GUI` directory. It is the configuration file that contains the details about the transition from the current to desired state. It consists of a collection of `<table_entry>` items. Each item contains the operation and the options for the resulting activity plan. The items do not cover all possible combinations: If you need additional combinations, modify the file (see the example for transition IC-BC to IC---). If the current state of your target is not specified in the `stable.xml` file, or the transition from current to desired state is not specified, you get an error when running Preview or Synchronization.

8.6 Inventory tuning

There are three critical factors when dealing with an Inventory infrastructure deployed using IBM Tivoli Configuration Manager:

- ▶ Number of interobject message (IOM) sessions

The CTOC processing uses pthreads which are configured by `max_input_thread` and `max_output_threads` (that is, `wcollect -t 10 @InvDataHandler:inv_data_handler, wcollect -o 10 @ManagedNode:mn`).

Be aware that each pthread opens an IOM channel to pick up data pack, and the same IOM channel is used if the data pack source is the same as previous connection. A new IOM channel is opened if the data pack source is different from previous connection.

Tip: RPC threads are used to maintain IOM channels for upcall (CTOC) and downcall (data packs). Therefore, the collector inventory activity might also influence the resource exhaustion of your gateway. Be careful with the number of MDist 2 sessions in use for Software Distribution and Inventory so that you do not consume all your sessions for one unique purpose.

- ▶ Gateway collectors output threads/data handler input: Threads balance

It is required that the sum of all your downstream collector output threads not to be greater than your data handler input threads.

- ▶ Database deadlock/performance

RIM connections might originate RDBMS deadlocks caused by too many connections. We recommend that you use a single RIM object and five connections as a start (no multiple RIM objects). Be aware there are not magic numbers. You need to tune each installation depending on the customer operations. The following patches introduced a solution to avoid database deadlocks:

- 4.0-INV-FP06, 4.2-INV-FP01, 4.2.1 GA

The code was modified so that commits are performed each 100 rows on all Inventory tables.

- 4.0-INV-FP07, 4.2-INV-FP02, 4.2.1 GA

Introduced the use of temporary tables to avoid unique constraint violation when inserting `FILE_PATH` and `FILE_DESC` tables in the Inventory schema.

Tip: Since 3.7.1-CLL-FP01 4.1-CLL-FP01, 4.2.1-INV-FP01, 4.2-INV-0021, and 4.0-INV-0052, new tuning parameters for the Inventory data handler (**wcollect -n**) were introduced to avoid performance problems due to large checkpoint files. We recommend that you balance the data handler input threads and gateway collector output threads so that the data handler input queue does not grow too large.

Isolated versus standard Inventory scans

Note that when you perform an isolated scan (using the **wepscan -i** option), instead of a standard Inventory scan, the field **TME_OBJECT_ID** in the table **COMPUTER** is filled in with the tag **#ISOLATED**. Therefore, you will not be able to perform a software distribution to an isolated machine later if that field is not set to the real endpoint object ID.

Tip: We recommend that you balance the number of scans to be performed among your Tivoli gateways through the **wepscan** commands so that you do not overload a specific gateway. When using isolated scans, you can schedule the inventory upload by scheduling the **wloadiso** command that uploads the inventory information gathered previously through an isolated scan.

8.7 Resource Manager

In this section, we discuss best practices when using the Resource Manager.

8.7.1 Tivoli Web Gateway: Device control

In order to properly managed the enrollment process of your PDA, we recommend that you set the **AutoEnrollment** property to **FALSE**. This way, you will control what PDAs are enrolled through the **DMSMsgx.log (x=1,2,3)**. You need to define the device by running the **wresource add** command.

For example, for a WinCe device named **nachopda**, use the command syntax:

```
wresource add -i -u Pervasive_Device nachopda web_gw Wince:nachopda:nachopda 2
```

If the device was not defined, an entry would not be inserted in the **DEVICE** table of the Device Management Server (DMS) RDBMS. In addition, we can check for an error in **DMSMsgx.log (x=1,2,3)**, as shown in Example 8-3 on page 343.

Example 8-3 DMSMsgx.log

```
07/07/2005 7:38_PM com.tivoli.dms.enrollserver.DeviceEnrollmentServlet service
Tivoli Tivoli Web Gateway Device_Manager_Server MX0000001660031
DYM3049E An error occurred enrolling device MX0090001Z50034:Wince. Error =
DYM2043E: A device entry was not inserted into the database because the server
setting indicates AUTO_ENROLL is set to false.
```

Then, you can run the **wresource add** command, and the device will enroll properly.

8.7.2 From Tivoli Web Gateway to Tivoli Configuration Manager: Results process

The Tivoli Web Gateway database stores all jobs for all devices and the results of all the jobs. Therefore, after a job completes or fails, the job completion status gets updated in the DMS database. To integrate the Web Gateway with Configuration Manager and get results back to the Configuration Manager Tivoli server, there is a results collector that runs every 5 seconds and checks the Web Gateway directories and the DMS database for any new results. If there is any, it sends them to mcollect, running on the endpoint. The endpoint then forwards the results up to the Tivoli server, through the Tivoli gateway. After the operation succeeds, the data is deleted from the Web Gateway directories and the DMS database.

We summarize the results process here. There are several tables in the DMS RDBMS that are involved in the results collection process. You can perform the following queries listed to search the results that are still pending to be processed:

- ▶ PACKAGE
select package_id from where package_id=directory_id
- ▶ PACKAGE_DIST_ASSC
mdist2_id=select dist_id from package_dist_assc where package_id=directory_id
- ▶ JOB_DIST_ASSC
jobmdist2_id=select job_id from job_dist_assc where dist_id=mdist2_id
- ▶ ACTIVE_JOB_HISTORY
dev_id=select device_id from active_job_history where job_id=jobmdist2_id
- ▶ DEVICE
select device_name from device where device_id=dev_id

► **JOB_RESULT**

```
select result_file_name,result_state from job_result where result_state='I'
```

Result process steps

You will find that every time a new operation (software distribution or inventory scan) is performed on the device or PDA, a directory is created under the `.../twg/device` directory on your Web Gateway machine. In addition, check that a message is inserted in the `$WASHOME/logs/DMS_Jobsx.log` ($x=1,2,3$), as shown in Example 8-4.

Example 8-4 *DMS_Jobsx.log*

```
07/04/2005 3:47_PM Job Executed Tivoli Tivoli Web Gateway NachoWebGW1
  Target Device = Wince:nachopda JobID = 50704134113999979 Job type =
INVENTORY Job priority = 100 Job interval = 0
Job interval unit = null Job Parms =
{DMS__INV_PROFILE_PARM=http://tivoliipda.itso.ibm.com:80/twg/device/507041341137
77859/config.dmp,DMS__INV_SCAN_TYPE_PARM=Hardware,Software,Configuration,DMS__I
NV_SCAN_TABLES_PARM=BATTERY,COMPUTER,DB_INFO,DEV_CARD,DEV_INFO,FILE_DESC,FILE_P
ATH,INST_DB_INFO,INST_DEV_CARD,INST_NATIV_SWARE,NATIV_SWARE,UNMATCHED_FILES,WIN
CE_CFG} Device Parms =
{}
```

The following steps summarize the result process:

1. A distribution to a resource group (can vary from one to many devices) is performed to the Web Gateway endpoint where the DMS application is installed. An MDist 2 transaction ID is assigned.
2. A directory in `.../twg/device/...` is automatically created.
3. An entry in the `PACKAGE` table is recorded in `PACKAGE_ID` pointing at the directory `.../twg/device/...`
4. An entry in the `PACKAGE_DIST_ASSC` table is recorded in the fields `PACKAGE_ID` and the MDist 2 transaction ID in the field `DIST_ID`.
5. An entry is recorded in `JOB_DIST_ASSC` with the job ID and MDist 2 transaction ID in the `JOB_ID` and `DIST_ID` fields.
6. An entry is recorded in `ACTIVE_JOB_HISTORY` with the job ID and device ID under the `JOB_ID` and `DEVICE_ID` fields.
7. An entry is recorded in `JOB_RESULT` when the device synchronizes itself. The `RESULT_STATE` for each entry is:
 - “I” means that the results are waiting to be sent to Tivoli Configuration Manager by the results collector.

- “P” means that the results converter reads the table JOB_RESULT, converts the results (see RESULT_FILE_NAME in JOB_RESULT), and generates them into the file specified in RESULT_FILE_NAME.
 - The mcollect method is called, passing the result file names. The Tivoli Configuration Manager collector sends the information up to Tivoli Configuration Manager Inventory database. The files in the results directory (TivTwg/results) are renamed to have a .don extension. The .don extension indicates that the results were correctly sent.
8. Cleanup is performed by a different thread that reads the results directory searching for the .don files. Any .don file that matches the RESULT_FILE_NAME in JOB_RESULT will delete that record from the JOB_RESULT table. In addition, the file .don will be deleted from the TivTwg/results.

DMS database and device management

You will find that every time a new operation (software distribution or inventory scan) is performed on the device or PDA, a directory is created under the .../twg/device directory on your Web Gateway machine. In addition, check that a message is inserted in the \$WASHOME/logs/DMS_Jobsx.log (x=1,2,3) as shown in Example 8-5.

Example 8-5 DMS_Jobsx.log

```
07/04/2005 3:47_PM Job Executed Tivoli Tivoli Web Gateway NachoWebGW1
  Target Device = Wince:nachopda JobID = 50704134113999979 Job type =
INVENTORY Job priority = 100 Job interval = 0
Job interval unit = null Job Parms =
{DMS_INV_PROFILE_PARM=http://tivolipda.itso.ibm.com:80/twg/device/507041341137
77859/config.dmp,DMS_INV_SCAN_TYPE_PARM=Hardware,Software,Configuration,DMS_I
NV_SCAN_TABLES_PARM=BATTERY,COMPUTER,DB_INFO,DEV_CARD,DEV_INFO,FILE_DESC,FILE_P
ATH,INST_DB_INFO,INST_DEV_CARD,INST_NATIV_SWARE,NATIV_SWARE,UNMATCHED_FILES,WIN
CE_CFG} Device Parms =
{}
```

Note that the directory ID 50704134113777859 is inserted in the PACKAGE_ID field in the PACKAGE table in DMS RDBMS.

After the inventory has been uploaded, a message will be inserted in the file DMS_Jobx.log as shown in Example 8-6.

Example 8-6 DMS_Jobx.log

```
07/04/2005 3:47_PM Job Completed Tivoli Tivoli Web Gateway NachoWebGW1
  Target Device = Wince:nachopda JobID = 50704134113999979 Job type =
INVENTORY Status = OK
```

Tip: To work with resources after you enroll or create them, you must make them part of a resource group. Consider creating a static resource group (one for project/customer, for example) instead of a dynamic group. We recommend that you maintain small or even one-to-one profile managers and PDA resource group subscribers in order to closely control the management of inventory and software distributions. Be aware that a directory will be created under the `../twg/device` in your Web Gateway machine and it will not be removed until all the PDAs that were distributed a profile effectively synchronize. Therefore, you might end up with many directories and entries in your database unless you closely control the PDA operations.

8.7.3 Jobs in “I” status remain in JOB_RESULT

The results collector looks at the database table `JOB_RESULT` to find new results. When results are put into this table, their `RESULT_STATE` is set to “I”. This means that they are waiting to be sent to Tivoli Configuration Manager. If you have tracing turned on for the results collector component in the Tivoli Web Gateway, you will see messages in the `DMS_stdout.log` file each time the results collector looks for new results. For example, Example 8-7 shows a couple of the messages you might encounter.

Example 8-7 DMS_stdout.log

```
: [11/6/03 8:02:39:164 EST] 250da4 SystemOut U [THREAD Thread-11]
com.tivoli.dms.resultscollector.HarvesterTask.getReadyResults(HarvesterTask.java:321)
```

There are no unprocessed results in the database.

This means that the results collector looked in the `JOB_RESULT` table for rows that had a `RESULT_STATE` of I and found none. If it did find at least one (in the following case, two were found), you would see the message shown in Example 8-8.

Example 8-8 DMS_Jobsx.log

```
[11/6/03 8:03:29:214 EST] 250da4 SystemOut U [THREAD Thread-11]
com.tivoli.dms.resultscollector.HarvesterTask.getReadyResults(HarvesterTask.java:306)
```

There are two results ready to be processed.

In this case, you need to wait for the results converter to convert the results under `/TivTwg/results`.

8.7.4 Jobs in “P” status remain in JOB_RESULT

After the results collector has found results, the next step is to put them into a format that the different applications (Inventory, Software Distribution, Resource Manager) can understand. Therefore, Web Gateway runs a results converter to do this, where the code to convert the results is provided by each of the applications. For example, in Tivoli Configuration Manager V4.2.1, the Inventory converter produces a .dat file, while the Software Distribution and Resource Manager converters produce .xml files. After the results have been converted, the files are put in the results directory under the root Web Gateway directory (for example, /opt/TivTgw/results). In addition the JOB_RESULT table in the Web Gateway database gets updated with the file name (RESULT_FILE_NAME) and the RESULT_STATE gets changed to P. Example 8-9 shows a couple example trace messages you would see from the Software Distribution results converter.

Example 8-9 Trace messages

```
[11/6/03 8:03:29:329 EST] 250da4 SystemOut U [THREAD Thread-11]
com.tivoli.dms.resultscollector.SoftwareDistributionResultsConverter.buildResul
ts(SoftwareDistributionResultsConverter.java:95)
  Entry - distID = 1749776576.5
[11/6/03 8:03:29:851 EST] 250da4 SystemOut U [THREAD Thread-11]
com.tivoli.dms.resultscollector.SoftwareDistributionResultsConverter.convertRes
ults(SoftwareDistributionResultsConverter.java:224)
  Entry - outFile =
/opt/TivTgw/results/1749776576#SoftwareDistribution/f8b123e83b
```

The final step is to send the results to mcollect. Web Gateway calls an mcollect method with the names of the results files. Mcollect then sends those files to mcollect on the Tivoli server and to the corresponding applications (through the Tivoli endpoint and gateway). Web Gateway waits for this mcollect method to return successfully before deleting those results from the JOB_RESULT table. When mcollect finishes, it renames the results files (in the TivTgw/results directory) to have a .don extension. This lets Web Gateway know that the results have been successfully sent.

Example 8-10 shows the trace message you will see when Tivoli Web Gateway calls mcollect to send the results.

Example 8-10 Trace message

```
[11/6/03 8:03:29:867 EST] 250da4 SystemOut U [THREAD Thread-11]
com.tivoli.dms.resultscollector.HarvesterTask.callMcollect(HarvesterTask.java:2
49)
Send '/opt/TivTgw/results/1749776576#SoftwareDistribution/f8b123e83b' to
MCOLLECT - begin
```

Tip: When the /TivTwg/results directory has no files with the .don extension, the records in “P” state will not be deleted. Check whether the Configuration Manager Inventory database for the selected device already has the device information. If so, you can safely then manually remove the entry in the “P” state from the JOB_RESULT table.

8.7.5 Remaining directories in /twg/device/: No jobs in JOB_RESULT

Web Gateway has a separate thread that runs to perform cleanup. This happens when the results are deleted after we know the mcollect method has successfully processed them. When it looks for results to delete, it looks in the results directory (TivTwg/results) for results that have the .don extension. It uses this file name to search the RESULT_FILE_NAME in the JOB_RESULT table to find the corresponding result, deletes this row from the table, and then deletes the results file.

Example 8-11 shows the trace messages you will see when results and files are deleted.

Example 8-11 Trace messages

```
[11/6/03 8:03:42:704 EST] 48c02f SystemOut U [THREAD Thread-12]
com.tivoli.dms.resultscollector.CleanupTask.cleanupResults(CleanupTask.java:130
)
    Deleting 2 result(s) from the database
[11/6/03 8:03:42:724 EST] 48c02f SystemOut U [THREAD Thread-12]
com.tivoli.dms.resultscollector.CleanupTask.cleanupFiles(CleanupTask.java:91)
    Deleting 2 file(s) from the machine
```

Tip: There might be scenarios where the directory under ../twg/device.. remains without being deleted. In this case, it would seem as though an operation (inventory or software distribution) has been performed, but some PDAs have not synchronized for a long time. Check the Configuration Manager Inventory database for the PDAs that are not synchronizing. Ensure the end user turns on the PDA or PDA is properly configured. You may remove the PACKAGE_ID (it is the directory name under ../twg/device/...) from the PACKAGE table and associated entries in PACKAGE_DIST_ASSC, JOB_DIST_ASSC and ACTIVE_JOB_HISTORY should you need to perform the operation again (inventory or software distribution).

8.7.6 Web Gateway logging settings

You might need to redirect your log files to different file systems. The `DMS_Stdout.log` and `DMS_stderr.log` files can be redirected using the WebSphere advanced administrative console.

The `DMSmsgx.log` and `DMSJobx.log` files can be redirected by editing the `commonConfig.properties` file and setting the variable to a different path than the default one in `WAS_LOGS_DIR=/usr/WebSphere40/AppServer/logs`.

Important: Tivoli Configuration Manager V4.2.1 Fix Pack FP03 provides a new Pocket PC device agent feature. With this feature, you can easily reinstall the IBM agent application after a hard reset or when the battery dies, without having to install the IBM agent application from scratch. To ensure a persistent one-click installation of the device agent after a hard reset or when the battery dies, enable this feature. This was also solved in Tivoli Configuration Manager V4.2.3.

8.8 Pristine Manager

Pristine Manager leverages the Automated Deployment Services and Remote Installation Service from Microsoft.

Automated Deployment Services is a new solution delivered with Windows Server 2003, Enterprise Edition, and Windows Server 2003, Datacenter Edition, and is designed for automated, high-speed server deployment.

Remote Installation Service was first delivered with Windows 2000 and has been enhanced in Windows Server 2003 to enable fully automated deployments. Remote Installation Service now supports deployments to servers as well as to desktops.

important: Of the two solutions, only Remote Installation Service supports deployment of desktops, that is, computers targeted to run a Windows client operating system, such as Windows XP. Automated Deployment Services is designed and optimized for deployment of servers, that is, Automated Deployment Services is targeted to run a Windows server operating system, such as Windows Server 2003.

Remote Installation Service deploys:

- ▶ Windows XP: All editions except the Home Edition
- ▶ Windows Server 2003: All 32-bit versions

- ▶ Windows Server 2003: 64-bit version (scripted setup only)
- ▶ Windows 2000: All editions (Professional and Server)

Automated Deployment Services deploys:

- ▶ Windows Server 2003: All 32-bit versions
- ▶ Windows 2000: All Server editions (does not deploy Windows 2000 Professional)

Tip: Remote Installation Service requires several Windows operating system components to identify the computers in your network: DHCP server, DNS, and Active Directory. Remote Installation Service also relies in a unique identifier that each manufacturer assigns to its equipment or universal or global unique identifier (UUID or GUID). Each machine is presented to their Remote Installation Service machine using the UUID or GUID.

Notice that Tivoli Configuration Manager also provides a Pristine Tool that supports native pristine installation using a boot floppy disk to load the operating system and applications. For more information, review *IBM Tivoli Configuration Manager: User's Guide for Software Distribution, Version 4.2.2, SC23-4711*.

8.9 Tivoli Configuration Manager V4.2.3

Microsoft announced the availability of Windows Software Update Services as part of their new software patching strategy to simplify and expand coverage across Microsoft products. Tivoli Configuration Manager V4.2.3 and Tivoli Provisioning Manager 3.1 both leverage Software Update Services (SUS) and Microsoft Baseline Security Analyzer 1.2.1 as part of their automated patching capabilities. We urgently recommend that you download both Software Update Services SP1 and Microsoft Baseline Security Analyzer 1.2.1 to integrate them with Tivoli Configuration Manager V4.2.3.

Microsoft has confirmed their support of existing Software Update Services implementations through the end of 2005. After that time, Microsoft expects all customers to use Windows Software Update Services and Microsoft Baseline Security Analyzer 2.0 for software updates.

In addition, Tivoli will be updating IBM Tivoli Configuration Manager V4.2.3 and IBM Tivoli Provisioning Manager V3.1 to include support for the new Microsoft patching process.



Microsoft security checklist

In this appendix, we provide the security checklist for Microsoft Windows environments. Microsoft has a differing security model than most UNIX-style computers and it is our intent here to answer a lot of the questions regarding security and authentication in the Microsoft Windows environment.

\$root_user

The \$root_user account is a privileged account. This privileged account will allow privileged methods to run as administrator. This applies only to Microsoft Windows 2000, Windows Server 2003, and Windows XP. Microsoft Windows NT is covered, but is no longer supported by Microsoft (December 2004) and Tivoli.

If the administrator account has been renamed, the use of a single \$root_user account assignment will not work. The \$root_user account is resolved to a single account Tivoli management region (TMR) wide and cannot be changed based on a domain name. This keyword, BuiltinNTAdministrator, resolves to the SID500 account on the local machine.

Root access

Root access is required for the following functions:

- ▶ Installation of files to optional locations: Only the root account can override permission setting.
- ▶ Inventory of software can only be accessible by a customer user or root.
- ▶ Processing optional files that cannot be access by just any group or user.

widmap

The **widmap** command is used to display or change what account is being used for root access.

To display what is being used:

```
widmap list_entries root_user
```

To change what is being used:

```
widmap add_entry root_user w32-ix86 <value>
```

For example:

```
widmap add_entry root_user w32-ix86 BuiltinNTAdministrator
```

Access rights

The \$root_user account needs to be assigned to the Tivoli_Admin_privileges group. The \$root_user is automatically assigned this access right during the installation of the product and must remain for the Tivoli software to operate successfully. This is especially true for endpoints. Prior to Version 16 of the TivoliAP.dll, user access rights provided by a domain local group were not in the

token. To work around this problem, add the rights to the user specifically or upgrade the TivoliAP.dll to Version 16 or later and reboot the machine as required after the upgrade.

Table A-1 summarizes access rights for the \$root_user.

Table A-1 Access rights for the \$root_user

Security policy	Name	Description
Act as part of the OS	SeTcbPrivilege	User can act as part of the operating system.
Replace a process-level token	SeAssignPrimaryTokenPrivilege	
Windows 2003: Adjust memory quotas for a process Windows 2000: Increase quotas	SeIncreaseQuotaPrivilege	User can modify a process access token.

The following sections describe these rights in more detail.

Act as part of the operating system

This user right allows a process to impersonate any user without authentication. The process can, therefore, gain access to the same local resources as that user.

Replace a process-level token

This security setting will determine which user accounts can call the CreateProcessAsUser() application programming interface (API) so that one service can start another. The default is Network Service, Local System.

Note: Abbreviation of application program interface, a set of routines, protocols, and tools for building software applications: A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together. Most operating environments, such as Microsoft Windows, provide an API so that programmers can write applications consistent with the operating environment. Although APIs are designed for programmers, they are ultimately good for users because they guarantee that all programs using a common API will have similar interfaces. This makes it easier for users to learn new programs.

Adjust memory quotas or increase quotas

Depending on which version of Microsoft Windows you are using, the setting name has changed from Windows 2000 to Windows 2003. Windows 2000 uses the increase quotas and the Windows 2003 uses the adjust memory quotas.

This privilege will determine what account can change the maximum memory that can be consumed by a process. This user right is defined in the default domain controller group policy object (GPO) and in the local security policy of workstations and servers.

tmersvd

This account is unprivileged. The installation of the Tivoli product does not add this account to any groups. It does not need to be added to a specific group for the endpoint to operate successfully, but there are rights for the account that are required for the endpoint to work properly.

Table A-2 summarizes operating system rights for the tmersvd user.

Table A-2 Operating system rights for tmersvd

Security policy	Name	Description
Windows 2000: Logon locally Windows 2003: Allow logon locally	SeInteractivLongonRight	Account has the ability to log on to the computer.
Bypass traverse checking	SeChangeNotifyPrivilege	Account has the ability to pass through a directory without the read right.

When read and execute rights are assigned, list folder contents and read privileges are also added, as shown in Table A-3.

Table A-3 Directory access rights

Directory	User rights
%systemroot%\system32	Read and execute
%systemroot%\tmp	Read and execute

The rights do not need to be given specifically, but tmersvd cannot be denied access to the previously mentioned directories. If the environment has hardened security, the rights must be given specifically.

Windows registry

Auxiliary processes will need to be run as an unprivileged process and are required to run as a specific or different user. See Table A-4 and Table A-5.

Table A-4 Registry rights

Key	User rights
HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001	Read
HKEY_LOCAL_MACHINE_SYSTEM/CurrentControlSet	Read

Table A-5 More granular registry rights

Key	User rights
HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001/Control	Read
HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001/Services	Read
HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Control	Read

Principals of the user model

Unrestricted access is required to manage the system. Auxiliary processes, which do not require special access, should be run with minimal authority. This will reduce the security exposures to the systems. Auxiliary processes might need to run as a particular application, as a particular task, or as a particular user.

TivoliAP.dll

The Windows environment does not have a user impersonation function. Therefore, the Tivoli authentication policy is installed. This TivoliAP.dll is loaded at startup and registered with the Microsoft Windows local security authority (LSA). The programs that use this TivoliAP.dll are `oserv` and `lcmd`, which call the `LsaLogonUser()` application program interface (API), along with a user or security identifier (SID) and a TivoliAP-specific key. A requisite token is generated and returned to the caller. The caller calls the `CreateProcessAsUser()` API passing the token and authentication of the Tivoli remote access account user. The credentials of the Tivoli remote access account user authentication are included in the token that is being passed.

To view the current active TivoliAP.dll, issue the following command:

- ▶ For endpoints: `wlcfatp`
- ▶ For managed nodes/gateways: `wsettap`

The output will look similar to that shown in Example A-1.

Example: A-1 Output of the wsettap command

```
C:\>wsettap
1
19
1
Wed Nov 19 20:05:41 2003
(null)\(null)
Primary Domain Controller
```

The second number is the version of the TivoliAP.dll that is active. The time stamp is the build date of the file. The last field shown is the domain and account for the Tivoli remote access account. The `wsettap` command will display an additional value that indicates whether requests will go just to the primary domain controller (PDC) or the domain controller (DC). If access is denied, it is an indication that the user issuing the command does not have the “act as part of the operating system” user right.

Tivoli remote access account

The account is a prespecified user and password that will be used to access remote resources, such as remote drives. The token generated by the TivoliAP.dll will not access remote resources without it. If the Tivoli remote access account is not valid, the TivoliAP.dll token generation will fail.

Active Directory and TivoliAP.dll

Domain account authentication requires the TivoliAP.dll to query a domain controller. Before Microsoft Windows 2000 and Active Directory native mode, such queries were allowed. Special Active Directory configuration steps must be made so that only processes that are authenticated with a domain account can query information on a domain account. To get around these restrictions, take one of the following actions:

- ▶ Add the following privileges to the “Tivoli_Admin_Privileges” group: SeTcbPrivilege (act as part of the operating system), SeAssignPrimaryTokenPrivilege (replace a process level token), and SeIncreaseQuotaPrivilege (adjust memory quotas for a process). This change must be done on each server.
- ▶ Add everyone to the pre-Windows 2000 compatible access group.
- ▶ Use the `dsacls` tool to allow anonymous queries against specific users.

Tivoli roles authority structure

The following list summarizes the Tivoli roles:

- ▶ User: View Tivoli data
- ▶ Admin: Manage resources
- ▶ Senior: Control Tivoli resource layout
- ▶ Super: Change all the rules

Invoking a method

The Tivoli server checks authorization when it invokes a method on an object that resides within a security group in the following manner.

If the methods' access control list (ACL) contains a role, the object call is authorized regardless of the caller. Otherwise, the oserv will obtain the roles associated with the principal. The oserv will acquire the security groups list associated to the target object. The the oserv will check to see whether the caller has roles in any security group to which the object belongs. If there are no roles distinguished, the object call will fail.

To display the ACL, use:

```
objcall $TMR.1.23 om_get_acl check_db
```

See Example A-2.

Example: A-2 To display the ACL

```
# objcall $TMR.1.23 om_get_acl check_db
super
senior
admin
```

To display the capabilities, use:

```
objcall $TMR.1.179 get_capabilities
```

See Example A-3 on page 358.

Example: A-3 To display the capabilities

```
# objcall $TMR.1.179 get_capabilities
global
super
senior
admin
user
install_client
install_product
policy set_role_role
security_group_any_admin
user1254000003.1.179
rconnect user
```



Scripts referenced in the book

This appendix provides source codes for the scripts described in this book. These scripts are given on *as-is* basis. You can also download these scripts from the ITSO FTP site. For download instructions, refer to Appendix C, “Additional material” on page 387.

assess.pl

This tool evaluates and assess Framework-based software and Tivoli region environments. The scripts are non-intrusive (read only). The output is presented in HTML format for easy viewing in HTML format.

Example: B-1 assess.pl

```
#####  
# perl script to perform assessment of customer Tivoli environment.  Designed  
# to use ONLY perl 4 and Tivoli CLI commands for portability.  If use of  
# shell commands is deemed utterly essential, restrict to those in Tivoli  
# "bash" or test for interp type and use interp-specific command.  
#####
```

Contact your local Tivoli support to obtain the scripts included with the assessment tool launching the assess.pl script.

gw_tuning.pl

Example B-2 shows the source code for the gw_tuning.pl script.

Example: B-2 gw_tuning.pl

```
#!/etc/Tivoli/bin/perl  
#  
#  
#  
# GOAL: Implement script to support the Gateway Tuning  
#       Field Guide and Presentation.  
#  
#  
#  
# OVERVIEW: There are 4 parameters that can be tuned  
#           for the TMR/Gateway/Epmgr regarding 'threads':  
#  
#           (oserv) rpc_max_threads --> odamdin get_rpc_max_threads  
#           (epmgr) max_epmgr_rpc_threads --> wepmgr get  
#           (gw)   rpc_threads      --> wgateway gw_label  
#           (gw)   max_concurrent_jobs --> wgateway gw_label  
#  
#           The field guide makes the following recommendations:  
#  
#           max_concurrent_jobs == number of eps to the nearest 100,  
#                               but between 200 <--> 1000.  
#           gw_rpc_threads      == max_concurrent_jobs + 50  
#           oserv rpc_max_threads == gw_rpc_threads/(0.60)
```



```

# max_epmgr_rpc_threads == (0.60)*(oserv_rpc_max_threads)
#
# These recommendations provide for the following strategy to
# tune a TMR.
#
# 1. Determine how many endpoints per gateway?
# TMR --> figure out what the max_concurrent_jobs should be based on
# of endpoints (minimum is 200) figure out the max_rpc_threads...
# max_concurrent_jobs + 50
#
# MINIMUM number of oserv threads is(with default settings...):
#
# TMR/epmgr --> 420 (250(max_epmgr_rpc_threads)/.6)
# TMR/epmgr/gw --> 840 (250(max_epmgr_rpc_threads + gw rpc threads(250))/.6)
#
# So, unless the TMR Gateway has more than 200 endpoints, one of the above is the
# MINIMUM setting.
#
# Gateway --> Find the MAXIMUM number of endpoints per gateway (EPMAX)
# MAX_JOBS == get max_concurrent_jobs from EPMAX.
# MAX_OSERV_RPC = (MAX_JOBS+50)/.6
#
# OSERV_THREADS == MAX_OSERV_RPC or what was calculated for TMR, whichever is greater.
#
#
# Based on the whitepaper, the maintuning factor is endpoints per gateway.
# But, actual value of OSERV_THREADS is limited by the file descriptors.
# So, an NT max value is 2038 but a unix may need to be adjusted...
#
# so... action would be to figure out what we wanted to set them to...
#
# Then set the OSERV_THREADS Then 'check' each GATEWAYS 'actual'
# value of OSERV_THREADS and set everything else appropriately
#
# We're only going to work ONE TMR at a time... Get gateways in this TMR:
#=====
#-----
# Setup Requirements and paths to search for require files.
#-----

push(@INC,"/etc/Tivoli/lib/perl");
push(@INC,$BINDIR."/contrib/lib/perl");
push(@INC,$BINDIR."/tools/lib/perl");

require "getopts.pl";          # Library provided by Tivoli, should exist
                              # on any ManagedNode/TMR Server

```

```

#-----
# Check for Tivoli Environment being sourced...
#-----

$TMR = $ENV{TMR};
if ($TMR eq "") {
    print "Please source your Tivoli Environment...\n";
    exit 0;
}

*****
#* Confirm Proper options passed, generate USAGE
*****
$usage = 1 if ($#ARGV < 0);
&Getopts('hSR');

if ($usage || $opt_h) {
    print <<END_OF_USAGE;

Usage: $0 [-h | -R | -S ]

    Calculate the recommended values of RPC Threads for
    the oserv, epmgr and gateway.  Additionally, calculate
    the max_concurrent_jobs setting for the Gateways.

    (Read Script Header for methodology, or the IBM/Tivoli
    Gateway Tuning Field Guide)

    This script only works on the TMR it is being run on.
    It should be run on each TMR in an environment.

    No Options      This Message

    -R              Calculate and show the recommendations ONLY.

    -S              Calculate the recommendations AND actually
                    set them in the TMR.

    -h              This usage statement.

END_OF_USAGE
    exit;
}

```

```

#-----
# Define Variables
#-----
$|=1;          # flush stdout...

$PERCENT      = 0.60;  # This is the 'recommended' percentage of
                      # oserv rpc threads that should be used
                      # by other processes (epmgr and gateway)

#-----
# Define and load hashes
#-----

%Gateways     = ""      ; &get_gws;          # maps gw_label=>OID
%num_eps_gw   = ""      ; &get_eps_per_gw;  # maps gw_label=>num_eps

#-----
# Calculate rpc_threads/max_concurrent jobs for all gw's in TMR
#-----

%max_jobs     = "";     # maps gw_label => max_jobs
%max_gw_threads = "";   # maps gw_label => max_rpc_threads (gw)
&calc_gw_settings;     # loads previous 2 hashes

%curr_max_jobs = "";
%curr_max_gw_threads = "";
&get_curr_gw_settings; # get the old gateway settings, place
                      # in hash above.

#-----
# oserv threads are a TMR 'global' setting
#
# We must calculate the MAX value of this based on what the largest
# number of max_concurrent jobs should be...
# This may be affected by epmgr threads and if a GW on TMR.
#
#-----

($MAX_OSERV_THREADS,$EPMGR_THREADS) = &calc_tmrsrvr_threads;

#-----
# Which are the problem gateways now...
# Answer: Any gw whose max_rpc_threads/gw_oserv_threads < 0.60
#-----

%gw_oserv_threads = "";
&get_oserv_threads;

```

```

#-----
# print recommended New Settings
#-----
print <<END_OF_HEADER;
=====
                        RECOMMENDED SETTINGS
=====
END_OF_HEADER
printf "%8s %20s|%27s|%13s\n", "Possible", "", "Current          ", "Recommended      ";
printf "%8s %20s|%6s %6s %6s %6s|%6s %6s\n",
"Problem", "GW_LABEL", "oserv", "epcnt", "jobs", "gwrpc", "jobs", "gwrpc";
print "=====\\n";
foreach (sort keys %Gateways) {
    $problem = "";
    if ($gw_oserv_threads{$_} > 0 ) {
        $problem = "***" if (($max_gw_threads{$_}/$gw_oserv_threads{$_}) > 0.60);
    } else {
        $problem = "??";
    }
    printf "%8s %20s|%6d %6d %6d %6d|%6d %6d\\n",
$problem,$_,$gw_oserv_threads{$_},$num_eps_gw{$_},$curr_max_jobs{$_},$curr_max_gw_threads{$_},$
max_jobs{$_},$max_gw_threads{$_};
}
print "=====\\n";
printf "          oserv threads: %5d epmgr threads: %5d\\n", $MAX_OSERV_THREADS,
$EPMGR_THREADS;
print <<END_OF_FOOTER;
=====
NOTE:  ** in the PROBLEM column indicate rpc_threads on the oserv
       may be limited by file_descriptors or oserv needs to be
       restarted

       ?? in the PROBLEM column indicate Gateway appears to be down.
=====
END_OF_FOOTER

#-----
# The Recommended values have been figured out now, now we want to
# set them.
#
#-----

if ($opt_S) {

    print "Setting oserv threads: odadmin set_rpc_max_threads $MAX_OSERV_THREADS\\n";
    &set_oserv_threads($MAX_OSERV_THREADS);

    print "Setting Gateway Parameters\\n";

```

```

    &set_gw_parms;

    print "Setting EPMGR Threads\n";
    &set_epmgr_threads($EPMGR_THREADS);
} elsif ($opt_R) {
    print "Recommendations Only, use -S to actually apply these settings\n";
}

exit 0;

#-----
# END OF MAIN
#-----

#-----
# Subroutines
#-----

sub calc_tmrsrvr_threads {
#-----
# Calculate the recommended number of OSERV Threads
# and Endpoint Manager threads. Return these in an
# ARRAY
#
# Calculates from the max_gw_threads hash.
#-----

    # get the largest number of gw_threads...

    @list = sort { $b <=> $a } values %max_gw_threads;
    $biggest = $list[0];
    $max = (int(($biggest/$PERCENT)/10)*10);

    # If TMR is a Gateway, then we need to calculate the 'other' numbers...
    #     MINIMUM number of oserv threads is:
    #
    #     TMR/epmgr    --> 420 (250(max_epmgr_rpc_threads)/.6)
    #     TMR/epmgr/gw --> 840 (250(max_epmgr_rpc_threads + gw rpc threads(250))/.6)
    #
    # Now we have the 'max_oserv_threads' based on the gateway settings... but this
    # may need to change based on the TMR if its a gateway...
    #
    # However, best practice dictates the TMR SHOULD not have many endpoints on it...
    #
    # so, the max based on TMR is: (tmr_gw_max_rpc_threads + epmgr_threads)/(0.60)
    #
    # but we have NO idea what epmgr_threads should be, so we want to take make sure 60%

```

```

# of the $max we've calculate so far is

#-----
# See if TMR is a Gateway
#-----
$tmr_gw_label = &get_tmr_gateway;

if ($tmr_gw_label ne "" ) {
#
# if TMR is a gateway, then we calculate what the epmgr_threads should be.
#

$epmgr_threads = ($max*($PERCENT) - $max_gw_threads{$tmr_gw_label});

# but, if epmgr_threads < $max_gw_threads, we need to increase $max...
# and set epmgr to be the same as max_gw_threads...

if ($epmgr_threads < $max_gw_threads{$tmr_gw_label}) {
    $epmgr_threads = $max_gw_threads{$tmr_gw_label};
    $max = (int(($epmgr_threads + $max_gw_threads{$tmr_gw_label})/$PERCENT)/10)*10;
}
}
return ($max,$epmgr_threads);
}

sub calc_gw_settings {
#-----
# Calculate the max_concurrent_jobs and rpc_maxthreads
#
# Logic for max_concurrent_jobs is based on:
#   MINIMUM: 200
#   MAXIMUM: 1000
#
# Value of max_concurrent_jobs should be number of eps
# on a gateway rounded UP to the nearest 100, but staying
# between the limits noted above.
#
# Value of rpc_maxthreads should be max_concurrent_jobs+50.
#
# Calculates from the num_eps_gw hash.
#
# Creates 2 GLOBAL hashes: %max_jobs and %max_gw_threads.
#-----

foreach (keys %num_eps_gw) {
    if ($num_eps_gw{$_} <= 200 ) {
        # less than 200, set it to 200.
        $mj=200;
    }
}
}

```

```

    } elsif ($num_eps_gw{$_} >= 1000 ) {
        # more than 1000, set it to 1000.
        $mj=1000;
    } else {
        # calculate nearest 100.
        $mj = ((int($num_eps_gw{$_}/100)+1)*100);
    }
    $max_jobs{$_} = $mj;
    $max_gw_threads{$_} = $mj+50;
}
}
sub get_curr_gw_settings {
    print "Getting Current GW Settings\n";
    foreach (keys %Gateways) {
        &print_dot;
        $curr_max_jobs{$_} = `idlatrr -tvG $Gateways{$_} max_concurrent_jobs along 2>&1`;
        $curr_max_gw_threads{$_} = `idlatrr -tvG $Gateways{$_} rpc_maxthreads ushort 2>&1`;
    }
    print "\n";
}
}

sub get_gws {
#-----
# Build %Gateways Hash with list of Gateways in
# local TMR only.
#
# Local TMR is defined by environment variable $TMR.
#
#-----
    open(GWS, 'wlookup -ar Gateway|');
    local(@line)= "";
    while (<GWS>) {
        next unless /$TMR/;
        chop;
        /(\S+)\s+(\S+)/ && do {
            $Gateways{$1} = $2 } ;
    }
}

sub get_eps_per_gw {
#-----
# Build num_eps_gw Hash
#
# Maps number of endpoints to gateway.
#
#-----
    print "Getting the number of endpoints per Gateway\n";

```

```

    foreach (keys %Gateways) {
        &print_dot;
        $count = `wep ls -g $_ | wc -l`;
        chop($count);
        # $num_eps_gw{$_} = int(rand(1500)) + 1;
        $num_eps_gw{$_} = $count;
    }
    print "\n";
}

sub get_tmr_gateway {
#-----
# Determine label of gateway on TMR
#-----
    foreach (keys %Gateways) {

        return $_ if ($Gateways{$_} =~ /\d+\.1\.\d+/);
    }
    return "";
}

sub get_oserv_threads {
#-----
# For Each Gateway, get the ACTUAL value
# its oserv rpc_max_threads are set.
#
# This seems to be limited on WHEN the gateway was created and
# is DEFINITELY limited by the Gateway's OS File Descriptors
#
#
# Build the %gw_oserv_threads hash.
#
#-----
    print "Getting actual oserv rpc_threads per gateway...\n";
    foreach $gw_label (keys %Gateways) {
        &print_dot;
        $oid = $Gateways{$gw_label};
        $oid =~ s/(\^d+\.d+\.)*$/1/;
        $oid .= "2" ;
        $line_out = `objcall $oid query get_rpc_max_threads 2>&1`;
        # in format "Maximum number of RPC threads is 2048"
        if ($? == 0) { # SUCCESS
            ($threads) = $line_out =~ /Maximum number of RPC threads is (\d+)/;
        } else {
            $threads = 0;
        }
        $gw_oserv_threads{$gw_label} = $threads;
    }
}

```



```

    print "\n";
}

sub set_oserv_threads {
#-----
# Set oserv_threads...
#-----
    $oserv_threads = shift;
    if (system("odadmin set_rpc_max_threads $oserv_threads 2>&1")) {
        return TRUE;
    } else {
        return FALSE;
    }
}

sub set_gw_parms {
    foreach (keys %Gateways) {
        print "Setting parameters for GW: $_ ";
        $rc1 = system("wgateway $_ set_rpc_maxthreads $max_gw_threads{$_} 2>&1");
        $rc2 = system("wgateway $_ set_max_concurrent_jobs $max_jobs{$_} 2>&1");
        # Note: Gateway must be restarted...
        $rc3 = system("wgateway $_ restart 2>&1");
        if ($rc1 | $rc2 | $rc3) {
            print "FAILED\n";
        } else {
            print "SUCCESS\n";
        }
    }
}

sub set_epmgr_threads {
#-----
# Set epmgr_threads...
#-----
    $epmgr_threads = shift;
    if (system("wepmgr set_max_epmgr_rpc_threads $epmgr_threads 2>&1")) {
        return TRUE;
    } else {
        return FALSE;
    }
}

sub print_dot {
    print ".";
}

```

find_rogue

Example B-3 shows the source code for the find_rogue script.

Example: B-3 find_rogue

```
#!/etc/Tivoli/bin/perl
die "Tivoli environment not defined\n" if ! $ENV{DBDIR};
# Either use log file in first argument, or default is standard location

if($ARGV[0]){
    open(EPMGRLOG, "$ARGV[0]") || die "Cannot open file $ARGV[0]\n";
}else{
    open(EPMGRLOG,"$ENV{DBDIR}/epmgrlog") || die "Cannot open Endpoint Manager
log file\n";
}
while(<EPMGRLOG>){
    if(! $start_time){
        if(/(\d+)\.(\d+)\.(\d+)\.(\d+):(\d+):(\d+)/){
            $st_yr=$1;
            $st_mo=$2;
            $st_da=$3;
            $st_hr=$4;
            $st_mi=$5;
            $st_sc=$6;
            $start_time=$st_mo . "/" . $st_da . "/" . $st_yr . " " . $st_hr . ":"
. $st_mi;
        }
    }
    # We look for login lines, grab 3 pieces of data:
    # $1 is dispatcher, $2 is login type, $3 is ip address
    if(/login:\s+od=(\d+)\s+(\w+)\s+addr=(\d+\.\d+\.\d+\.\d+)/){
        $pending{$1}=1;
        $attempts{$1}++;
        $attempts_by_ip{$3}++;
        $attempts_by_type{$2}++;
        # track number of initial attempts by IP address.
        # More than one is problem
        $init_cnt{$3}++ if $2 eq "INITIAL";
        # Migration and isolation logins both come from isolated ep's.
        # Build list of both
        if($2 eq "ISOLATION"){
            $iso_count{$3}++;
            $isolations{$1}=$3;
        }
        if($2 eq "MIGRATION"){
            $migr_count{$3}++;
            $iso_count{$3}++ if $iso_count{$3};
            $migrations{$1}=$3;
        }
    }
}
```

```

    }

    $ip_addr{$3} = $1 if ! $ip_addr{$3};
    next;
}
# Build list of isolated endpoints by endpoint label
if(/Isolated endpoint is orphaned:\s\d+\.\(\d+\)\.\d+\+\s+\((\S+)\)/){
    $orphan_labels{$1}=$2;
    $orphan_ips{$1}=$isolations{$1};
    $orphans{$1}++;
}

if(/: resource `endpoint label: (\S+)' exists/){
    next if $exists_problems{$1};
    $prob_label=$1;
    $tmp=`wlookup -r Endpoint $1`;
    $tmp=~/$ENV{TMR}\.\(\d+\)\./;
    $exists_problems{$prob_label}=$1;
}
}
close(EPMGRLOG);
# We want to know about multiple IP addresses using the same
# dispatcher, so we look for this and build a hash
foreach $address (sort keys %ip_addr){
    next if $ip_addr{$address}==0; # skip initial logins
    $disps{$ip_addr{$address}}=$disps{$ip_addr{$address}} . "$address\n";
}
# Now remove from the hash those dispatchers with only one IP address
foreach $dispatcher (sort keys %disps){
    @list_of_ips=split(/\n/, $disps{$dispatcher});
    if (@list_of_ips == 1){ # no problem, delete
        delete($disps{$dispatcher});
    }
}
# Now we eliminate duplicate entries, i.e. an orphaned endpoint comes
# out of the isolation hash and migration hash.
foreach (keys %orphan_labels){
    delete($migrations{$_}) if $migrations{$_};
    delete($isolations{$_}) if $isolations{$_};
}
# Isolation logins that don't complete show up again as migrations,
# so zap those as well
foreach(keys %isolations){
    delete($migrations{$_}) if $migrations{$_};
}

# Finally, many isolations occur when multiple ip's use the same
# dispatcher--since these will be treated elsewhere, remove them
foreach (keys %disps){

```

```

delete($migrations{$_}) if $migrations{$_};
delete($isolations{$_}) if $isolations{$_};
}

&get_time;
print "=====\n";
print "Endpoint Management Report run on $timestamp\n";
print "Analysis of the region $ENV{TMR} Endpoint Manager log\n";
print "Log start: $start_time\n";
print "=====\n";
print "Three login types are recorded by the endpoint
manager:\n\tINITIAL\n\tISOLATION\n\tMIGRATION\n\nThe following number of
attempts were recorded:\n";

printf("%-15s %s\n","Login type","Total attempts");
foreach(sort keys %attempts_by_type){
    printf("%-15s %s\n",$_,$attempts_by_type{$_});
}
if(%init_cnt){
    print "-----\n";
    print "Analysis of INITIAL login attemps\n";
    print "-----\n";
    print "Endpoints ordinarily make only ONE initial login attempt.\nMultiple
attempts therefore represent a potential problem.\nONLY multiple attempts are
recorded in this report.\n\n";
    foreach (sort keys %init_cnt){
        if($init_cnt{$_} > 1){
            print "$_ made $init_cnt{$_} initial login attempts\n";
        }
    }
}
if(%migrations){
    $migration_count=keys %migrations;
    print "-----\n";
    print "Endpoint Migration Activity\n";
    print "-----\n";
    print "Migration logins occur when an endpoint is moved to a
different\ngateway. Ordinarily only one login attempt should be
needed.\nMultiple attempts from one endpoint should be investigated.\nONLY
multiple attempts are included in this report\n\n";
    print "Total endpoints attempting migration: $migration_count\n\n";
    foreach(sort keys %migrations){
        next if $attempts_by_ip{$migrations{$_}} == 1;
        printf("Dispatcher %6s (%s) made %s login
attempts\n",$_,$migrations{$_},$attempts_by_ip{$migrations{$_}});
    }
}
if(%isolations){
    $isolation_count=keys %isolations;

```

```

print "-----\n";
print "Endpoint isolation Activity\n";
print "-----\n";
print "Endpoints perform an ISOLATION login when they cannot contact\ntheir
primary gateway. This activity should be investigated.\n\n";
print "Total endpoints isolated: $isolation_count\n\n";
print "Isolated endpoints sorted by IP address:\n\n";
@iso_ips=sort values %isolations;
foreach(@iso_ips){
    printf("Dispatcher %6s (%s) made %s login
attempts\n",$ip_addr{$_},$_,$attempts_by_ip{$_});
}
}
if(%orphans){
    print "-----\n";
    print " Orphaned Endpoint Login Activity\n";
    print "-----\n";
    print "An orphaned endpoint is an endpoint whose database entry has\nbeen
deleted for some reason. Unexplained deletions should\nbe investigated.\n\n";
    @orphan_list=sort keys %orphans;
    $orphan_count=@orphan_list;
    print "Total number of orphaned endpoints observed in log:
$orphan_count\n\n";
    print "The following orphaned endpoints seem to be having a problem,\nas
evidenced by multiple isolation login attempts:\n\n";
    printf("%-20s %-15s %3s %s\n","Endpoint Label","IP address","Nr","Remarks");
    foreach(sort keys %orphans){ # key ($_ ) is the dispatcher number
        undef($found);
        if($orphans{$_} > 1){
            for $label(sort keys %exists_problems){
                if($orphan_labels{$_} eq $label){
                    printf("%-20s %-15s %3s Resource exists, disp
%s\n",$orphan_labels{$_},$orphan_ips{$_},$orphans{$_},$exists_problems{$label})
;
                    $found="yes";
                    last;
                }
            }
            printf("%-20s %-15s %3s
\n",$orphan_labels{$_},$orphan_ips{$_},$orphans{$_}) if ! $found;
        }
    }
}
if(%disps){
    $disp_count=keys %disps;
    print "-----\n";
    print "Dispatchers attempting logins using different IP addresses:\n";
    print "-----\n";

```

```

    print "A dispatcher number is unique to one host. Multiple IP
addresses\nattempting to use the same dispatcher should be investigated.\n";
    foreach $dispatcher (sort keys %disps){
        @list_of_ips=split(/\n/, $disps{$dispatcher});
        if (@list_of_ips > 1){
            $ip_count=$ip_count + @list_of_ips;
            print "\n$dispatcher\t";
            for($i=0;$i < @list_of_ips;$i++){
                if($i == 0){
                    print "$list_of_ips[$i] made $attempts_by_ip{$list_of_ips[$i]}
login attempts\n";
                }else{
                    print "\t$list_of_ips[$i] made
$attempts_by_ip{$list_of_ips[$i]} login attempts\n";
                }
            }
        }
    }
    print "Total IP Addresses: $ip_count using $disp_count dispatchers\n";
}
sub get_time{
    ($sec,$min,$hour,$mday,$month,$year,$yday,$isdst)=localtime(time);
    if($year && $month && $mday){
        $year = $year + 1900;
        $month++;
        if($month < 9){
            $month = "0" . $month;
        }
        if($mday < 9){ $mday = "0" . $mday;}
        $timestamp = $month . "/" . $mday . "/" . $year . " " . $hour . ":" . $min;
        return(1);
    }else{
        return(0);
    }
}
}

```

protect_epmgr.sh

Example B-4 shows the source code for the protect_epmgr.sh script.

Example: B-4 protect_epmgr.sh

```
#!/bin/sh
```

```
# This script will protect your Endpoint Manager when logins storms are
detected
```

```

# It will change your allow_install and select_gateway policy to reject any
login until
# the Endpoint Manager could process all its on-going logins
# DISCLAIMER: This script is provided as-is with no warranty of any kind.
# It has not been subjected to any formal review or regression testing.

. /etc/Tivoli/setup_env.sh

num_encry_minor=31
num_encry_major=65
timeout=20

logfile=/us/sduser/ALLOW/protect_epmgr.log
UP=`uptime`
echo $UP >> $logfile
date=`date`
echo $date >> $logfile

while true
do

    date2=`date`

    echo $date2 >> $logfile

    numero_encry=`odstat -cv | grep encry | wc -l`

    echo $date2 "Encrypted logins are: $numero_encry" >> $logfile

    if [ -f "/us/sduser/ALLOW/exit1.flag" ]

    then

        if [ $numero_encry -lt $num_encry_minor ]
        then
            `wputteppol allow_install_policy <
/us/sduser/ALLOW/allow_install_policy.pl`
            `wputteppol select_gateway_policy <
/us/sduser/ALLOW/select_gateway_policy_OK.pl`
            rm /us/sduser/ALLOW/exit1.flag
            echo "$date2: Policies loaded without exit1" >> $logfile
        fi

        if [ $numero_encry -gt $num_encry_major ]

    then

        if [ -f "/us/sduser/ALLOW/exit1.flag" ]

```

```

then
echo "Still finding encrypted logins" >> $logfile
else

    `wputepool allow_install_policy <
/us/sduser/ALLOW/allow_install_exit1.pl`
    `wputepool select_gateway_policy <
/us/sduser/ALLOW/select_gateway_policy_exit6.pl`
    touch /us/sduser/ALLOW/exit1.flag
    echo "$date2: Loaded policies with exit " >> $logfile
fi
fi

sleep $timeout

date3=`date`
echo $date3 "Encrypted logins are: $numero_encry" >> $logfile

done

```

metodos.pl

Example B-5 shows the source code for the motodos.pl script.

Example: B-5 *metodos.pl*

```

#-----#
----#
# Name:      metodos.pl          #
#
#
# Description:                    #
#   Script that reads Index.v5 in an endpoint, generates a zip file with all
# the methods needed. Be aware you need to stop an endpoint before unzipping
# the methods generated
#
# Version:   1.0
#
#
# Date:      09/01/2005
#
#
# Input file: c:\Tivoli\lcf\dat\1\cache\Index.v5
#

```



```

#
#
#-----#
----#

##### VARIABLES #####

$bindir="c:/Tivoli/lcf";
$fichindex="$bindir/dat/1/cache/Index.v5";
$fichindexbin="C:\\Tivoli\\lcf\\dat\\1\\cache\\Index.v5";
$zipfile="methods.zip";
$listfiles="c:/list.txt";
# This is for remote control
$levelsos2dir="C:\\Tivoli\\lcf\\pcremote\\os2-ix86\\LEVELS";
$levelsw2kdir="C:\\Tivoli\\lcf\\pcremote\\w32-ix86\\LEVELS";
$cont=0;

##### MAIN #####

#Operating system
@aux=`cmd /c ver`;
@aux=grep(/windows|\\2/i,@aux);
if ($aux[0] =~ /\2/){
    #OS/2
    $OS="OS2";
}
else {
    $OS="W2K";
}
print "Platform: $OS\n";

$pkzipc="C:\\\\util\\pkzipc.exe";
$pkzipc="C:\\util\\pkzip.exe" if($OS eq "OS2");
#deleting the zip file
unlink "$zipfile";

#stop endpoint and remote control (do the same before unzipping the methods)

`net stop lcf` if($OS ne "OS2");
`net stop tme10rc` if($OS ne "OS2");

#Generating a file where we insert index.v5
open (LIST,">$listfiles");
print LIST "$fichindexbin\n";

#Analizing each line of Index.v5 to insert the method in the zipped file

```

```

open (INDEX,"<$fichindex");
<INDEX>;#removing comments
while (<INDEX>) {
    $dir=(split(/\|/,$_))[8]; $dirbin=$dir;#get directory
    $fich=(split(/\|/,$_))[9]; $fichbin=$fich;#get file
    $dir=~tr/\|\|/;/;#Convert \| to /
    $fich=~tr/\|\|/;/;#Convert \| to /
    if (($OS eq "OS2") && ($dir =~/pcremote/i)){
        #El directorio PCremote lo añadiremos manualmente al zip, ya que
        #no aparecen todos los metodos en el fichero index.v5
        next;
    }
    if ((-e "$dir$fich")||(-e "$dir$fich.exe")) {
        #file exist and we included it
        #sometimes the .exe is missing for some methods
        $cont++;
        print LIST "$dirbin$fichbin\n" if(-e "$dir$fich");
        print LIST "$dirbin$fichbin.exe\n" if(-e "$dir$fich.exe");
    }
    else{
        #looking for more directories
        #i.e.: $dir= C:\tivoli\lcf\inv\scan and $fich=INV\SCAN\wepscan
        #need to search in wepscan
        $aux=$fich;
        $fich=~tr/^\\//; $fich=(split(/\\/, $fich,2))[1];
        $fichbin=~tr/^\\//; $fichbin=(split(/\\/, $fichbin,2))[1];
        while ((! -e "$dir/$fich")&&! -e "$dir/$fich.exe")&&($fich ne "")) {
            $fich=~tr/^\\//; $fich=(split(/\\/, $fich,2))[1];
            $fichbin=~tr/^\\//; $fichbin=(split(/\\/, $fichbin,2))[1];
        }
        $cont++ if($fich ne "");
        print LIST "$dirbin\\$fichbin\n" if(-e "$dir/$fich");
        print LIST "$dirbin\\$fichbin.exe\n" if(-e "$dir\\$fich.exe");
        print "ERROR: the file for the method does not existo $aux\n" if($fich
    eq "");
    }
}
if ($OS eq "OS2"){
# #We include the whole directory pcremote

    system("$pkzipc /add /dir=root /directories $zipfile
c:\\tivoli\\lcf\\pcremote\\");
}
else {
    opendir (DIR,"$bindir/PCREMOTE/w32-ix86/LEVELS");
    @dir=grep(!/^\.\/,readdir(DIR));#read dirs.
    closedir(DIR);
    foreach (@dir){

```

```

        print LIST "$levelsw2kdir\\$_\n";
        $cont++;
    }
}

close LIST;
close INDEX;
print "Creating zip file for methods\n";
system ("$pkzipc -add -dir=root $zipfile \@$listfiles") if($OS ne "OS2");
system ("$pkzipc /add /dir=root $zipfile \@$listfiles") if($OS eq "OS2");

$cont++;#adding 1 for Index.v5
print "Number fo compressed files: $cont (Including Index.v5)\n";
print " adding folder C:\\Tivoli\\lcf\\PCREMOTE\n" if ($OS eq "OS2");

unlink "$listfiles";

#Restarting endpoint and remote control if OS is W2K

`net start lcfd` if($OS ne "OS2");
`net start tme10rc` if($OS ne "OS2");

exit 0;

```

coord.sh

Example B-6 shows the source code for the coord.sh script.

Example: B-6 coord.sh

```

#!/bin/sh

# Script requires plan name as argument: coord.sh myplan
#
# This script may be used AS-IS
# For other databases than db2 use the proper syntax

nom_plan=$1

sql_file=/us/sduser/scripts/sql$$$.txt

echo connect planner > $sql_file
echo . >> $sql_file
echo "select NAME:s from ACTIVITY_PLAN where NAME='$nom_plan'" >> $sql_file

```

```

echo . >> $sql_file
echo quit >> $sql_file
echo . >> $sql_file

plan_name=`wrmsql < $sql_file | grep NAME | cut -d":" -f2 | sed s/" //"g`
rm $sql_file 1> /dev/null 2>&1

if [ -z "$plan_name" ]; then
    echo Plan $nom_plan does not exist. Please type another one
    exit 0
fi

# Getting plan rows

planner_db=BPLANNER
plan_user=sduser
planpwd=myplan
myschema=sduser
db2 connect to $planner_db user $plan_user using $planpwd > /dev/null

plan_id=`db2 "select plan_id from activity_plan where name='$1'"`

plan_id=`echo $plan_id| awk '{print $3}'`

echo $plan_id

rows=`db2 "select count(*) from act_layout where plan_id='$plan_id'"`

rows=`echo $rows| awk '{print $3}'`

#Need to repeat recursively until xx times
# Now we build a file coord_input.lst
# This file has the x and y coordenates to redraw the plan
# The new coordenates are updated in the table act_layout for the plan and its
activities
veces=`expr $rows / 12`

set -A x_coord 40 180 290 400 510 620 730 840 950 1060 1170 1280

rm /us/sduser/scripts/coord_input.lst
i=1

y_coord=0

until [ $i -gt $veces ]
do
    do
        x=0

```

```

until [ $x -gt 11 ]
do

echo "${x_coord[$x]}"$y_coord >> /us/sduser/scripts/coord_input.lst

x=`expr $x + 1`

done

i=`expr $i + 1`
y_coord=`expr $y_coord + 70`

done

cont=1

while read linea
do

x=`echo $linea |awk '{print $1}'`
y=`echo $linea |awk '{print $2}'`

db2 "update act_layout set x_coord=$x,y_coord=$y where plan_id='$plan_id' and
activity_id=$cont"

cont=`expr $cont + 1`

done</us/sduser/scripts/coord_input.lst

db2 commit

db2 terminate

```

del_plan.sh

Example B-7 shows the source code for the del_plan.sh script.

Example: B-7 del_plan.sh

```

#!/bin/sh
#
# del_plan.sh AS-IS script. Minimum Maintenance to remove a plan from the
Planner monitor
# as input requires a list of plan ids generated by wlstpln -l command

```

```

# pluser must have delete grant access to the planner rdbms
#
#
pluser=sduser
pspw=myplan
rdbms=bplanner

db2 connect to $rdbms user $pluser using $pspw

echo "Start deleting plans"
log=/SWD/delete_plan.log
list_plan=$1
fecha=`date`
echo $fecha >> $log
echo "Start deleting plans" >> $log

cat $list_plan | awk '{print $1}' > myplans.lst

for i in `cat myplans.lst`
do
nom_plan=`cat $list_plan | grep $i | awk '{print $4}'`

echo "$i deleting $nom_plan"
echo "$i deleting $nom_plan" >> $log

db2 "delete from act_plan_status where plan_id = '$i'"
db2 "delete from variable where plan_id = '$i'"
db2 "delete from plan_target_spec where plan_id = '$i'"
db2 "delete from act_status_tgt where plan_id = '$i'"
db2 "delete from activity_status where plan_id = '$i'"
db2 "delete from act_parameter where plan_id = '$i'"
db2 "delete from activity where plan_id = '$i'"
db2 "delete from activity_plan where plan_id = '$i'"

echo "$i deleted $nom_plan"
echo "$i deleted $nom_plan" >> $log

done

db2 commit

db2 terminate

```

gen_cat.sh

Example B-8 shows the source code for the gen_cat.sh script.

Example: B-8 *gen_cat.sh*

```
#!/bin/sh
#
#-----#
# Name:gen_cat.sh      #
#                   #
# Description: :Catalogs entries in local catalog. Need      #
#                   wdlspp and wdinstsp disconnected command line #
#                   #
# Path:c:/gestion/swd/gen_cat.sh#
#                   #
# Version:1.0         #
#                   #
# Fecha:25/02/2005   #
#                   #
# Input file:c:/packages/paquete_xxxx.spd#
#                   c:/gestion/swd/gen_cat.ini
#                   Format of gen_cat.ini: packagename.version.IC--- #
#                   #
# Output file:c:/packages/paquete_yyyy.spd#
#                   c:/gestion/log/gen_cat.log#
#-----#

dir_spd="c:/gestion/swd"
spd_file_ini="c:/packages/paquete_xxxx.spd"
paq_file="c:/gestion/swd/gen_cat.ini"
log_file=c:/gestion/log/gen_cat.log

fecha=`date`
echo $fecha > $log_file

# Load the Tivoli environment

speditor_lib_path=C:/Tivoli/swdis/1/speditor/w32-ix86/classes/./lib
speditor_bin_path=C:/Tivoli/swdis/1/speditor/w32-ix86/classes/./bin
LCF_BINDIR=C:/Tivoli/lcf/bin/w32-ix86/mrt

PATH="$PATH:$speditor_lib_path:$speditor_bin_path:$LCF_BINDIR:$LCF_BINDIR/./to
ols"

# Obtenemos el tipo de maquina y la lista de Paquetes que lleva instalados

if [ ! -f $paq_file ]; then
    echo Need the input file $paq_file >> $log_file
```

```

        exit 1
    else
        echo Ok. File $paq_file is available >> $log_file
    fi

while read i
do
    echo reading the file >> $log_file
    # Splitting package tag in name and version
    nom_ver_paq=`echo $i | awk '{print $1}'`
    nom_paq=`echo $nom_ver_paq | cut -d"." -f1`
    ver_paq=`echo $nom_ver_paq | cut -d"." -f2`
    estado=`echo $nom_ver_paq | cut -d"." -f3`

    # Creating the new spd
    spd_file=`echo $dir_spd/paquete_$ver_paq.spd`
    temp_file=`echo $dir_spd/paquete_temp.spd`
    spb_file=`echo $dir_spd/paquete_$ver_paq.spb`

    if [ ! -f $spd_file_ini ]; then
        echo File $spd_file_ini does not exist >> $log_file
        exit 1
    fi
    # We replace the tags nombre_xxxx and version_xxxx by name and version in spd
    template file
    cat $spd_file_ini | sed s/nombre_xxxx/$nom_paq/g > $temp_file
    cat $temp_file | sed s/version_xxxx/$ver_paq/g > $spd_file

    # Creating spb via spd
    wdcrtsp -o -f $spd_file $spb_file > /dev/null
    if [ $? -ne 0 ]; then
        echo Could not create $spb_file using the spd file $spd_file for the
        package $nom_ver_paq >> $log_file
        exit 1
    fi
    # Analyze the status of the package

    echo $estado | grep ICU > /dev/null
    if [ $? = 0 ]
    then
        wdinstsp -f -uy $spb_file > /dev/null
        if [ $? -ne 0 ]; then
            echo Error installing $spb_file for package $nom_ver_paq >> $log_file
            exit 1
        fi
    fi

    echo $estado | grep "IC---" > /dev/null
    if [ $? = 0 ]

```



```
then
wdinstsp -f $spb_file > /dev/null
if [ $? -ne 0 ]; then
    echo Error installing $spb_file for package $nom_ver_paq >> $log_file
    exit 1
fi
fi

# Removing temp files
rm $spd_file > /dev/null
rm $temp_file > /dev/null
rm $spb_file > /dev/null

done < $paq_file

exit 0
```

Archived

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described in this appendix.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246454>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246335.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
SG246454.zip	Zipped scripts referenced in the book

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	10 MB minimum
Operating system:	Microsoft Windows/Linux

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material ZIP file into this folder.

Abbreviations and acronyms

ADS	Automated Deployment Services
API	application programming interface
BDT	bulk data transfer
DC	domain controller
DNS	Domain Name System
GPO	group policy object
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
LSA	local security authority
MBSA	Microsoft Baseline Security Analyzer
PDA	personal digital assistant
PDC	primary domain controller
RIS	Remote Installation Service
SMIT	System Management Interface Tool
SPE	Software Package Editor
SSL	Secure Sockets Layer
SUS	Software Update Services
TMA	Tivoli management agent
TMR	Tivoli management region

Archived

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 392. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *All About IBM Tivoli Configuration Manager Version 4.2*, SG24-6612

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Tivoli Configuration Manager Version 4.2.2: User's Guide for Deployment Services*, SC23-4710
- ▶ *IBM Tivoli Configuration Manager: User's Guide for Software Distribution, Version 4.2.2*, SC23-4711
- ▶ *IBM Tivoli Configuration Manager Version 4.2.2: Planning and Installation Guide*, GC23-4702
- ▶ *IBM Tivoli Configuration Manager Release Notes*, GI11-0926
- ▶ *Introducing IBM Tivoli Configuration Manager*, GC23-4703
- ▶ *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide Version 5.1*, SC32-1359
- ▶ *Tivoli Management Framework Reference Manual, Version 4.1.1*, SC32-0806
- ▶ *Tivoli Management Framework Release Notes*, GI11-0890
- ▶ *Tivoli Management Framework: Planning for Deployment Guide*, GC32-0803

Online resources

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

- ▶ *Tivoli Field Guide: IBM Tivoli Configuration Manager Upgrade, Version 4.2.1: Guidelines for a Smooth Upgrade*
http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html
- ▶ *Tivoli Field Guide: The Rights Needed on a Windows Endpoint*
http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html
- ▶ *Tivoli Field Guide: Tivoli Assessment Tool (assess.pl) Guidelines for the use of the Assess tool and analysis of results obtained*
http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html
- ▶ *Gateway-Endpoint Communication Security*
http://www.ibm.com/software/sysmgmt/products/support/supp_tech_exch.html
- ▶ The latest version of browser Java plug-in download site
<http://java.sun.com/j2se/>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

\$root_user 352

Numerics

3.7.1-TMF-0075 22

A

access control list 357
Access Manager 8, 28, 30
Active Directory 10, 350, 356
Activity Plan
 Editor 4
 Editor window 5
 Monitor 4–6, 11
activity plan 4–6, 9, 11, 336, 340
Activity Planner 4–6, 13, 15–20, 31–32, 312–314,
336–338
AIX 5L machine 218, 229, 259, 266
 IBM Tivoli Access Manager WebSEAL V5.1
 266
 WebSphere Application Server V5.1 259
alternate gateway 324
apm.ini 338
APMHandler 338
application programming interface (API) 353
apply maintenance 41
architectural considerations 47

B

backup strategies 52
bash shell 107
Bulk Data Transfer (BDT) 28
busy endpoint 326

C

C shell 39
cache_global_target_info 337
cache_local_target_info 337
Cancel Pending status 340
Catalog recovery 336
Change Manager 6, 13, 15, 17–18, 20, 32,

313–314
Cisco routers 56
Command
 wmvrim 37
commonConfig.properties 349
complexity of the plan 339
COMPUTER table 335
conditioned activity 338–339
conditioned plans 339
Conditioning mechanism 340
conn_retry_interval 333–334
consistent install 41
creating
 deployment plan 30
custom install 43
custom server install 43

D

data handler input threads 342
Data Moving 10
DB2 UDB
 8.1 229
 client 229
default tablespace 43
dependency checking at submission 335
deploying components 31
deployment plan 30
deployment services 38
Desktop for Windows 45
DHCP server 350
differencing mechanism 340
directory 42
directory context 42
Directory Query 9, 276–277, 279–286
 ABBC_BR_Users 285
 ABBC_US_Users 284
 context 276, 282
 Library 9, 279–280
 Name 282, 284–285
DirectoryContext object 276–279
 Working 276
disable broadcasting 51
disconnected commands 336

- distribution deadline 334
- Distribution Manager 334
- Distribution Status Console 42
- DMS database 343, 345
- DMS RDBMS 342–343, 345
 - DEVICE table 342
- DNS 350
- Domain Local Group 352
- Domain Name System
 - server 28
- downcall 326
- downcall_limit 326
- dynamic link libraries 95

E

- element 337
- encrypted logins 323, 375–376
- encryption level 47
- endpoint caching 337
- endpoint health check 326–327
 - implementation 326
 - session 326
 - session timeout 326
- endpoint information 337
- endpoint label 105, 371, 373
 - isolated endpoints 371
- endpoint manager 107, 215, 315–318, 320–324, 337, 365, 370, 372, 374–375
 - max_epmgr_rpc_threads variable 318
 - thread 321
- endpoint status 326, 334
- endpoint_login_encrypted 321
- Enterprise Directory
 - integration 275, 279, 287
 - Query component 19, 314
 - Query Facility 9–10, 15, 17, 32
 - Query service 19–20
- Enterprise Directory Query Facility 43
- epcheck_interval 327
- epcheck_sess_timeout 326
- epmgr thread 322–323, 363–365
- exchange resources 52

F

- faster deployment 2
- FAT file system 40
- file descriptor 315–317, 361, 368
- firewall 57

- flexibility 2
- Force Reboot type 334
- foreign key 314
- forward and reverse resolution 209
- Framework dependency mechanism 312
- Framework V4.1.1 323–328
 - maximum concurrent endpoint logins 324
- Framework V4.1.1-TMF-0010 328
- fresh install 312
- FRESH subdirectory 43

G

- gateway 107
- gateway debug level 324
- gateway retry_ep_cutoff 334
- group policy object (gpo) 354

H

- health check parameters 326
- high-bandwidth links 333
- HTML-3 capable Web browser 313
- HTTPS protocol 8
- hub 49
- hub TMR 49, 51
- hub-spoke architecture 49
- hub-spoke configuration 314
- hub-wide management activities 49

I

- IBM agent 349
- IBM DB2 Universal Database 217
- IBM Tivoli
 - Access Manager 266, 268, 272–274
 - Access Manager policy server 267
 - Access Manager Runtime package 267–268
 - Access Manager WebSEAL 268, 274
 - Configuration Manager 1–2, 12, 14–21, 25, 27–30, 52, 311–315, 318, 323, 325, 332, 335–336, 339, 341, 343–344, 346, 348, 350
 - Configuration Manager component 17
 - Configuration Manager Release Notes 30
 - Configuration Manager V4.1 14
 - Configuration Manager V4.2 14, 20
 - Configuration Manager V4.2.1 19
 - Configuration V4.2.2 13
 - Directory Client 267–268
 - Directory Server 237–238, 245, 258–259, 267

- Directory Server configuration tool 245, 255, 258–259
- Directory Server installation 245
- Directory Server Version 5.2 238, 258, 276
- IBM Tivoli Configuration Manager
 - 4.1 13–14
 - 4.2 12–15, 19–20, 22
 - 4.2 FP01 17
 - 4.2.1 13, 17, 19
 - 4.2.2 19–20
 - 4.2.3 13, 20, 350
 - architecture 312
 - component 30
 - deployment 12
 - evolution 12
 - history 13
 - planning 31
 - release 12
 - Release Notes 30
 - upgrade 22
 - Version 4.2.2 218
- IBM Tivoli Configuration Manager components 31
 - for endpoints 33
 - for gateways 32
 - for managed nodes 32
 - for Tivoli server 31
- IBM Tivoli Configuration Manager V4.2 15, 17, 312, 314, 347, 349–350
 - Inventory component 22
- IBM Tivoli Configuration Manager V4.2.1
 - Fix Pack 01 338
 - Fix Pack FP01 312, 336, 338
 - Fix Pack FP02 335
 - Fix Pack FP03 17, 23, 349
 - FP01 340
 - Inventory component 23
 - Web Gateway 314
 - Web Gateway component 17, 314
- IBM Tivoli Provisioning Manager V3.1 350
- inetd mode 319
- information exchange 47
- Informix 38
- installation type 222, 233
- installing
 - AIX 5L endpoint 212
 - checking the installed products 201
 - DB2 Client on AIX 5L 229
 - demo 55
 - desktop install 44
 - endpoints 203
 - expertise required 27
 - gateway on Linux 156
 - GSKit 267
 - IBM Tivoli Directory Client 267
 - installation methods 40
 - Inventory Gateway 177
 - Java Virtual Machine 43
 - large enterprise 111
 - Linux endpoint 108
 - Linux managed node 158
 - OS/400 endpoint 214
 - planning 25
 - pre-install checks 38
 - Pristine Manager Gateway 183
 - proof of concept 55
 - required roles 33
 - Resource Manager Gateway 189
 - Scalable Collection Service 171
 - server install 41
 - small/medium enterprise 55
 - Software Distribution Gateway 195
 - Tivoli Access Manager 266
 - Tivoli Access Manager WebSEAL 266
 - Tivoli Configuration Manager Server on AIX 5L 114
 - Tivoli desktop 148
 - Tivoli server 41
 - Tivoli server on Microsoft Windows 56
 - Web Gateway 45
 - WebSphere Application Server 259
 - Windows endpoint 108
 - Windows endpoint locally 203
 - Windows endpoints 203
- InstallShield technology 99
- integrated desktop install
 - Change Manager GUI 44
 - components to install
 - Activity Planner GUI 44
 - Distribution Status Console 44
 - Inventory GUI 45
 - Software Package Editor 45
 - Tivoli Desktop for Windows 44
 - Tivoli Java components 44
- integrated install
 - benefits 41
 - hints and tips 38
 - installation types for Tivoli Configuration Manager 41

- integrated desktop install 44
- integrated endpoint install 45
- Java Virtual Machine 44
- overview 41
- pre-install checks 38
- server install 41
 - installation programs 43, 46
 - typical install 42–43
- Interim Fix 19
- intermittent networking problems 326
- INTERRUPTED state 332
- Inventory component 3, 15, 18–19, 22–23, 31–32
- Inventory RDBMS 332
- Inventory RIM object 38
- Inventory scan 7, 9, 13, 32, 318, 330, 342
- Inventory V4.0 tables 15
- Invoking a method 357
- IP address 28, 370–371, 373
- ISMP 41
- isolated endpoint 371
- ISOLATION login 320, 322, 373

J

- Java
 - 1.3 for Tivoli 42
 - Client Framework for Tivoli 42
 - RDBMS Interface Module 42
- Java components 42
- Java mandatory prerequisites 45
- JOB_RESULT table 348
- JOBQ thread 318, 329
- junction point 295

K

- keystore 277

L

- large checkpoint files 342
- latency period 319
- LDAP
 - concepts 237
 - configuration steps 245
 - creating the database 248
 - installation 237
 - LDIF file 255
 - populating the database 255
 - system requirements 237

- LDAP administrator
 - password 277–278
 - user cn 277
- LDAP configuration
 - tool 246
 - tool window 245
- LDAP tree 29, 237–238
- LDIF file 255
- lenient option 335
- less fragmentation 339
- Linux server 99
- local catalog 334
- local security authority (LSA) 355
- local Tivoli region 46
- login storms 323
- login type 370, 372
- low session 329, 331

M

- max_num_transactions 332
- mdist2_result method 333
- method download 312, 324, 328
- Microsoft 2000 Server 56
- Microsoft Baseline Security Analyzer 350
- Microsoft patching process 350
- MS_SQL 38
- multiple Tivoli regions 47
- multistart mechanism 339
- multi-TMR environment 51

N

- name registry 336–337
- network topology 49
- nofiles 317
- notification manager 329, 332, 338
 - queue indicator 336
 - restart timeout 332
- notify_interval 333
- NTFS file system 40

O

- odadmin odlist 167
- odb.log 325
- one-way connection 48
- operating system 11, 27, 31, 313, 315–316, 349, 353–354, 356
- optimized for lookups 237

Oracle 38
OS/400 endpoint 214
oserv 357
oserv thread 316–317, 361, 363–365
 MINIMUM number 365
 recommended number 365
oserv threads 317
oslevel -r 114

P

patches.lst 172
PDA 342, 344–345, 348
Planner Editor panel 339
Planner monitor 337, 339, 381
plug-ins 45
policy region 51, 280
populate an LDAP database 255
pre_loading_tmf_object_threshold 337
primary domain controller (PDC) 356
principal 357
pristine installation 324
Pristine Manager 10–11, 13, 17, 32–33, 313–314,
336, 349
Pristine Tool 350
production catalog 336
profile distribution 42
proxy machine 108

Q

QSHHELL 214

R

RDBMS considerations 34
RDBMS server 115
Red Hat Linux 56
Redbooks Web site 392
 Contact us xiii
reexec 108
region connection types 48
region number 47
Remote Installation Service (RIS) 33, 349
replace a process-level token 353
report_thread_limit 331
report_transaction_timeout 332
repqueue.dat 338
Resource
 Manager Gateway 42

Resource Manager 7–8, 15, 17–18, 20, 32, 303,
313–314, 342, 347
results collector 347
results converter 347
retry_ep_cutoff 334
rexec 158

RIM

ccm RIM object 42
change the managed node 38
considerations 36
delete 38
inv_query 42
invdh_1 42
MDist 2 RIM object 42
object 36
RDBMS considerations 34
re-create 38
requirements 36
vendor specification 38

RIS machine 350

rlogin 108

RPC thread 315, 317–318, 321, 329, 341,
361–363, 365, 368

 maximum number 316, 368
 recommended values 362

rpc_max_threads 316

RSTLICPGM 214

RSTOBJ 214

S

sample hub-spoke architecture 49

schema scripts 43

SD_CM_STATUS table 332

 too many locks 332

SD_INST table 335

Secure Sockets Layer (SSL) 277

server load 52

setup_aix.bin 43

setup_env 78

signature integration 335

simplified maintenance 41

simulate a package installation 336

single server installation 42

slow links 57

Software Distribution

 component 3

software distribution 1–4, 6–9, 13–21, 31–33, 312,
320, 327, 329–330, 332, 334–336, 338–339, 341,

- 344–345, 347–348
 - MDist 2 support 13
- Software Package
 - Editor 13, 15, 17, 33, 336
 - software package blocks 40
 - Software Update Services (SUS) 350
 - source host 32, 314, 329–330, 332–333, 335–336
 - sourcing the Tivoli environment 107
 - spd_eng 334
 - spoke 49
 - spoke Tivoli server 314
 - spoke TMR 50–51
 - SSL keystores 292–293
 - stable.xml 340
 - STATUS data 328
 - STRTMEEPT 216
 - Successful Target (ST) 340
 - swdis.ini 333
 - Sybase 38
 - synchronous 45
 - system performance goals 49
 - system-level backups 53
- T**
 - table SD_CM_STATUS 15, 314
 - tablespace 43
 - target resolution at activity level 337
 - Task Library tasks 336
 - TCP backlog 318–319, 328
 - tcpBackLog directive 319
 - THREAD Thread-11 346–347
 - threads 315
 - Tivoli
 - Management Framework 42
 - name registry 52
 - Tivoli Access Manager
 - setup menu 272
 - WebSEAL installation 266, 273
 - WebSEAL product 217
 - Tivoli backups 53
 - Tivoli Configuration Manager
 - 4.2.3 350
 - Tivoli desktop 33, 280
 - Tivoli Enterprise Console 49
 - Tivoli Enterprise Console integration 335
 - Tivoli Enterprise products 46
 - Tivoli environment 7–9, 30–31, 53, 278, 313, 316–317, 320–321, 360, 362, 370, 383
 - deployment plan 30
 - endpoint machine 8
 - enterprise directories 9
 - Tivoli Firewall Toolkit 28
 - Tivoli Inventory 13, 15, 18–19
 - Tivoli management agent (TMA) 8, 11
 - Tivoli Management Framework 1, 8, 21–22, 29–30
 - 4.1 22
 - Fix Pack 22
 - patch 22
 - recent levels 22
 - Reference Manual 318
 - Release Notes 30
 - rev. B 22
 - software requirements 30
 - upgrade Version 3.7 22
 - upgrade Version 4.1 22
 - Version 3.7.1 22
 - Version 4.1 22
 - Version 4.1.1 22
 - Tivoli management region (TMR) 8, 28, 275–276, 279, 313, 315, 352, 357, 360–363, 365, 367–368, 371–372
 - Tivoli physical topology 49
 - Tivoli remote access account 104, 355
 - Tivoli Remote Execution Service 108
 - Tivoli roles 357
 - Tivoli server 31, 53, 107, 313–314, 316–317, 322, 329–333, 335, 338, 343, 347
 - Tivoli server configuration 316
 - Tivoli Software
 - Distribution 13, 16, 18–20
 - Tivoli Software Distribution
 - 4.0 13, 21
 - 4.1 13, 21
 - Tivoli Web Gateway
 - database 343, 347
 - device management 345
 - directory 343
 - logging settings 349
 - tables 343
 - upgrade 314
 - Tivoli_Admin_Privileges 356
 - TivoliAP.dll 352–353, 355–356
 - tmersvd 354
 - TMF_object_cache 337
 - TMR boundaries 52
 - TMR connections
 - one way 48

- two way 49
- transaction 325
- transactions locks 325
- tunneling 28
- two-way connections 49
- Type of configuration
 - Default tablespaces and run schema scripts 43
 - None 43
 - Schema scripts only 43

U

- uninstall scripts 46
- uninstallation 46
- Universal Database (UDB) 217–219, 223, 228–229, 236
- unreachable endpoint 326
- upgrade 312
- UUID/GUID 350

W

- w4inslcf 214
- wapmplugin 147
- wchkdb 52
- wcommtsp 335
- wcrtgate 170
- wdel 38
- wdllsp 336
- Web Gateway 33
 - component 7
 - hardware requirements 290
 - install 45
 - software requirements 290
- Web Interface 7–8, 15, 17–18, 20, 33
 - component 18
 - service 18, 20
- Web objects 307
- WebSEAL 307
- WebSphere technology 8
- wep ls 216
- wep status 323
- wepscan 342
- wgateway 170, 324
- Windows 2000 10, 349, 352–354, 356
- Windows Server 2003 349
- Windows Server 2003, Datacenter Edition 349
- Windows XP 44, 349
- Windows XP Home Edition 349
- winstall 40

- winstlcf 203
- wloadiso 342
- wlsinst 201
- wmonpln 338
- wmsgbrowse 332
- wmvrim 38
- wpatch 40
- wrapper script 46
- wrimtest 146
- wrklnk 216
- wsetrim 97
- wsetrimpw 97
- wsettap 356
- wswdcfg 331–332
- wswdcfg -s 333
- wsyncsp 336
- wuninst 47

Archived



Deployment Guide Series: IBM Tivoli Configuration Manager

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Deployment Guide Series: IBM Tivoli Configuration Manager



Redbooks

**Step-by-step
deployment guide for
IBM Tivoli
Configuration
Manager**

**Best practices and
advanced tuning**

Real-life scenarios

IBM Tivoli Configuration Manager controls software distribution and asset management inventory in a multiplatform environment. It is designed for configuration, distribution, change, version, and asset management in a distributed computing environment. Working on top of IBM Tivoli Management Framework, IBM Tivoli Configuration Manager provides an integrated solution for managing complex, distributed enterprise environments.

This IBM Redbook introduces the IBM Tivoli Configuration Manager logical and physical components and covers detailed planning and implementation steps to deploy IBM Tivoli Configuration Manager in small-to-medium and large-sized environments, including IBM AIX 5L, Microsoft Windows, Linux, and IBM OS/400 systems.

In addition, we talk about best practices, advanced customization, and tuning topics for IBM Tivoli Configuration Manager.

This IBM Redbook will be useful for IT specialists responsible for implementing IBM Tivoli Configuration Manager V4.2.x in customer environments.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

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

**For more information:
ibm.com/redbooks**