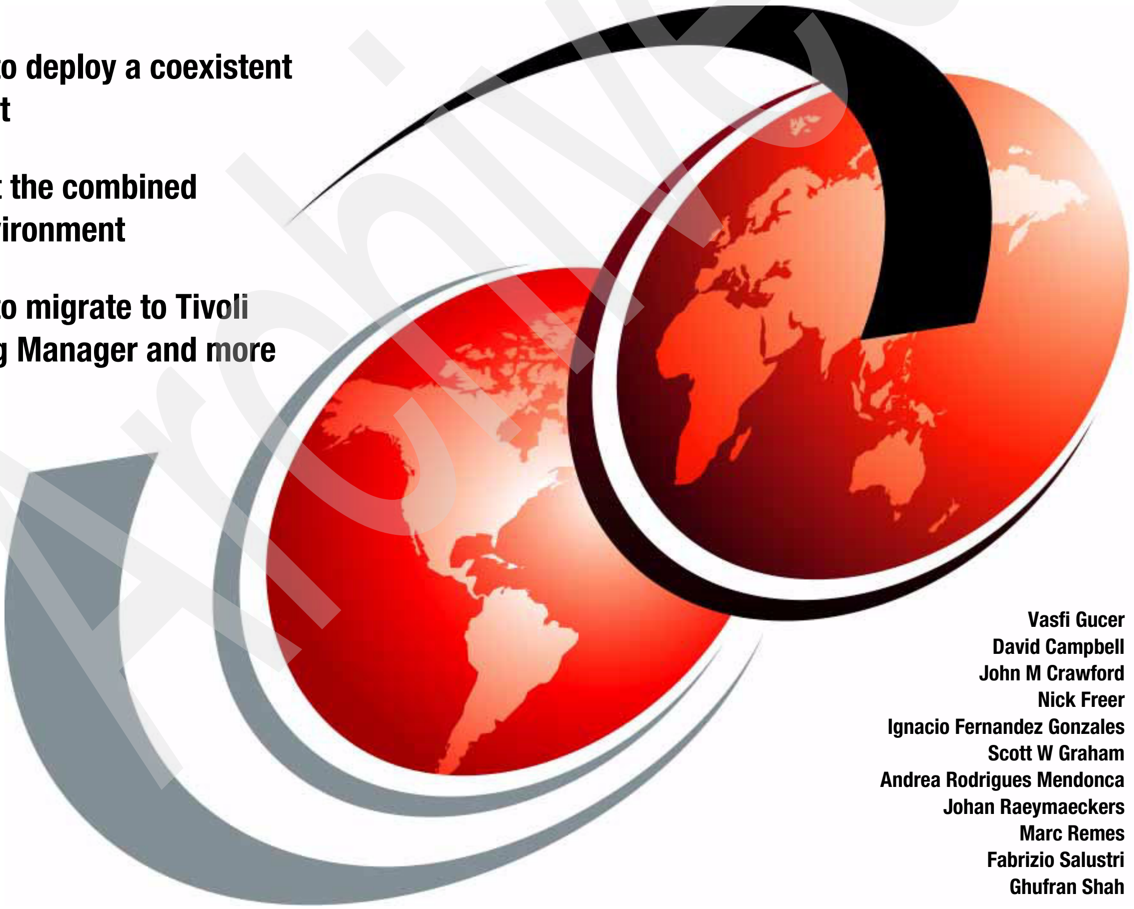
Tivoli. software

IBM

IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software Coexistence and Migration Considerations



Learn how to deploy a coexistent environment

Learn about the combined product environment

Learn how to migrate to Tivoli Provisioning Manager and more

Vasfi Gucer
David Campbell
John M Crawford
Nick Freer
Ignacio Fernandez Gonzales
Scott W Graham
Andrea Rodrigues Mendonca
Johan Raeymaeckers
Marc Remes
Fabrizio Salustri
Ghufran Shah

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software Coexistence and Migration Considerations

November 2007

SG24-7426

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

First Edition (November 2007)

This edition applies to IBM Tivoli Configuration Manager Version 4.3.3, Tivoli Provisioning Manager for Software Version 5.1, and Tivoli Provisioning Manager Version 5.1.

Contents

Notices

Trademarks

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

AIX®
Cloudscape™
DB2 Universal Database™
DB2®
IBM®
iSeries®
Library Reader™
pSeries®
Redbooks®
Redbooks (logo) ®
Tivoli Enterprise™
Tivoli Enterprise Console®
Tivoli Management Environment®
Tivoli®
Wake on LAN®
WebSphere®
zSeries®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Java, JumpStart, JDBC, JDK, JRE, JVM, J2EE, Solaris, Solaris JumpStart, Sun, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Internet Explorer, Microsoft, SQL Server, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

i386, Intel, Itanium-based, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries 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

This IBM® Redbooks® publication focuses on migration of a data center from a Tivoli® Framework environment and a Tivoli Configuration Manager-based management model to the new Service-Oriented Architecture (SOA), provided by Tivoli Provisioning Manager for Software V 5.1. In addition to migration, we also discuss the coexistence environment, where Tivoli Provisioning Manager for Software drives the Tivoli Management Framework environment.

Because Tivoli Provisioning Manager for Software shares its roots with Tivoli Provisioning Manager 5.1, all of the migration scenarios we describe in this book apply to Tivoli Provisioning Manager 5.1, as well.

To give you ideas for various migration possibilities, we describe several customer environments and then describe the suggested approach for moving to a Tivoli Provisioning Manager environment. We also tell you how the two environments can coexist during the migration process. We provide a feature-by-feature comparison between common Frameworks, Tivoli Configuration Manager operations, and their equivalent in Tivoli Provisioning Manager.

This book is a reference for IT Specialists who implement data center migration from a Tivoli Management Framework and Tivoli Configuration Manager-based management model to Tivoli Provisioning Manager for Software V 5.1 or Tivoli Provisioning Manager V 5.1.

The team that wrote this book

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

Vasfi Gucer is an IBM Certified Consultant IT Specialist working at the ITSO Austin Center. He worked with IBM Turkey for 10 years and has been with the ITSO since January 1999. He has more than 12 years of experience in systems management, networking hardware, and distributed platform software. He worked on various Tivoli customer projects as a Systems Architect in Turkey and in the United States. Vasfi is also a Certified Tivoli Consultant.

David Campbell is a Technical Consultant for the IBM Software Group Services for Tivoli in the United Kingdom (UK). He is a Senior IT Specialist and Tivoli Certified Consultant and has worked with Tivoli software both as a client and within IBM for 10 years. He uses many Tivoli products and now specializes in Tivoli Configuration Manager. He has worked with many UK and international clients including several of the UK's largest financial institutions. He lives in Essex in the UK.

John M Crawford is a Managing Consultant with IBM Software Services for Tivoli and has deployed and supported Tivoli products since 1998. A subject-matter expert in Tivoli Configuration Manager and software packaging for deployment, Mr. Crawford also has experience with Tivoli Framework and automation. Relatively new to the provisioning arena, he supported Tivoli Configuration Manager 4.2.3 for Patch Management and is studying to learn Tivoli Provisioning Manager 5.1 and Tivoli Provisioning Manager for Software 5.1. He is an IBM-Certified Information Technology Specialist.

Nick Freer is an IBM Certified Consultant Senior IT Specialist who worked for the IBM Software Group EMEA north region Tivoli Services for the last nine years. He worked on many Tivoli projects mainly in the UK as a Technical Consultant dealing with the core Tivoli products. Prior to technical consulting, he gained valuable experience by working for 11 years with UNIX® and Wintel systems and network management.

Scott W Graham is a member of the IBM/Tivoli Global Response Team with IBM Support and Services for Tivoli and has supported and deployed the Tivoli products since 1997. Mr. Graham worked for two years in Tivoli Level 2 Support and two years as a Tivoli Services consultant prior to joining the Global Response Team as an on-site troubleshooting expert, where he works with the whole breadth of Automation products. He is also the author of many widely-used classic framework tools, such as gw_tune.pl and tools for endpoint management.

Ignacio Fernandez Gonzales works for a major financial institution in the United Kingdom as a senior team member of the Systems Management and Monitoring Department in the ISD-CIO Division. His current interests are IT Provisioning and Service Management. In the past he worked as an IT Architect in Tivoli Services within the IBM Software Group for 10 years and gained extensive experience with large deployments of the IBM Tivoli suite of products. He played a key role as the Systems Management Team Leader during Sydney 2000 Olympic Games. He holds a master’s degree in Telecommunications Engineering from Universidad Politecnica de Madrid (UPM). Ignacio is a member of the Institute of Electrical and Electronic Engineers, Inc. (IEEE) and has published two patents by the U.S. Patent and Trademark Office.

Andrea Rodrigues Mendonca has worked with IBM Tivoli products since 1999. Since 2003, she worked for IBM Brazil in Global Technology Services, first as a member of the Strategic Outsourcing team and, in the last two years, as an IT Specialist in the Tivoli Support Group Level 1. She worked on Tivoli Management solutions, which involves Tivoli Configuration Manager deployment and on-going maintenance for several customers in Brazil and also in the United States, to include financial institutions.

Marc Remes is an IT Specialist working for IBM Belgium. He has more than 10 years of experience in Enterprise system management. Before entering the systems management field, he did application development and focused on communication protocols and human-machine interfaces. He is a Tivoli Certified Consultant with expertise in Tivoli Framework and Deployment. He worked on various large Tivoli client projects and is currently working in IBM Global Services.

Johan Raeymaeckers is an IT Consultant working for JorSy Systems Management in Belgium. He has six years of Enterprise Systems Management experience. He is a Certified Tivoli Consultant and Instructor. He worked on various large-scale Tivoli projects and is involved in Tivoli course development for IBM.

Fabrizio Salustri is a Software Support Specialist working for Italy ITM in Tivoli Customer Support within IBM Global Services. He has worked for IBM since 1996 and has extensive experience with the Tivoli products suite. Throughout his career, Fabrizio was involved in several projects to implement Tivoli solutions for clients of IBM Italy. Before joining the Tivoli Support team, he worked as a Certified AIX® System Administrator in AIX Technical Support. In March 2005, he got an IBM Tivoli Monitoring 5.1.1 Deployment Professional Certification and in April 2006 an IBM Tivoli Monitoring 6.1 Deployment Professional Certification.

Ghufran Shah is a Tivoli Certified Enterprise Consultant and Instructor with os-security.com in the United Kingdom. He has eight years of experience in Systems Development and Enterprise Systems Management. He holds a degree in Computer Science from the University of Bradford. His areas of expertise include Tivoli Systems Management architecture, implementation, and Tivoli training, together with business process improvement and return on investment modeling. He writes extensively about event management, monitoring, and business systems management integration and has taught Tivoli courses worldwide.

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

Arzu Gucer
International Technical Support Organization, Austin Center

KaTrina Love Abram
International Technical Support Organization, Atlanta

James Jennings, Steve Haertel, Michael McKay, Rajeeta Shah, Joe Ross, Patrick Woods
IBM USA

Mauro Arcese, Gianluca Bernardini, Massimiliano Celli, Michele Crudele, Alessandro Cupaiolo, Cristiana D'Amore, Maria Gallo, Antonio Gallotti, Luca Gimondo, Viviana Tripodi, Maercello Velati
IBM Italy

Harpreet Garcha
IBM Canada

Become a published author

Join us for a two-to-six week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Tivoli Provisioning Manager for Software: Introduction and architectural overview

In this chapter, we provide an overview of Tivoli Provisioning Manager for Software as well as an explanation of the purpose of the software. We also explain our approach to creating this book and a description of the lab environment we used to test the software against its advertised capabilities.

We also provide a comparison between Tivoli Provisioning Manager for Software and Tivoli Provisioning Manager 5.1 and define the software components of Tivoli Provisioning Manager. Additionally, we discuss the following topics:

- “Tivoli Provisioning Manager for Software new features” on page 7
- “Tivoli Provisioning Manager for Software components” on page 8
- “Server components and capabilities” on page 10
- “Scalable Distribution Infrastructure” on page 15
- “Tivoli common agent” on page 18
- “SOAP client” on page 20

Because this is a reference for migration from Tivoli Configuration Manager to Tivoli Provisioning Manager for Software, we discuss the expected configuration of the Tivoli Framework and the Tivoli Configuration Manager components, as well as the required patches and fix packs. Additionally, we discuss the Configuration Repository of Tivoli Configuration Manager.

1.1 Introduction to the covered material

Tivoli Provisioning Manager for Software is based on the code for Tivoli Provisioning Manager 5.1. This document focuses on the migration of a data center from a Tivoli Framework and Tivoli Configuration Manager-based management model to the new Service Oriented Architecture (SOA). Because Tivoli Provisioning Manager for Software shares its roots with Tivoli Provisioning Manager 5.1, a number of the definitions, explanations, process flows, and other diagrams are familiar to individuals who have read the ITSO publication *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261.

We wrote this manual using the Tivoli Provisioning Manager for Software General Availability (GA) code with a beta version of the Tivoli Provisioning Manager for Software Fix Pack 2 installed. According to development, Fix Pack 2 for Tivoli Provisioning Manager for Software and Fix Pack 2 for Tivoli Provisioning Manager 5.1 provided *leveling* of the two products at a common fix pack version. All of the code modifications in Tivoli Provisioning Manager 5.1 are included in Tivoli Provisioning Manager for Software.

1.1.1 Purpose of Tivoli Provisioning Manager for Software

Tivoli Provisioning Manager for Software was developed to provide a migration path from Tivoli Management Framework to a Scalable Distribution Infrastructure (SDI). SDI is based on the Operations, Administration, Maintenance and Provisioning infrastructure (OAMPi). The new approach provides the system management community with tools for the management of system resources and IT assets. It is a “life cycle” management approach that automates resource management at every stage.

1.1.2 Approach for meeting the scope of this book

In writing this publication, we approached the topic from various points-of-view. The common goal was to take the product as it was described and documented in release notes and other publications, install it in a lab environment along with a

Tivoli Framework environment, follow the suggested migration strategies, and document the process and steps throughout the process. This provides us with real-life scenarios that allow us to document any hints or tips gleaned from the process, as well as providing screens to capture and include here.

1.1.3 Overview of the lab environment setup for this book

We installed the lab environment we used to perform the migration from Tivoli Management Framework to Tivoli Provisioning Manager for Software on host name cairo, on the Microsoft® Windows® platform. Cairo ran Tivoli Management Framework version 4.1.1 with Tivoli Management Framework Fix Pack 5. We installed Tivoli Configuration Manager 4.2.3 with Fix Pack 4, including Software Distribution, Inventory, and Activity Planner.

The Tivoli Configuration Repository is comprised of a DB2® database installed on the Tivoli Management Framework server (cairo). The Tivoli Management Framework Server was configured as a ManagedNode, SourceHost, gateway, and endpoint. Five endpoints were installed and assigned to the Tivoli Management Framework and the Toronto gateway, which is also on the Windows platform.

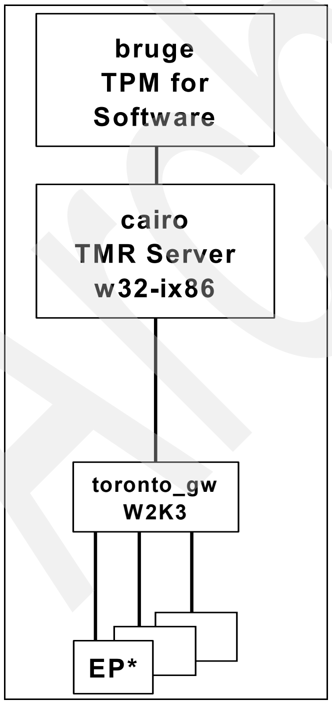


Figure 1-1 The lab environment used in the preparation of this manual

Example 1-1 through Example 1-5 on page 7 outline the specifics of the products, patches, and configuration of the lab systems as reported by the Tivoli Framework on each Tivoli Management Region Server.

The **wisinst** command, shown in Example 1-1, lists the installed products and patches or both products and patches in the Tivoli region.

Example 1-1 wlsinst -ah on cairo

```
*--------------------------------------------------------------------*
                                 Product List
*--------------------------------------------------------------------*


Tivoli Management Framework 4.1.1
    cairo          w32-ix86
    toronto        w32-ix86

Inventory Gateway, Version 4.2.3
    cairo          w32-ix86
    toronto        w32-ix86

Inventory, Version 4.2.3
    cairo          w32-ix86

Tivoli Java Client Framework 4.1.1
    cairo          w32-ix86

Java 1.3 for Tivoli
    cairo          w32-ix86

Tivoli Java RDBMS Interface Module (JRIM) 4.1.1
    cairo          w32-ix86

JavaHelp 1.0 for Tivoli
    cairo          w32-ix86

Activity Planner, Version 4.2.3
    cairo          w32-ix86

Distribution Status Console, Version 4.1.1
    cairo          w32-ix86

Software Distribution, Version 4.2.3
    cairo          w32-ix86

Software Distribution Gateway, Version 4.2.3
```

```
    cairo          w32-ix86
    toronto        w32-ix86

Software Distribution Software Package Editor, Version 4.2.3
    cairo          w32-ix86



*----------------------------------------------------------------------*
                                  Patch List
*----------------------------------------------------------------------*

Tivoli Framework Patch 4.1.1-LCF-0042  (build 11/13)
    cairo          w32-ix86
    toronto        w32-ix86

Tivoli Framework Patch 4.1.1-TMF-0061  (build 11/05)
    cairo          w32-ix86

Tivoli Framework Patch 4.1.1-TMF-0080  (build 11/05)
    cairo          w32-ix86
    toronto        w32-ix86

Activity Planner, Version 4.2.3, Fix Pack 4.2.3-TIV-APM-FP0004 (U810931
- 2007/03)
    cairo          w32-ix86

Scalable Collection Service, Version 4.2.3, Fix Pack
4.2.3-TIV-CLL-FP0004 (U8109381 - 2007/03)
    cairo          w32-ix86
    toronto        w32-ix86

Inventory, Version 4.2.3, Fix Pack 4.2.3-TIV-INV-FP0004 (U810931 -
2007/03)
    cairo          w32-ix86

Inventory Gateway, Version 4.2.3, Fix Pack 4.2.3-TIV-INVGW-FP0004
(U810931 - 2007/03)
    cairo          w32-ix86
    toronto        w32-ix86

Software Distribution Gateway, Version 4.2.3, Fix Pack
4.2.3-TIV-SWDGW-FP0004 (U810931 - 2007/03)
    cairo          w32-ix86
    toronto        w32-ix86
```

```
Software Distribution Software Package Editor, Version 4.2.3,  Fix Pack
4.2.3-TIV-SWDJPS-FP0004 (U810931 - 2007/03)
    cairo         w32-ix86

Software Distribution, Version 4.2.3, Fix Pack 4.2.3-TIV-SWDSRV-FP0004
(U810931 - 2007/03)
    cairo         w32-ix86

Tivoli MDist 2 Graphical User Interface 4.1.1 002 Patch
    cairo         w32-ix86

Tivoli Java Client Framework 4.1.1 Patch JCF411-0003
    cairo         w32-ix86

Scalable Collection Service, Version 4.2.3
    cairo         w32-ix86
    toronto       w32-ix86
```

The **odadmin** command, in Example 1-2, provides a command line interface to many runtime configuration setting settings and management operations for an object dispatcher.

Example 1-2 odadmin odlist on cairo

```
Region           Disp  Flags  Port            IPaddr   Hostname(s)
1713357986          1    ct-    94          9.3.5.56
cairo.itsc.austin.ibm.com,cairo
                    6    ct-    94         9.3.4.200
toronto,toronto.itsc.austin.ibm.com
```

The **wlookup** command, used in Example 1-3 and Example 1-4, searches the Tivoli name and registry for object information about a resource.

Example 1-3 wlookup -ar ManagedNode on cairo

```
cairo1713357986.1.347#TMF_ManagedNode::Managed_Node#
toronto1713357986.6.7#TMF_ManagedNode::Managed_Node#
```

Example 1-4 wlookup -ar Gateway on cairo

```
cairo-gw1713357986.1.590#TMF_Gateway::Gateway#
toronto-gw1713357986.6.66#TMF_Gateway::Gateway#
```

The **wep** command, in Example 1-5, performs actions on endpoint information that is contained in the endpoint list.

Example 1-5 wep ls (with interps added manually) on cairo

```
G     1713357986.1.590  cairo-gw
cairo-ep
klkzz9l-ep w32-ix86
toronto-ep w32-ix86
G      1713357986.6.66  toronto-gw
rp3410-ep hpux
rx2620-ep hpux
```

The rx2620 and rp3410 processors: rx2620 is an Itanium-based™ processor that is referred to as HP Integrity server. rp3410 is a PA-RISC-based processor that is referred to as HP PA-RISC server. Both rx2620 and rp3410 platforms are supported as an endpoint platform for IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software, which run in the coexistence mode.

In our coexistence tests, we used all of the endpoints in our lab with different interps types, and we did not get any problem specific to a platform.

1.2 Tivoli Provisioning Manager for Software new features

Tivoli Provisioning Manager for Software provides the following new capabilities:

- Operating system installation

  Automated processes to manage your operating systems, which includes installing operating system software and capturing and restoring disk images.
- Software installation

  Automated processes to distribute and install software to all the managed endpoints in your enterprise.
- Software patching and upgrade management

  Easily deploy needed software patches to computers in your enterprise and check the software installation status of target computers.
- Compliance checking and remediation

  Determine whether the software and security configuration matches your desired setup. If systems are noncompliant, Tivoli Provisioning Manager for

Software can make recommendations to fix the systems and, in some cases, automatically perform the recommended action.

- Hardware and software discovery

  Discover the presence and characteristics of system devices, such as hard disks, CD-ROM drives, USB devices, and network printers. Discover the presence of software based on a software signature or registry entry.

1.2.1 Tivoli Provisioning Manager for Software architecture

Tivoli Provisioning Manager for Software has a flexible architecture that is comprised of a central provisioning server and optional local depot servers. You can easily expand it to cover additional targets and new geographical locations. The dynamic content delivery service provides a centralized control for optimizing the use of resources when requests are received to upload or download files, for example during software distributions.

1.2.2 Tivoli Provisioning Manager for Software Web User Interface

A Web User Interface provides centralized access to all the tasks you can perform with role-based security, based on a directory (LDAP) server. It also includes an extensive set of predefined reports and facilities that enable you to define your own custom reports.

Web User Interface Editors

The Web User Interface includes a Software Package Editor and an Activity Plan Editor that are similar to those available in Tivoli Configuration Manager. The Editors enable you to be up-and-running on software distribution and activity planning tasks with minimum disruption after you move to a coexistence environment.

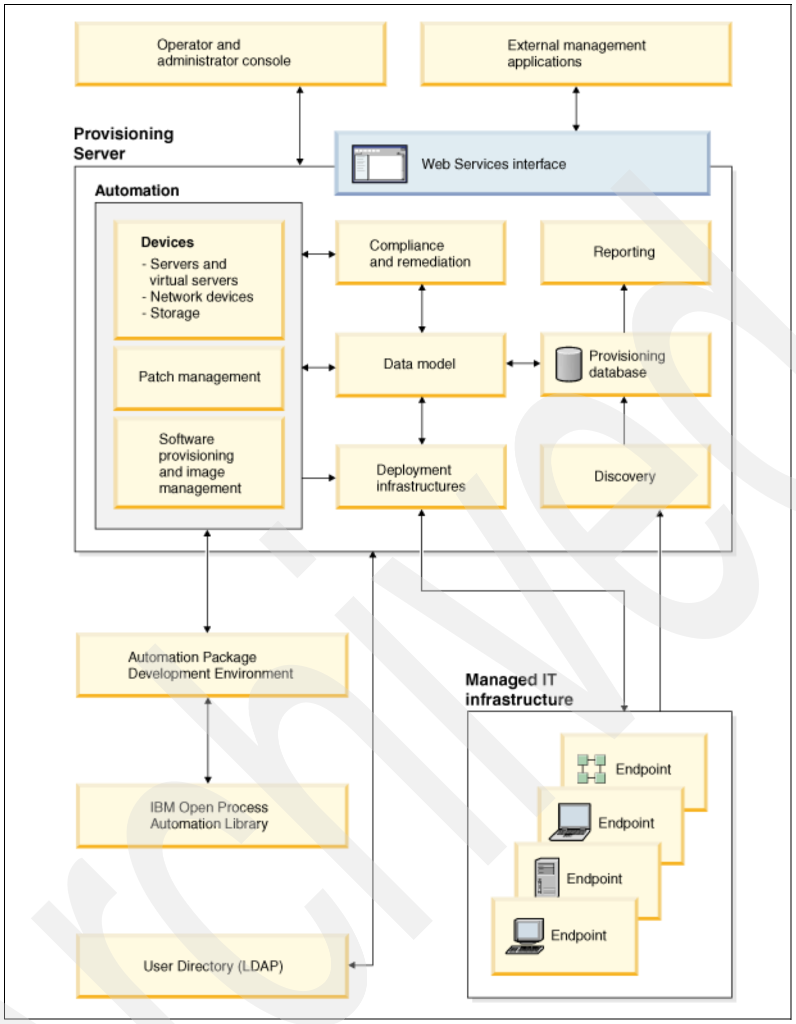
1.3 Tivoli Provisioning Manager for Software components

Tivoli Provisioning Manager for Software, like Tivoli Provisioning Manager, provides a new paradigm for managing IT systems and assets. When properly installed and configured, it provides you with a single interface to actions (called workflows in Tivoli Provisioning Manager) that were written to streamline day-to-day activities and to ensure that business policies are enforced. When running workflows to perform such activities as patching an operating system or installing new software on a server, you and the management of your organization can be confident that the same business process is being followed each time the workflow is executed. Well-written workflows can eliminate the possibility of errors, and the security model within Tivoli Provisioning Manager guarantees that only individuals with appropriate access levels can perform certain actions.

To provide the capabilities documented in this manual and in the *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261, the developers created a WebSphere® application that serves as your view into the system. Beneath the surface, Tivoli Provisioning Manager is comprised of a DB2 database that maintains information about all of the “assets” under management, and IBM Tivoli Directory Server maintains Tivoli Provisioning Manager security in a separate DB2 database. Additionally, the Automation Package Development Environment (APDE), which provides for writing custom workflows or modifying the workflows that are provided out of the box, is included with the software.

Important: The Automation Package Development Environment (APDE), which provides an easy way to write custom workflows or to modify the workflows that are provided out of the box, is only shipped with Tivoli Provisioning Manager. It is not included with Tivoli Provisioning Manager for Software.

Figure 1-2 on page 10 illustrates the main components of Tivoli Provisioning Manager and Tivoli Provisioning Manager for Software. It also shows how the components interact with your managed IT infrastructure and with each other.



Operator and administrator console
External management applications
Provisioning Server
Web Services interface
Automation
Devices
- Servers and virtual servers
- Network devices
- Storage
Patch management
Software provisioning and image management
Compliance and remediation
Reporting
Data model
Provisioning database
Deployment infrastructures
Discovery
Automation Package Development Environment
IBM Open Process Automation Library
User Directory (LDAP)
Managed IT infrastructure
Endpoint
Endpoint
Endpoint
Endpoint

Figure 1-2 Tivoli Provisioning Manager and Tivoli Provisioning Manager for Software components

The following sections provide a high-level reference of Tivoli Provisioning Manager for Software to which you can refer as you read the rest of the book.

1.3.1 Server components and capabilities

In this section we discuss the components of Tivoli Provisioning Manager for Software server.

Provisioning server

The provisioning server is the computer hardware on which Tivoli Provisioning Manager for Software is installed. The provisioning server contains the following sub-components:

- Provisioning database

  The provisioning database is the physical database for Tivoli Provisioning Manager for Software. It holds the data center model.

- Data center model

  The data center model (DCM) is a representation of all of the physical and logical assets that Tivoli Provisioning Manager for Software manages, such as computers, switches, load balancers, application software, VLANs, and security policies. It keeps track of the data center hardware and applications and monitors and records any changes to configuration. When a workflow successfully completes a requested change, the data center model is updated to reflect the current data center infrastructure.

  Additionally, the DCM can store information about allocated and non-allocated servers in resource pools for tier management. This information can include server identifiers, resource pool size, the number of active and idle servers, server priority, and similar information. Discovery and configuration management features also use the data model to identify configuration changes that are made outside of Tivoli Provisioning Manager for Software. You can review changes and use the change information to restore a data center asset to a previously known state.

  Note: To ensure that the DCM is a true representation of the physical data center configuration, use Tivoli Provisioning Manager for Software to make any changes to the environment.

  Although the discovery option can detect some environmental changes, there is a possibility that changes are made that discovery may not detect, which causes the DCM and the physical environment to be out of sync. With the physical environment out of sync, decisions made or actions taken based on information provided by Tivoli Provisioning Manager for Software can lead to errors.

- Automation Package

  An automation package is a collection of workflows, scripts, and other commands and tools that apply to the operation of a specific type of software component or a physical device. The deployment engine manages the deployment of workflows and associated components in an automation package. Tivoli Provisioning Manager for Software provides automation packages to automate the provisioning of software, patches, images and operating systems, and devices.

- Compliance and remediation

  Compliance management allows you to examine the software and security setup you have on a target computer (or group of computers) in your managed infrastructure. You can then compare that setup to your desired configuration in order to determine if they match. If they do not match, noncompliance occurs and recommendations (remediation) on how to fix the noncompliance issues are generated. Note that remediation is not necessarily automatic, but there are cases where you may want it to be. An example is detection that anti-virus software was either out-of-date or missing. Should this occur, Tivoli Provisioning Manager for Software can automatically apply an updating patch or reinstall the missing application.

- Reporting

  Reports allow you to retrieve current information about data center inventory, activity, and system compliance. Tivoli Provisioning Manager for Software reporting functionality includes:

  – Several predefined reports.
  – A Web-based query builder that allows you to customize existing reports or to create new ones.
  – Easier access to information in the data model through more than 40 predefined views.
  – Easier sharing of report definitions through enhanced import and export capabilities in the web interface.
  – Charts and graphs.
  – Schedule reports to run at a specified time, at a repeating interval, or both.
  – E-mail distribution or notification of reports.
  – Integration with third-party reporting software.

- Discovery

  Discovery provides automated processes that allow the administrator to find resources—as well as any changes to existing resources—within the IT infrastructure managed by Tivoli Provisioning Manager. Tivoli Provisioning Manager for Software provides the following discovery technologies:

  – Microsoft Active Directory® discovery

    Active Directory Discovery locates systems based on organizational unit (OU), Active Directory groups, and computer attributes defined in Microsoft Active Directory.

  – Tivoli Provisioning Manager for Software network discovery

    Network Discovery locates systems, their host names, and networking information. It also locates new devices and modifications made to existing Tivoli Provisioning Manager for Software managed devices.

    • Tivoli Provisioning Manager for Software Inventory discovery

      Inventory Discovery finds configuration changes in hardware, software, or both on managed devices.

  – IBM Discovery Library Reader™

    The Library Reader discovers new devices, modifications and removals of existing managed devices, as well as system configuration changes. Additionally, the Library Reader can discover new software on devices and any changes in relationships between managed resources.

- Deployment infrastructure

  Tivoli Provisioning Manager for Software supports reconfiguration and reallocation of resources in your managed environment using two different deployment infrastructures, a scalable software distribution infrastructure and a deployment engine infrastructure:

  – Scalable software distribution infrastructure

    The scalable software distribution infrastructure is based on Service Oriented Architecture (SOA). It provides standard services for performing software distribution and compliance activities in a scalable two or three tier implementation that can include branch office management. A deeper description of the Scalable Distribution Infrastructure implementation for Tivoli Provisioning Manager for Software is provided in section 1.3.2, "Scalable Distribution Infrastructure" on page 15.

  – Deployment engine infrastructure

    The deployment engine infrastructure is responsible for automated provisioning. It creates, stores, and runs workflows and communicates their success or failure in performing an operation. Workflows can be triggered by commands from you, by external tools that send SOAP commands to Tivoli Provisioning Manager for Software, or by recommendations from the policy engine. The Deployment Engine infrastructure is more appropriate to a small environment with a limited number of changes. The SDI provides a scalable and robust set of tools more appropriate to large organizations whose infrastructure may include one or more branch offices

- Web Services interface

  Web Services, including Web Services Resource Framework (WSRF) services, allow you to access the Tivoli Provisioning Manager for Software data center model directly rather than launching the Web interface. Using Web Services, you can access, manipulate, or change objects directly in the data center model.

  The command-line interface (CLI) provides access to Tivoli Provisioning Manager for Software features with SOAP. You have the flexibility to perform tasks, such as creating scripts that run specific SOAP commands or setting up external tools to send SOAP commands in response to an event.

- Operator and administrator console

  The Web-based operator and administrator console allows Tivoli Provisioning Manager administrators to interact with the Tivoli Provisioning Manager for Software provisioning server. The operator and administrator console provides a graphical representation of the data center, includes wizards to simplify configuration, and contains other features, such as reporting and task status tracking, which are not available from the command-line interface.

- Automation Package Development Environment

  The APDE is an Eclipse-based plug-in environment that automation package developers can use to customize existing automation packages or to create new ones. Note that APDE is shipped only with Tivoli Provisioning Manager, not with Tivoli Provisioning Manager for Software.

- IBM Open Process Automation Library

  The IBM Open Process Automation Library (OPAL) is an IBM-managed shared library of process automation that is available to individuals internal and external to IBM. It is a comprehensive online repository that contains more than 500 IBM Tivoli and Business Partners Product Extensions, including automation packages, integration adapters, agents, documentation, and supporting information.

- User directory

  Tivoli Provisioning Manager for Software integrates with several directory servers, allowing you to manage user accounts and user authentication using a directory server of your choice.

1.3.2 Scalable Distribution Infrastructure

The Scalable Distribution Infrastructure (SDI) for Tivoli Provisioning Manager for Software provides a scalable infrastructure for performing software distributions through Tivoli Provisioning Manager for Software. SDI is also known as the *Operations, Administration, Maintenance and Provisioning infrastructure*. If the software is distributed to peer endpoints in the same remote office or within a specific subnet range, Tivoli Provisioning Manager for Software leverages the entire infrastructure by using the targets as depots through software subagents that are optionally installed on targets outside the data center. By using the existing systems, you eliminate the need for dedicated hardware in remote locations, which reduces overall capital costs. The following list describes the main components of SDI:

- Tivoli Common Agent Services

  Tivoli Common Agent Services provide an infrastructure for managing computer systems, which enables secure connections between managed systems and storing information about the managed systems and the software running on them.

- Dynamic content delivery service

  Dynamic content delivery service (CDS) enables the efficient distribution of files and content to large numbers of targets through intermediate depot components, peer-to-peer distributions between agents, or both. Some publications may refer to CDS as DCDS or DCD. In any case they refer to the same functionality.

- Device management service

  The device management service (DMS) provides a solution for managing various devices by performing jobs that can be targeted to individual Tivoli

Common Agent (also referred as common agent or TCA) devices or to groups of devices.

Each component of SDI can perform its management activity through specific subcomponents installed on the Tivoli Provisioning Manager server and on the managed systems. Additionally, the concept of “Distributed SDI” is possible by co-locating the device management service that federates the server in the datacenter with the Tivoli Provisioning Manager server and placing additional device management service servers (known as eDMS servers) at points in the network (such as large branch offices). Due to the caching properties of a device management service server, this architecture puts the tasks closer to the targets and reduces WAN traffic.

Device management service federator

The device management service federator is installed on the provisioning server and is configured to act as a federated server. It implements a job distribution policy that pushes incoming jobs to remote agents. Jobs are actually submitted into the device management service federator to be sent to device manager subagents (installed on targets as a part on the common agent) through intermediate Federating Agent components. The results are returned in the reverse direction.

Device management service federating agent

The device management service federating agent periodically polls the federator server for jobs (default interval is 10 minutes) and any results that were sent back from common agent and subagents.

Although Figure 1-3 on page 19 shows a remote device management service federating agent—to give a complete picture of the way this architecture will evolve—currently only a single federating agent is implemented on the Tivoli Provisioning Manager server, while the remote federating agent will be supported in future releases. Federating agents are also referred to as federated agents.

Device manager subagent

The device manager client component is implemented as subagent of the common agent. It communicates with federating agents, polling for new jobs with a default value of 60 minutes. Device manager subagents are not shown in Figure 1-3 on page 19. They are actually installed on the target systems as part of the common agent.

Dynamic content delivery services management center

The dynamic content delivery services management center is the central component of the dynamic content delivery services and provides overall control of the other dynamic content delivery service components. In particular, it maintains a list of the files stored on each depot server and replicates files between depots. It also authorizes clients to download files and creates download plans.

Depot

A depot server is a system that stores files, which are ready for distribution to target systems, in a designated directory. Depot servers can also replicate these files to other depot servers to optimize network traffic. There must be at least one Upload Depot, which is also referred to as Preferred Upload Server, that replicates files to the other depots. Since it is installed as a Tivoli Common agent subagent and since common agent is not supported on a Tivoli Provisioning Manager server, a Tivoli Provisioning Manager installation always needs at least two separated systems in the central management environment: one for Tivoli Provisioning server and the other for the preferred upload server.

Dynamic content delivery services subagent

Clients are installed as common agent subagents on all the target managed systems. The subagent can request to download files from depot servers or from other clients (peers). In this case, the subagents work as miniature depot servers, which means they can hold copies of distributed files in a cache and act as sources for these files during downloads by their neighbors. Dynamic content delivery services subagents are not shown in Figure 1-3 on page 19. A dynamic content delivery service subagent installed on a common agent must contact a Management Center to request and receive download plans and notify the Management Center when a download from depot or peers is complete.

Common Agent Services agent manager

The Common Agent Services agent manager is installed on a provisioning server and provides functions that allow clients to obtain information about agents and resource managers. It also includes a registration service that provides authentication and authorization services and maintains a registry of configuration information about the managed computer systems in your environment. The registration service handles security certificates, registration, tracking of common agents, resource managers, status collection, and forwarding.

Tivoli common agent

Tivoli common agent (common agent or TCA), installed on depot servers and on targets systems, is a common container for all of the subagents. It provides shared system resources and secure connectivity. In Tivoli products, using common agent subagents is an increasing practice. Tivoli Provisioning Manager uses Tivoli common agent version 1.3.

Tivoli common agent subagents

The common agent's architecture is designed to allow code to be added to the common agent so that it becomes more powerful over time. The common agent subagents become the code that is added for a particular purpose. The subagent code may be used to enhance the discovery of hardware or software on a target or to provide management information to the systems administrator. The possibilities for enhanced capability are many and varied.

Peers

We discussed the concept of "peers" in the introduction to this section. It is a method by which existing systems can dynamically (and for a limited time frame) act as depots or repositories. This capability is an integral part of Scalable Distribution Infrastructure's dynamic content delivery service.

Figure 1-3 on page 19 shows the DMS components and the process flow during typical interaction between the Tivoli Provisioning Manager for Software server and other systems in the enterprise.

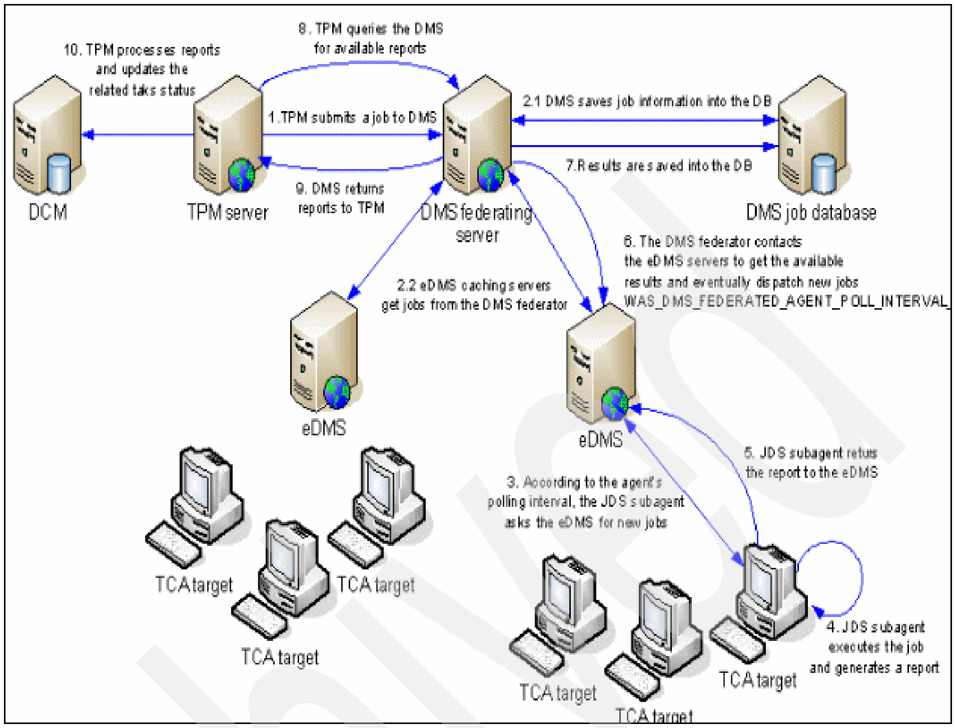

10. TPM processes reports and updates the related taks status
8. TPM queries the DMS for available reports
1.TPM submits a job to DMS
9. DMS returns reports to TPM
2.1 DMS saves job information into the DB
7.Results are saved into the DB
DCM
TPM server
DMS federating server
DMS job database
6. The DMS federator contacts the eDMS servers to get the available results and eventually dispatch new jobs WAS_DMS_FEDERATED_AGENT_POLL_INTERVAL
2.2 eDMS caching servers get jobs from the DMS federator
eDMS
eDMS
5. JDS subagent retus the report to the eDMS
3. According to the agent's polling interval, the JDS subagent asks the eDMS for new jobs
4. JDS subagent executes the job and generates a report
TCA target
TCA target
TCA target
TCA target
TCA target
TCA target

Figure 1-3 DMS components and process flow

Adaptive bandwidth control

Adaptive bandwidth control is a method that Tivoli Provisioning Manager for Software uses. It adjusts the sending rate of a depot to minimally interfere with other network traffic. The amount of network traffic is measured by watching the delay between when the sender sends a transmission control protocol (TCP) packet and when the TCP packet is returned from the server. In response to network traffic anywhere between the sender and receiver, the transfer rate is reduced if congestion is detected, and it is increased if congestion decreases.

We used the TCP Nice algorithm, which is available for review on the Web at the following address:

`http://www.cs.utexas.edu/~arun/pubs/nice.pdf`

In the case of two connections, a Nice peer connection and another generic TCP connection, the impact of the Nice transfer on the other network transfer is in the range of 10 percent. Without Nice, the impact of the peer connection on the other connection is in the vicinity of 50 percent.

SOAP client

As discussed in an earlier section, Tivoli Provisioning Manager for Software provides a Web-services interface to the DCM that can be compared, roughly, to the CLI that was available in the Tivoli Provisioning Manager for Software environment. The SOAP client makes tasks and operations available as Web Services through Windows Services Resource Framework (WS-RF).

You can invoke operations using the SOAP command line. Available SOAP commands are used for data center administration and for workflow invocation. You can submit SOAP commands from the Tivoli Provisioning Manager server or from a remote system that allows Tivoli Provisioning Manager for Software workflows to be invoked by other management systems, such as a help desk application or an in-house Web application to which users have access, in order to request software downloads or to perform on-demand inventory (discoveries) of their equipment to verify software compliance.

2

Tivoli Configuration Manager upgrade strategy and planning

In this chapter, we cover a methodology for upgrading the Tivoli Configuration Manager environment to Tivoli Provisioning Manager for Software using the coexistence features. We also discuss how customers with an existing Tivoli Configuration Manager environment can benefit from these features and plan for coexistence without affecting their current deployment.

2.1 Planning for coexistence

The fact that Tivoli Provisioning Manager for Software and Tivoli Configuration Manager can coexist is powerful. A methodology exists to transform the classic Tivoli Configuration Manager environment into the new Tivoli Provisioning Manager environment based on SOA. You can use coexistence to incrementally take advantage of the new technologies without sacrificing the familiarity with the classic technologies. This coexistence is useful because it gives you the ability to slowly gain the knowledge necessary to manage and administer the new technologies without requiring a complete switch in products.

As an administrator or implementer of a current Tivoli Configuration Manager environment, you must consider where IBM is going with its Configuration Management technology. Tivoli Provisioning Manager for Software is currently slated to replace Tivoli Configuration Manager, and most, if not all, customers can benefit by implementing a coexistence environment and learning the new technology prior to it being required. This section discusses what you should think about when planning your coexistence environment.

2.1.1 Analyzing your Tivoli Configuration Manager environment

How you use your Tivoli Configuration Manager environment is integral in planning how you can take advantage of coexistence with Tivoli Provisioning Manager for Software. It is critical that you consider the following concepts prior to deploying Tivoli Provisioning Manager for Software:

- How do you use the Inventory components of Tivoli Configuration Manager?
  - How often do you automatically scan with inventory?
  - What interfaces do you use to access inventory data?
- How do you use the Software Distribution components of Tivoli Configuration Manager?
  - What automation is involved?
  - How do you send software packages?
  - What features do you use, for example install, commit, rollback, and so on?
  - Is there an automated distribution mechanism?
  - How many of your clients rely on the Mobile Client?
  - What customizations, scripts, and interfaces have you developed to interface with Tivoli Configuration Manager?
- How do you use Activity Planner and CCM?

- How do you use the Patch Management features in Tivoli Configuration Manager 4.2.3?
- How many Tivoli tasks do you execute and what do they do?
- Do you currently use the Web Download features?
- What current deployment considerations do you have (for example, Branches and Servers versus Workstations, Roaming endpoints, and so on)?

Section 2.2, “Strategy for coexistence” on page 26, discusses the stages of deployment, which includes what you can and cannot migrate. We analyze the current Tivoli Configuration Manager environment to determine where you can receive immediate value from the Tivoli Provisioning Manager for Software coexistence. Most commonly, you will be able to provide a new user interface to your customers and new ways to consume inventory data. Eventually, you will also receive significant value for branch offices and roaming endpoints (notebooks).

2.1.2 Security considerations

One important aspect of implementing coexistence is your roles and users, which will change; therefore, you need to define new roles and users for the Tivoli Provisioning Manager for Software environment, even to do the same things you did in Tivoli Configuration Manager. For example, presuming you have Administrators, Software Distribution users, and Inventory users, you must create similar groups in Tivoli Provisioning Manager for Software.

Deploying the interface allows you to redesign user roles and groups. For more information about deploying the interface, see 2.2.1, “Stage 1: Using Tivoli Provisioning Manager for Software to drive Tivoli Configuration Manager” on page 27. It also gives you the opportunity to become familiar with these roles and lets you create meaningful groups that are applicable to your organization. We discuss these configurations in detail in Chapter 10, “Working in a firewall environment” on page 291.

2.1.3 Project considerations

As with any project, a good project plan is critical to success. We describe each stage of coexistence in 2.2, “Strategy for coexistence” on page 26; however, the following list includes the highlights that you should consider for a project plan.

- Testing environment

  First, you must deploy a testing environment so that mistakes and development efforts do not affect production. If you have a Test/QA Tivoli Configuration Manager environment, then point the Tivoli Provisioning

Manager for Software server to it. If not, then you can point the Tivoli Provisioning Manager for Software server to your production Tivoli Management Region (TMR) and “practice” driving activities through Tivoli Provisioning Manager for Software.

Note: You can point multiple Tivoli Provisioning Manager for Software servers, at the same time, to the Tivoli Configuration Manager environment This is useful for having a test Tivoli Provisioning Manager for Software and a production Tivoli Provisioning Manager. Also consider that in production, you should execute *all* activities through the Tivoli Provisioning Manager for Software server to ensure that you have one definitive source of data. Activities generated from Tivoli Configuration Manager do not synchronize with Tivoli Provisioning Manager for Software.

- Process design

  Use the test environment to redesign the activities for your customers to use Tivoli Configuration Manager. You need to redesign the user interaction with your Inventory Process, Software Distribution Process, and any tasks or activity plans. You can redesign the user processes in stage 1 without changing the underlying technology.

- User roles design

  Redesigning the user processes helps you define the necessary roles for your administrators and customers. Again, you can accomplish this in stage 1 without altering the underlying.

- Stage 1: Inventory scanning process

  Migrate all of your scheduled inventory scans in Tivoli Configuration Manager to Tivoli Provisioning Manager for Software Discoveries. This is the simplest and easiest task to migrate first and gives you experience using the Tivoli Provisioning Manager for Software to drive Tivoli Configuration Manager and access the new data model. Do not forget about how your inventory data is consumed. Make sure that you migrate the consumption of data from custom tools and web interfaces to the new Tivoli Provisioning Manager for Software data store.

- Stage 1: Software distribution process

  Migrate deployment of software packages to the Tivoli Provisioning Manager for Software interface using your new process. Deploy all new packages through Tivoli Provisioning Manager for Software, and determine how you can integrate compliance checking to ensure that all clients are at the appropriate levels.

- Stage 1: Activity plans and tasks

  Ensure that all tasks and activity plans are migrated to being executed through the Tivoli Provisioning Manager for Software interface. This is a good time to revisit many of the automated tasks and ensure that they are still necessary.

- Stage 1: Redesign automation

  Migrate automation that is implemented through Web interfaces and scripts that utilize the Tivoli Configuration Manager command line services and tasks to Tivoli Provisioning Manager for Software by driving these through the SOAP interface. This step is critical to ensuring that Stage 2 runs smoothly.

  Note: When you execute tasks through the SOA infrastructure, they do not occur immediately because the execution occurs when the common agent checks in the next time. This may significantly affect your current automation designs.

- Stage 1: Patch management

  If you are using Patch Management, it will be completely replaced and needs to be migrated as well. However since it uses the same fundamental technology, it should be familiar to you.

- Stage 2: Determine endpoint subsets

  At this point, you must determine how to subset your endpoints to receive the most value from common agents and dynamic content delivery service technology. Consider branch offices and mobile (mobile computers) endpoints.

- Stage 2: Deploy common agents

  Utilize the existing Tivoli Configuration Manager endpoints to deploy common agents. This is the simplest method for deployment and has the added benefit of providing two methods to deliver software to the endpoints.

- Stage 3: Continue migration

  Stage 1 and stage 2 should accommodate all of the process changes necessary to completely move to the new infrastructure. After they are complete, then you can deploy the common agents to the whole infrastructure.

2.2 Strategy for coexistence

The process we describe in this section allows you to gradually migrate to the new Tivoli Provisioning Manager for Software infrastructure using a staged approach that emphasizes coexistence and minimizes risk.

We define three stages for this approach:

- Stage1: Move one or more Tivoli Configuration Manager regions into a coexistence environment where configuration management operations are driven from the Tivoli Provisioning Manager for Software server, utilizing the Tivoli Management Framework for the distribution infrastructure.
- Stage 2: Gradually roll out the new scalable software distribution infrastructure and deploy the common agents while using Tivoli Provisioning Manager for Software to manage the coexistence environment from a single interface.
- Stage 3: All Tivoli Manager Framework agents have common agents and all configuration management operations are moved to the new Tivoli Provisioning Manager for Software infrastructure.

Table 2-1 includes the description and the objectives of each stage.

Table 2-1 Upgrade strategy stages

Stage	Description	Objective
Stage 1	Tivoli Provisioning Manager for Software using the Tivoli Management Framework for distribution: Install Tivoli Provisioning Manager for Software V5.1 and link to Tivoli Configuration Manager infrastructure.	Provide new capabilities without requiring migration to new infrastructure, which minimizes risk and flattens the migration learning curve. Preserve existing investment in Tivoli Management Framework infrastructure: profiles, activity plans, and logical grouping
Stage 2	Roll out new SDI infrastructure and common agent endpoints: Pilot of the SOA based infrastructure.	Enable gradual introduction of new managed endpoints in logical regions or branches. Deployment of new agents on all new hardware.
Stage 3	All Tivoli Management Framework infrastructure is duplicated with the Tivoli Provisioning Manager for Software infrastructure. Tivoli management agents all contain Tivoli common agents (TCA): Start real migration.	Reuse or migrate core assets for continued use in Tivoli Provisioning Manager for Software V5.1. Migrate existing Tivoli management agent endpoints (TMA) to TCA ones.

2.2.1 Stage 1: Using Tivoli Provisioning Manager for Software to drive Tivoli Configuration Manager

The first step when moving to Tivoli Provisioning Manager for Software is to bring your existing Tivoli Configuration Manager infrastructure within the management of the Tivoli Provisioning Manager for Software server (we refer to as the Provisioning server).

Tivoli Configuration Manager objects and data are replicated in the provisioning server data center model (DCM), which allows the provisioning manager server to work with endpoints, software packages, profile managers, and other objects that you defined in the Tivoli Configuration Manager environment.

The objectives of this stage are to:

- Take advantage of the Tivoli Provisioning Manager for Software Web Interface and new capabilities to increase the value of the Tivoli Configuration Manager deployment.
- Use the Tivoli Provisioning Manager for Software infrastructure as a learning platform and proof of concept for migrating operational tasks performed in Tivoli Configuration Manager to Tivoli Provisioning Manager.
- Begin planning and designing the complete migration away from Tivoli Configuration Manager.

New capabilities

Tivoli Provisioning Manager for Software offers many new features and capabilities that you can exploit to improve your current Tivoli Configuration Manager implementation. The initial stage of coexistence offers the following features:

- A new Web User Interface to perform existing Tivoli Configuration Manager V4.2.3 operations.
- New logical groups and dynamic groups that can coexist with imported profiles, inventory, software catalogs, and so on.
- New reporting engine and reports.
- Windows Patch Management.
- Compliance management.

Customer benefits

Tivoli Provisioning Manager for Software offers a new operations-oriented graphical user interface (GUI) that provides easy access to Tivoli Configuration Manager data and execution of activities.

Following are some of the customer benefits:

- Easier endpoint installation and reinstallation through Tivoli Provisioning Manager for Software workflows
- Improved reporting
- Improved desktop management
- Support for new capabilities utilizing unchanged Tivoli Configuration Manager Tivoli Management Framework infrastructure

Architecture

The Tivoli Management Framework architecture of servers, repeaters, gateways, and source hosts continues to be used as the distribution infrastructure.

Figure 2-1 shows an example of a complex Tivoli Management Framework architecture that connects multiple Tivoli Management Regions using the hub and spoke infrastructure. In the later figures that show the migration phases, the structure of the regions is simplified for readability, but you can bring both complex and simple architectures into the coexistence environment.

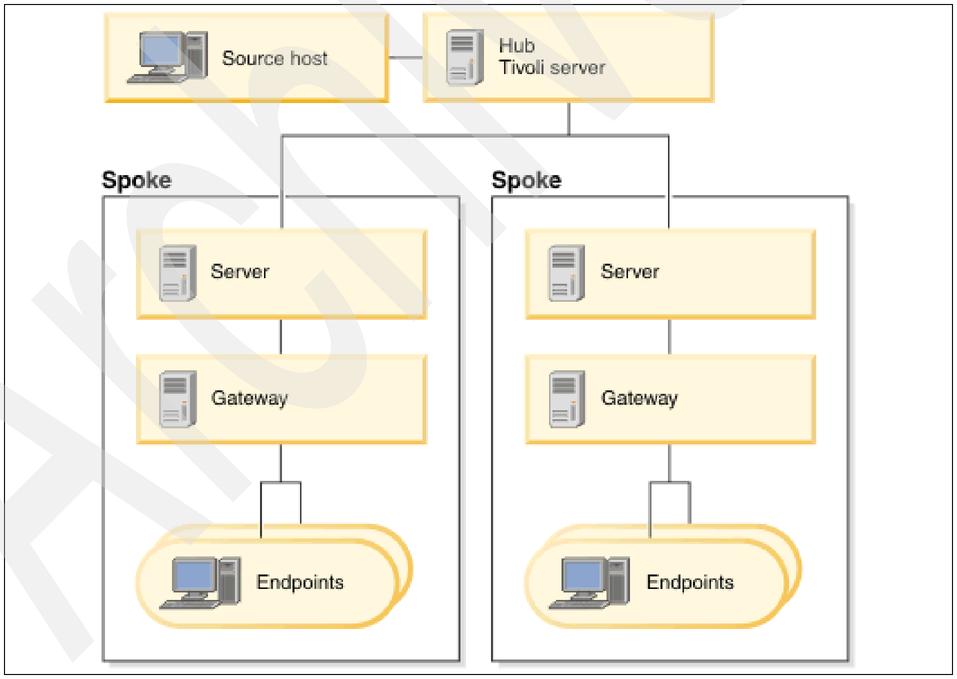


Figure 2-1 Hub and spoke architecture

Coexistence between Tivoli Configuration Manager for Software and Tivoli Management Framework depends on the definition of a bridge between the Tivoli Management Framework infrastructure and the Provisioning server.

You define the bridge by providing Tivoli Provisioning Manager for Software with information about the servers and databases in one or more Tivoli Management Regions.

Figure 2-2, shows a coexistence environment on which a Tivoli Management Region (TMR B) can be recognized and managed by a provisioning server. In this scenario, Tivoli region TMR A has not yet been added to the coexistence environment and can only be managed from the Tivoli desktop.

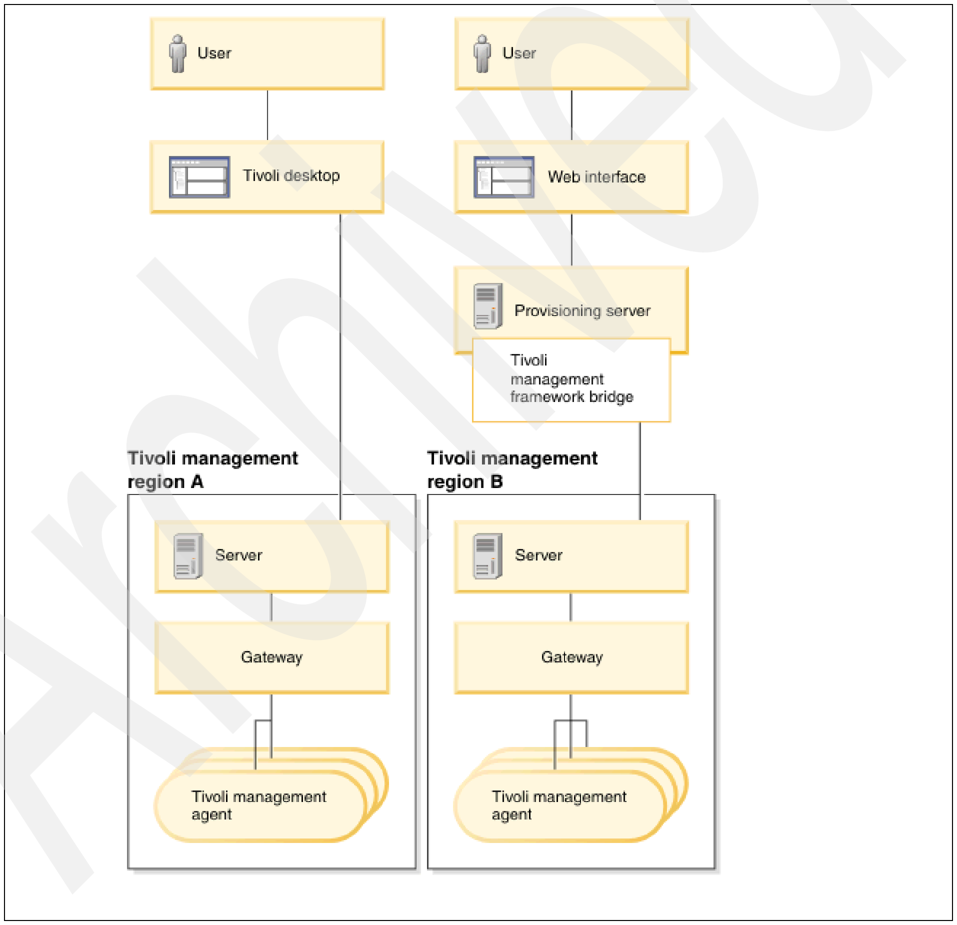


Figure 2-2 Coexistence between Tivoli Provisioning Manager for Software and Tivoli Management Framework topology

To bring a Tivoli region into a coexistence environment, as shown in Figure 2-2 on page 29 for TMR B, you must complete the following tasks:

- Provide the Provisioning server with information about Tivoli Configuration Manager servers and databases.

  Using tasks on the Tivoli Provisioning Manager for Software server Web User Interface, you can create objects that represent the servers and databases that are configured in your Tivoli Management Framework environment. Options are available to create hub and spoke servers and their associated inventory and activity planner databases.

  You can complete the definition of your Tivoli Management Framework environment in stages. Each stage must include at least a hub server, any spoke servers that are related to it, and the associated databases.

- Populate the Tivoli Provisioning Manager for Software DCM with replications of the objects related to the Tivoli Configuration Manager servers and databases.

  After you create the objects for at least one hub server and its associated spoke servers and databases, you can import information about Tivoli Configuration Manager objects that are associated with the servers and databases.

  Using the Provisioning server Web User Interface, you create a Discovery Configuration to replicate the information in the Tivoli Configuration Manager databases. When you run the Discovery task, the objects discovered in the database are imported into the Tivoli Provisioning Manager for Software DCM and mapped to Tivoli Provisioning Manager for Software object types. For example, a Software Distribution profile becomes a Software Definition and an Inventory profile becomes a Discovery Configuration.

- Migrate user login credentials and role authorizations to the LDAP server.

  In Tivoli Provisioning Manager for Software, authentication of user credentials and authorization of users to perform tasks on the Web User Interface are controlled by an LDAP server. This provides a centralized mechanism for access control that can be reused for other applications.

  Supported LDAP servers are IBM Tivoli Directory Server for both UNIX and Windows Operating Systems. In a Windows environment you can optionally use a Microsoft Active Directory installation.

  During the process of migration to a coexistence environment, details of user credentials and security roles that are stored in the Tivoli Management Framework database are extracted, and, after checking and modifying, are imported into the LDAP server.

Profile manager and role objects from the Tivoli Management Framework database are replicated in the Tivoli Provisioning Manager for Software DCM as access groups and access permission groups. An access group defines the types of objects that can be worked with. The access permission groups that are associated to an access group define groups of individual objects that can be worked with.

When the bridge between Tivoli Management Framework and Tivoli Provisioning Manager for Software is established for one or more Tivoli Management Regions, you can manage the Tivoli endpoints from the Tivoli Provisioning Manager for Software Web User Interface.

The objects that you require to continue software distribution, activity planning, and inventory tasks are available from the Tivoli Provisioning Manager for Software Web User Interface. You do need to redefine any inventory scan schedules that you had defined because the schedules are not replicated. In addition, you have access to an extensive set of reports that you can use to perform software compliance checks.

Figure 2-3 on page 32 shows a mixed environment that includes a Tivoli Management Region that has not been brought into the coexistence environment (TMR A), a Tivoli Management Region that has been brought into the coexistence environment (TMR B), and computers where the common agent is installed, which are within the Tivoli Provisioning Manager for Software scalable software distribution infrastructure. You might want to use this type of gradual approach to allow users to become accustomed to working on the Tivoli Provisioning Manager for Software Web User Interface.

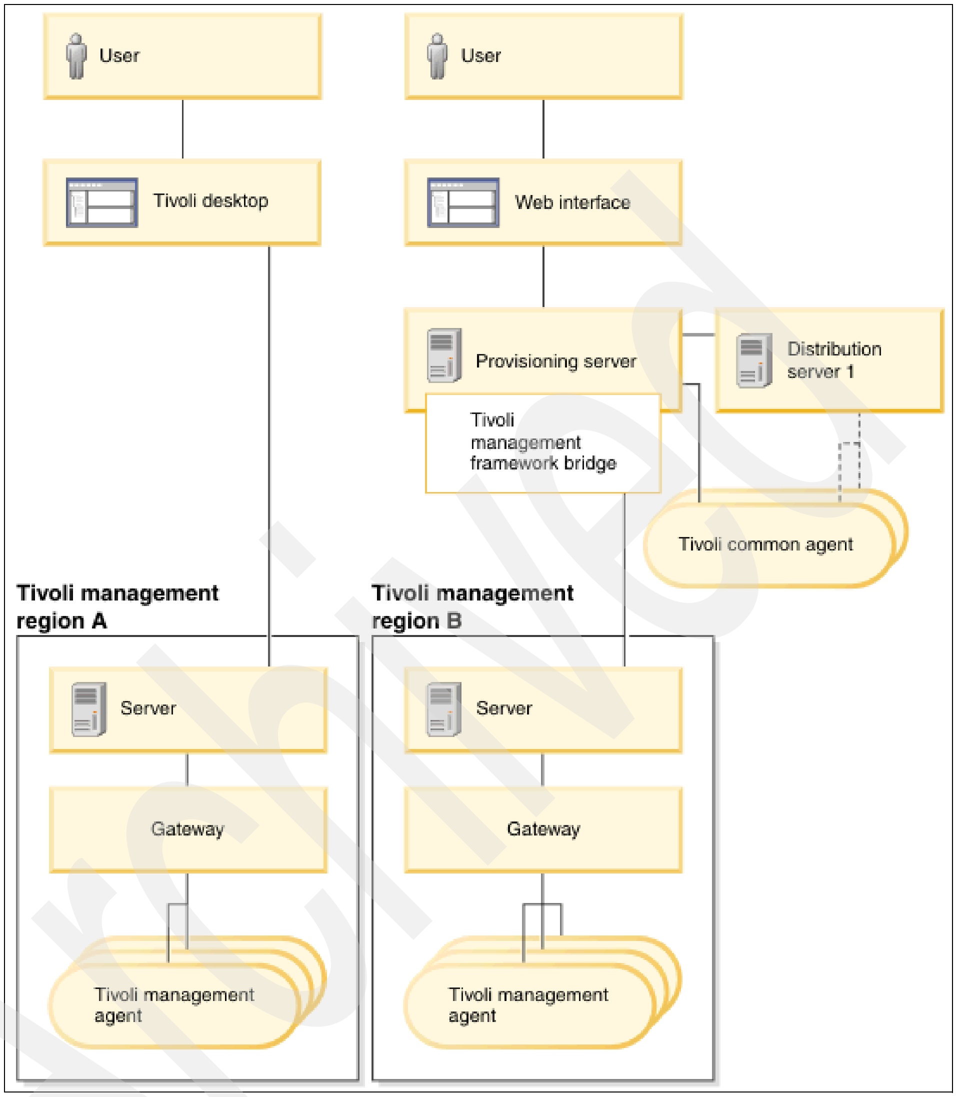


Figure 2-3 Working on both Tivoli endpoints and common agents

TMR A continues to be managed from the Tivoli desktop. TMR B is now managed from the Tivoli Provisioning Manager for Software Web User Interface.

Although the Tivoli Management Framework infrastructure is still in place, including the Tivoli desktop, avoid using the Tivoli desktop to manage the endpoints in TMR B because problems could occur if the same operation is performed from both the Tivoli Provisioning Manager for Software Web User Interface and the Tivoli Desktop.

Using the Web User Interface, you can define and perform software distribution, activity planning, and inventory operations that include the Tivoli management agents in TMR B and the common agents that are connected to distribution server 1.

Working in the coexistence environment, you can do the following:

- Perform software distributions, activity planning, and inventory tasks from the Tivoli Provisioning Manager for Software interfaces.

  The Web User Interface provides integrated access to many of the tasks that are available from the Tivoli desktop, and a more complete set of software distribution tasks are available if you installed the Eclipse plug-in Software Package Editor. The names assigned to tasks within Tivoli Provisioning Manager for Software, both on the Web User Interface and the Software Package Editor, differ from those used on the Tivoli desktop.

  For example, Software Distribution becomes Software Management, and Inventory becomes Discovery.

- Define target lists that include computers that are running the common agent and computers that are running the Tivoli management agent.

  In a mixed environment, operations such as software distributions and inventory scans can include targets that have either of the supported agents installed. For example, in the scenario in Figure 2-3 on page 32, a list of targets for a software distribution can include endpoints in TMR B and the computers running the common agent attached to distribution server 1.

- Use the extensive reporting capabilities of the Web User Interface Reports, which allows you to retrieve current information about data center inventory, activity, and system compliance.

  The Tivoli Provisioning Manager for Software reporting functionality includes:

  - Numerous predefined reports.
  - A Web-based query builder, which allows you to easily customize existing reports or create new reports.
  - Easier access to information in the DCM through more than 40 high-performance views.
  - Easier sharing of report definitions through enhanced import and export capabilities in the Web User Interface.
  - Charts and graphs.
  - The ability to schedule reports to run at a later time including repeating intervals.
  - E-mail report distribution and notification.
  - Integration with third-party reporting software.

When moving to the coexistence environment, part of the Tivoli Management Framework queries are replicated and are available from the reports menu on the Web User Interface.

- Perform software compliance checks and automatically resolve out-of-compliance situations.

  Software compliance checks are used to check whether certain software applications should be on a computer or not. You can create software compliance checks for software groups, software products, software patches, and software stacks.

  When the compliance check runs, it compares the compliant state that was defined against the actual state of computers retrieved by the most recent inventory scan. If it discovers out-of-compliance situations on computers, it generates recommendations for bringing the computers to a compliant state. You can review out-of-compliance situation recommendations for remediation.

  If the software that caused the out-of-compliance situation has an associated software template, you can automatically perform the recommended installation, upgrade, or uninstallation. A software template identifies a software resource and its associated configuration details that you can use to create a software resource on a managed computer.

2.2.2 Stage 2: Using Tivoli Provisioning Manager for Software to manage the Tivoli Configuration Manager and SDI infrastructure

After you move to the coexistence environment, the next step is to make a plan to gradually migrate to a full implementation of Tivoli Provisioning Manager for Software.

Tivoli Configuration Manager objects (endpoints and repeaters) are gradually migrated to TCAs and SDI depots. This allows tasks driven through Tivoli Provisioning Manager for Software to interact with both TCAs and Tivoli Configuration Manager endpoints.

The objectives of this stage are to:

- Use existing infrastructure to deploy TCAs and depots
- Become familiar with and begin utilizing new SDI infrastructure
- Use the new Tivoli Provisioning Manager for Software Security Compliance features

New capabilities

Tivoli Provisioning Manager for Software offers many new features and capabilities for TCA endpoints using the SDI infrastructure. It is also very easy to exploit the new functionality quickly by utilizing the existing infrastructure:

- New bandwidth throttling and peer-to-peer distribution
- Highly scalable depot system with better failover than existing GW infrastructure
- New basic compliance function
- “Daytime” software distributions possible with new bandwidth throttling and peer-to-peer
- Large file transfer capability

When you upgrade to a full implementation, you can take advantage of all the functions available from the Tivoli Provisioning Manager for Software Web User Interface, from the scalable software distribution infrastructure, and from the common agent, for example:

- Security Compliance management and remediation

  Security compliance checks can be used to check for a variety of security issues, such as whether appropriate passwords or improper settings were set. There are a series of predefined security compliance checks for security requirements that are generally applicable. For example, you can check for the existence of hard disk and power-on passwords or for the values of firewall settings. You can also define your own security checks to cover situations that are not included in the predefined list.

  When a security check is run against a group of computers, it identifies out-of-compliance situations that you can then act to resolve.

- Optimized use of distribution resources

  A dynamic content delivery service management center is installed on the Tivoli Provisioning Manager for Software server and maintains a data set of record for the bulk data to be distributed to the depots. The dynamic content delivery service management server implements a distribution policy that sends bulk data to some or all of the distributed depot servers in advance of when it is needed.

  A number of regional distribution servers ensure the communication between the management server and the clients installed on the endpoints. The distribution servers communicate to the management server to get data requested by agents. Clients installed as subagents on all the managed systems or endpoints at the regional branch request to download files from distribution servers or from other clients.

If you are running in a multi-homed environment made of a hub-spoke architecture, you can minimize disruption of your plan to migrate one region at a time by ensuring that all migration-related problems are resolved before you begin to migrate the next region.

Figure 2-4 shows an environment on which the gradual migration process has started.

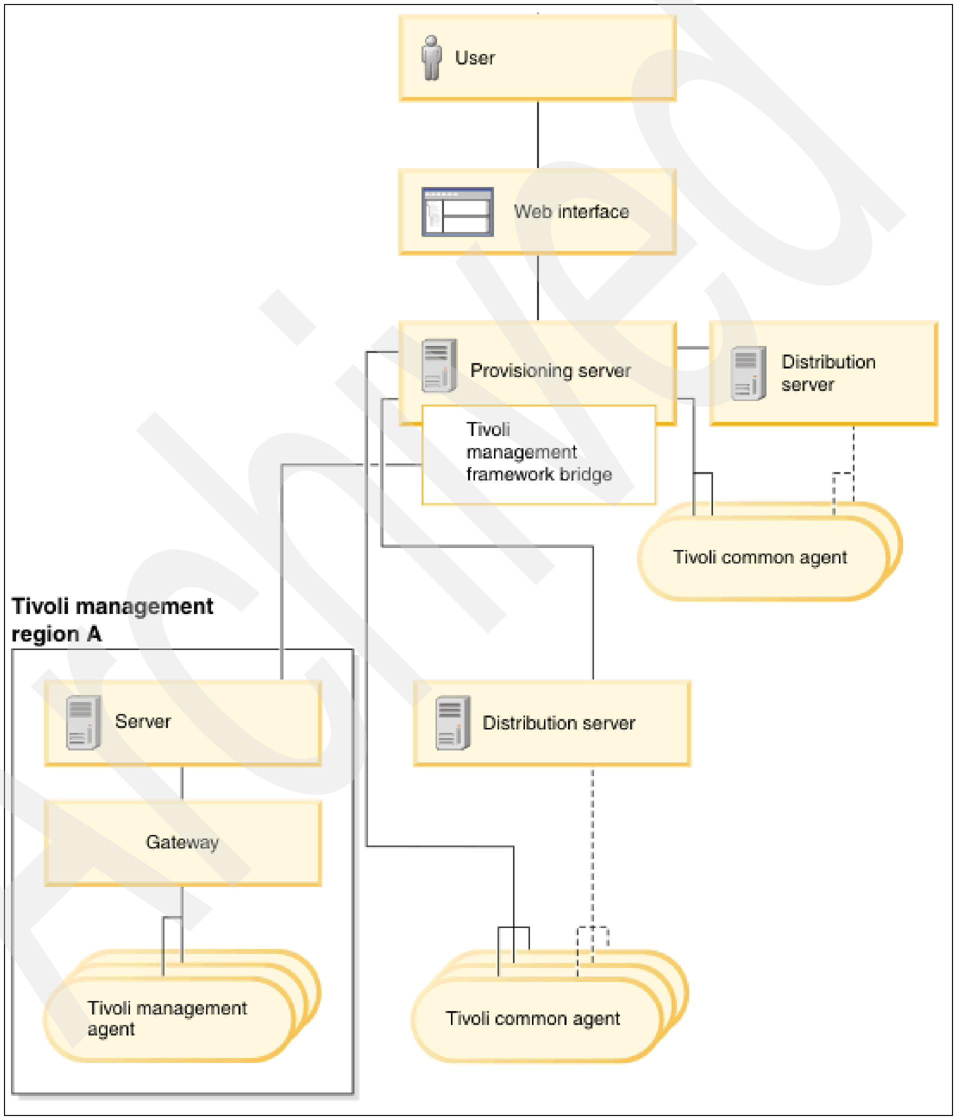


Figure 2-4 Moving to a Tivoli Provisioning Manager for Software environment

The computers that were in Region TMR B now have the common agent installed and a distribution server was deployed in their geographical location.

The deployment of the common agent is performed from the Tivoli Provisioning Manager for Software Web User Interface using an automation package that distributes the package to the Tivoli endpoints using the Tivoli Management Framework infrastructure. After the common agents are installed, they connect to the provisioning server, and the Tivoli management agents that are still installed can no longer be used for operations performed from the provisioning server.

Distribution servers provide a geographically distributed set of depots to which bulk data can be sent. They are deployed from the Web User Interface. In some ways, they are the equivalent of gateways and during the migration process, you can choose to deploy a distribution server for each gateway you deployed in the Tivoli Management Framework infrastructure.

2.2.3 Stage 3: Using Tivoli Provisioning Manager for Software to manage only common agents

Following the deployment of the common agent to the managed computers in a region, you manage the computers from the provisioning server through the scalable software distribution infrastructure to the common agent. The Tivoli Management Framework infrastructure and Tivoli management agents are no longer used by the Provisioning server.

The objectives of this stage are to:

- Completely replace Tivoli Configuration Manager infrastructure with depots and endpoints
- Deploy and configure depots and peering throughout the infrastructure
- Utilize all features of Tivoli Provisioning Manager for Software instead of Tivoli Configuration Manager

Customer benefits

The infrastructure can now exploit all of the capabilities of Tivoli Provisioning Manager for Software. Some of the benefits beyond stages one and two are noted in the following list:

- Depot system is more scalable with better availability, failover, and security characteristics
- Reduced hardware expenses from gateway architecture
- Reduced management costs on servers

You might decide to keep the Tivoli management agents on the computers where the common agent is installed, as shown in Figure 2-5.

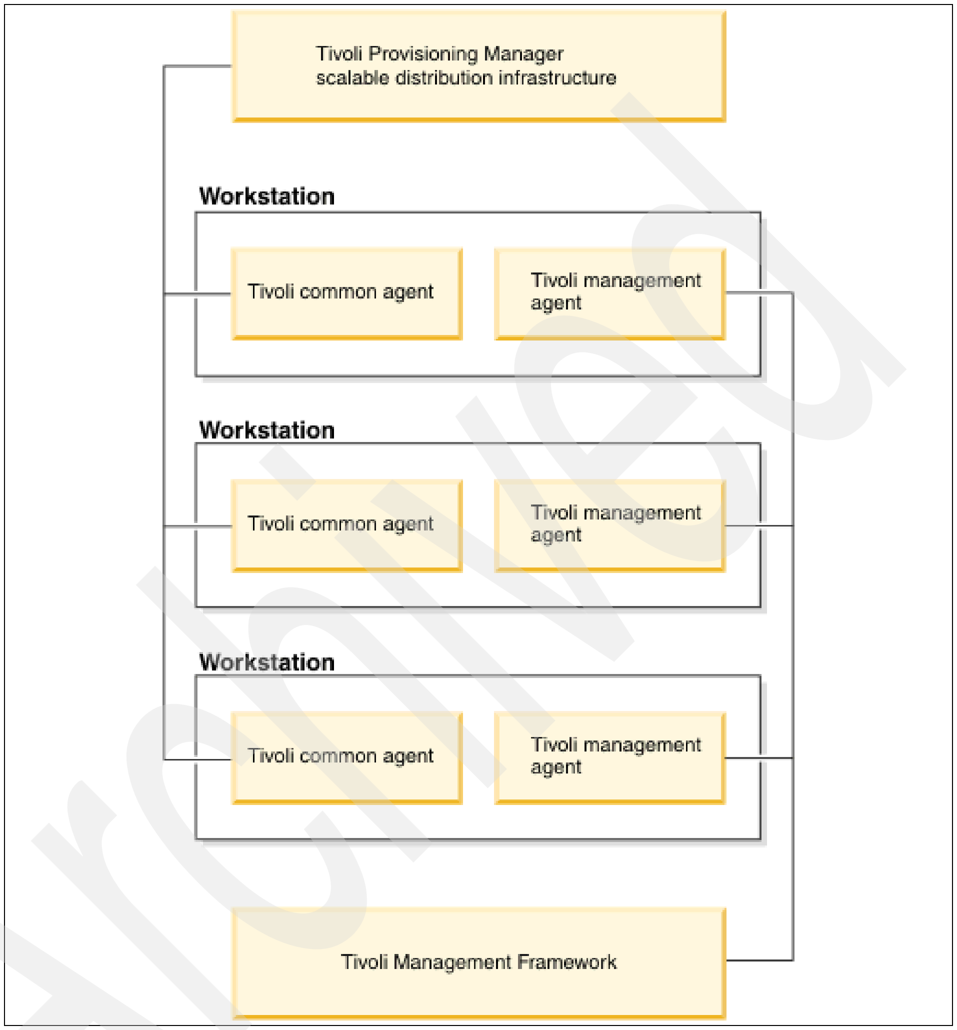


Figure 2-5 Computers running both the common agent and the Tivoli management agent

In this scenario, all software distribution, activity planning, and inventory tasks are performed from the Provisioning server, using the scalable software distribution infrastructure and the common agent.

The Tivoli Management Framework and the Tivoli management agent are retained to support other Tivoli Management Framework applications, for example Tivoli Remote Control.

Remote control: IBM intends to provide a remote control product that does not require the Tivoli Framework.

You could also use the Tivoli Management Framework for some Tivoli Configuration Manager functionality that is not supported by Tivoli Provisioning Manager for Software, for example Data Moving and support for software distribution to mobile endpoints.

3

Comparing Tivoli Management Framework to SDI

In this chapter, we compare the Tivoli Management Framework infrastructure to the new Scalable Distribution Infrastructure (SDI) of Tivoli Provisioning Manager. We start the chapter with a brief overview of the most important Tivoli Manager Framework components. Next, we describe the architecture of SDI and cover the most important components of the SDI in detail, including:

- Device management service (DMS)
- Common Agent Services (CAS)
- Dynamic content delivery service (CDS)

We also compare the functionality provided by Tivoli Management Framework's Multiplex Distribution 2 (MDist2) and the distribution service included with SDI. Finally the chapter provides a comparison between Tivoli Management Framework's Scalable Collection Services and the collection of inventory data using SDI.

This chapter has the following sections:

- “Tivoli Management Framework infrastructure” on page 42
- “Scalable Distribution Infrastructure” on page 47
- “Tivoli Provisioning Manager for Software Inventory collection” on page 76

3.1 Tivoli Management Framework infrastructure

In this section we briefly cover the most important components of the Tivoli Management Framework architecture. We also map these Tivoli Management Framework components to their corresponding SDI components. In this unit we assume you are familiar with the Tivoli Management Framework architecture.

3.1.1 Tivoli Management Framework components

A Tivoli Management Framework environment consists of a number of Tivoli Management Regions (TMRs), which can be optionally interconnected. The Tivoli Management Region consists of a three-tier architecture:

- A single TMR server
- A number of Tivoli gateways
- A number of Tivoli management agents (TMA)

Tivoli Management Server

The Tivoli Management Server controls and tracks all management tasks in a TMR. The TMR server maintains a distributed, object-oriented database that contains configuration information. Data, regarding definitions of Tivoli administrators and application specific profiles, such as software packages, are stored in this database.

Tivoli gateways

The Tivoli gateway provides communication between a set of TMAs and the TMR server. The gateway communicates with the TMR server on behalf of the TMA. The Tivoli gateway has the MDist2 repeater functionality built-in. This distribution mechanism uses a hierarchy of repeaters to efficiently send large amounts of data from a source host to a large number of TMAs. The source host is a server where all software package source data is stored.

Tivoli management agents

The TMA is the actual client in the Tivoli Management Framework. It is the target for all management operations. The TMA runs a program named the lightweight client framework daemon (lcfd). This program performs all communication with the Tivoli management server gateway.

3.1.2 Tivoli Management Framework to Tivoli Provisioning Manager for Software mapping

The equivalent of the TMR server is the Tivoli Provisioning Manager server. As we previously explained, TMR servers can be interconnected. The concept of interconnecting Tivoli Provisioning Manager servers is not available. In the next section we discuss how you can replace different TMR hierarchies with Tivoli Provisioning Manager server(s).

In a software distribution scenario the main function of a Tivoli gateway is to send the software distribution data to the target TMAs. In Tivoli Provisioning Manager, this functionality is provided by the Tivoli Provisioning Manager depot. The depot server stores and distributes files to common agents using the dynamic content delivery services, which we cover later in this chapter.

The equivalent of the TMA is the Tivoli common agent (TCA). The common agent is installed on target systems and is a common container for a number of subagents, each implementing a specific functionality.

The equivalent of the Source Host functionality in the Tivoli Management Framework environment is Tivoli Provisioning Manager's File Repository. A File Repository is a server that stores installable software images and other files that are installed on target systems.

3.1.3 MDist2 distribution service

MDist2 is a framework service used to efficiently transfer large amounts of data from a source host to a number of target machines (TMAs). The MDist 2 service includes the following functionality:

- Asynchronous delivery

  MDist2 distributions are submitted asynchronously. Intermediate status reporting is provided and distributions are set to expire at a configurable date.

- Distribution prioritization

  With MDist2 there are three distribution priorities: high, medium, and low.

- Distribution monitoring and control

  In addition to checking the status of the distribution, you can take action on it (pause, resume, or cancel) using the CLI and a GUI.

- Mobile computing support

  A graphical interface allowing users to download and install distributions at their convenience.

- Disconnected endpoint support

  Enables users to download and save distributions in a local storage directory for installation at a later date.

- Roaming endpoint support

  Enables endpoints to receive a distribution when an endpoint migrates to a different gateway during distribution.

- Installation from CD or file server

  Enables administrators to specify a CD or file server to use rather than a repeater.

- Wake on LAN® support

  Enables administrators to send distributions to powered off computers. If Wake On LAN is enabled, the machine wakes up to receive the distribution.

- Multicast support

  Enables system administrators to improve performance by using parallel distribution.

- Checkpoint restart

  A distribution can be restarted at the last successful checkpoint, which provides faster recovery from failed distributions, by not re-transmitting the entire distribution package.

MDist2 transfers its data through a repeater hierarchy using a store and forward mechanism. The distribution route starts from the software package source host, which sends the data through intermediate repeaters to the Tivoli gateway repeater, which sends the data to the target TMAs. The repeater hierarchy is configured statically using the Tivoli Framework command line interface. Table 3-1 shows a sample MDist2 repeater hierarchy.

Table 3-1 MDist2 repeater tuning parameters

MDist2 setting	**Description**
rpt_dir	The depot directory, which contains all the segments stored in the database and must have enough free space to hold the value of disk_max.

MDist2 setting	Description
permanent _storage	Configures the repeater to be a depot. If set to TRUE, the depot retains segments marked for permanent storage after the distribution finishes.
max_sessions_high	Specifies the maximum number of concurrent connections a repeater opens for high-priority distributions. These connections are shared among all active distributions. If the repeater runs out of high-priority connections, it tries to use a medium-priority connection.
max_sessions_medium	Specifies the maximum number of concurrent connections a repeater opens for medium-priority distributions. These connections are shared among all active distributions. If the repeater runs out of medium-priority connections, it tries to use a low-priority connection.
max_sessions_low	Specifies the maximum number of concurrent connections a repeater opens for low-priority distributions. This number must be one or greater. These connections are shared among all active distributions. If the repeater runs out of low-priority connections, it waits for an open connection to complete before opening additional connections.
disk_max	Specifies the amount of disk space allocated to the repeater depot. Units are in megabytes (MB).
mem_max	Specifies the amount of memory (in MB) that are used to buffer data being sent to targets.
notify_interval	Specifies the frequency (in minutes) of status reporting. When the notify_interval elapses or the distribution completes on all targets, the results are flushed. The results are sent to the application using MDist 2 and updated in the MDist 2 database.
conn_retry_interval	Specifies the frequency (in seconds) that unavailable or interrupted targets are retried.
retry_ep_cutoff	Specifies the amount of time (in seconds) to continue retrying an unavailable or interrupted endpoint. Unavailable or interrupted repeaters are retried until the distribution deadline is reached.
net_load	Specifies the maximum amount of network bandwidth (in kilobytes per second) that the repeater is allowed to use.
target_netload	Specifies the maximum amount of network bandwidth (in kilobytes per second) that can be sent to an individual target.

MDist2 setting	Description
repeater_multicast	When an application sends a multicast distribution, indicates whether the gateway uses multicast when distributing packages to other repeaters.
endpoint_multicast	When an application sends a multicast distribution, indicates whether the gateway uses multicast when distributing packages to its endpoints.

3.1.4 Scalable Collection Services

Besides the MDist2 distribution service, the Tivoli Management Framework infrastructure also includes a Scalable Collection Service (SCS). SCS enables efficient, asynchronous collection of large amounts of data across complex networks. SCS significantly reduces the time needed to scan and store vast amounts of data. SCS provides the ability to scale data collection. SCS sends scan data to its destination as the scan on each target completes, so it sends smaller chunks of data through the network.

Another advantage of SCS is its ability to stage data through a hierarchy of collector nodes (repeaters), which distribute the processing load across the Tivoli management region (Tivoli region). In essence, SCS more effectively manages the stream of data while creating multiple pathways to the configuration repository. This functionality results in faster scans and increased control of network resources.

SCS is triggered by a target that has inventory data to be collected. This target informs the gateway that the data is ready by sending the CTOC (Collection Table Of Contents). A collector daemon, running on the gateway, then retrieves the data from the target and stores it on the collector in persistent storage. Depending on the configuration of your network, the data may then travel through a hierarchy of collectors before arriving at its destination. With Inventory, all collectors send data to the inventory data handler, which then sends the data to the configuration repository (RDBMS).

The following SCS features illustrate the advantages of asynchronous data collection:

- Asynchronous data collection and asynchronous processing lead to better distribution of the processing load across more nodes in the Tivoli region, as well as better use of the network. SCS stages the data at different collectors in the network as it flows to the configuration repository, and it allows control over the data flow between successive collectors in the hierarchy.

- After data is collected from a target, that target can be disconnected without affecting SCS.
- The inventory data handler can write data in parallel to one or more RIM objects, each of which has one or more connections to the RDBMS.
- Using SCS and the Tivoli scheduler, you can schedule scans and data collection traffic for times when network traffic is at a minimum.
- Collectors in the collector hierarchy store collected data in depots. If a failure occurs, the collector can resend the data when it is back online, so you do not have to scan all the targets again.
- SCS provides functionality through which you can get information about the status of collectors, CTOCs, and scans.
- With SCS, you can determine the status of individual targets as they complete, rather than having to wait until all targets complete.
- SCS allows you to retry collecting data from targets or other collectors following a failure. You can also set the specific number of retries.

3.2 Scalable Distribution Infrastructure

In this section, we discuss the details of SDI components.

3.2.1 SDI components

The SDI components implement a scalable infrastructure and manage large distributed environments. The SDI infrastructure manages computers with the common agent deployed. SDI consists of three main components:

- Common Agent Services (CAS)
- Device management service (DMS)
- Dynamic content delivery service (CDS)

In the next sections we cover each of these components in detail.

3.2.2 Common Agent Services

The Common Agent Services (CAS) architecture provides a shared infrastructure for managing computer systems in a distributed IT environment.

The CAS infrastructure consists of three main components:

- Common agent
- Agent manager
- Resource manager

Agent manager

The agent manager is the server component of the Tivoli Common Agent Services, which provides functions that allow clients to get information about agents and resource managers.

The agent manager enables secure connections between managed endpoints, maintains the database information about the endpoints and the software running on those endpoints, and processes queries against that database from resource managers. Additionally, the agent manager includes a registration service, which handles security certificates, registration, tracking of common agents and resource managers, and status collection and forwarding.

Resource manager

Each product that uses Tivoli Common Agent Services has its own resource manager and subagents. For example, Tivoli Provisioning Manager has a resource manager and subagents for software distribution and inventory scanning.

The resource manager interacts with the agent manager:

- Registers with the agent manager
- Queries common agents to identify which ones are present in the environment
- Requests an initial security certificate
- Delivers subscription-based notification of common agent registration and configuration

The resource manager interacts with the common agent:

- Deploys bundles for its subagents
- Starts and stops the subagents
- Queries the configuration of the common agent and installed bundles

The resource manager interacts with the bundles and subagents installed in the common agent:

- Resource manager runs commands on the common agent
- Subagent sends information to the resource manager

Tivoli Common Agent

The Tivoli Common Agent consists of common agent services code and product-specific subagent code. For example, Tivoli Provisioning Manager includes subagents for deploying software and obtaining inventory from managed endpoints. The product-specific subagents consist of one or more OSGi bundles. A bundle is an application that is packaged in a format defined by the Open Services Gateway Initiative (OSGi) Service Platform specification, which is implemented in a light weight runtime based on WebSphere Everywhere Deployment technology.

The common agent services code is installed once on a managed endpoint. For example, if you have two management applications on the same endpoint (application A and application B), the common agent code is installed only once on the endpoint. However, there are two product-specific subagents: one for application A and one for application B.

The common agent provides the following functionality:

- Continuous operation. Self-healing features ensure that the common agent and subagents are always available. If the common agent stops, a "watchdog" process, called the nonstop service, automatically restarts it.
- A single set of security credentials and a common security infrastructure for all management applications.
- Automated management of security credentials. When common agent certificates near their expiration date, they are automatically renewed.
- Deployment and life cycle management of subagents. Resource managers can remotely install, upgrade, patch, or uninstall bundles on any common agent. This helps keep the common agent deployment current without having to take explicit action on each common agent system.
- Common agent health monitoring and configuration monitoring. The common agent has a "heartbeat" function that sends periodic status and configuration reports to the agent manager. The common agent allows any subagent to participate and to provide status information. Management applications can register to receive these updates. Updates are initiated by certain bundle events and periodically by the common agent. You can turn off periodic updates or control the frequency of updates. The default frequency is 24 hours.

The common agent contacts the agent manager and reports its status and any configuration changes at the following times:

- When a common agent starts or stops.
- After a configurable period of time. The default is 24 hours.
- Any time a bundle is installed, upgraded, or removed.

Figure 3-1 shows the three components of the CAS architecture and their interaction.

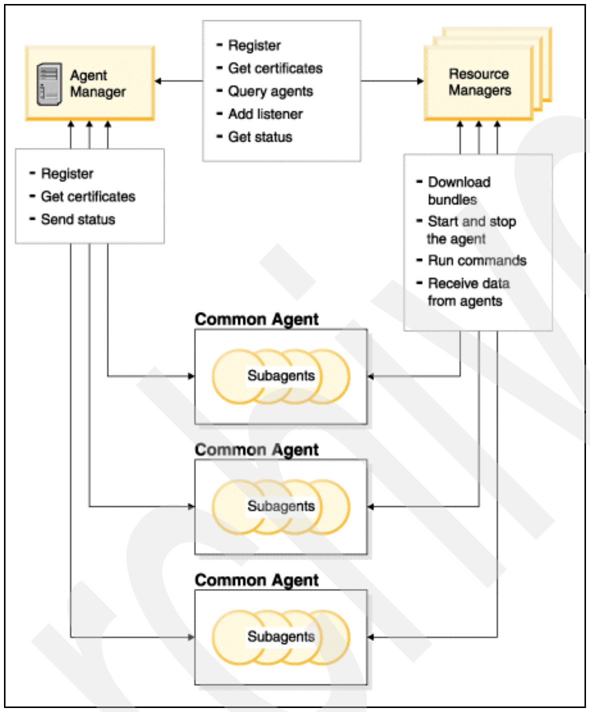


Figure 3-1 Common Agent Services architecture

3.2.3 Device management service

The device management service provides a flexible solution for online and offline device management. The device management service mainly performs *job management operations*. Within software management, the device management service initiates software downloads, runs installation actions, and collects results. It can also be used for device configuration, inventory scanning, and data collection.

Device management service tracks the progress of jobs and maintains a history of past jobs. Management actions performed using the device management service are called jobs. Jobs can be applied to or targeted to individual devices or device groups. You can also monitor the job progress on various devices. If applied to a specified device, a job is run when the device connects to the device manager server.

Currently, a two-tiered federated environment is supported, including the following elements:

- *Device manager federator*, which is installed on the provisioning server at the enterprise, is configured to act as a federated server. The federator implements a job distribution policy that delivers incoming jobs to all of the regional federated agents that are configured to communicate with it.
- Device manager *federated agents* are installed as device manager servers at the branch and are configured to communicate with the federator periodically to get jobs and to distribute them to the set of currently known clients installed on the endpoints in the branch. The agents determine whether there is a federator job available to be distributed to the clients and work in conjunction with a federator plug-in to run the job.

Clients are installed as device manager subagents (Job Execution Service or JES subagent) on the managed systems or endpoints at the branch. They are used for receiving command jobs from and returning results to the federated agents.

Job processing

The device manager federator provides the "federation" capability for submitting jobs and collecting results from the intermittently connected device manager federated agents. The job instructions are typically a series of encapsulated Job Execution Service (JES) commands in XML format.

The device manager federator and federated agents do not parse the job instructions or results. Their role is to just pass the job instructions through the servers to the device manager subagents and collect the results:

- The device manager federated agents periodically contact the device manager federator to see if there are any jobs waiting in the job queue. If jobs are waiting, they are automatically propagated to the federated agents.
- The device manager subagents periodically connect to the federated agents to see if there are any jobs waiting in the job queue, obtain the jobs, process the jobs, and send the job results back to the federated agents.
- From the federated agents, the job results are periodically returned to the federator, where they are finally retrieved by the Tivoli Provisioning Manager server.

Important: The difference between the concept of jobs in Tivoli Management Framework and Tivoli Provisioning Manager is substantial. If Tivoli Management Framework jobs are sent to the target machine and if the machine is not online, the job fails, or for software distribution jobs, the gateway periodically retries the job until it expires. In Tivoli Provisioning Manager a job is submitted to the management server and only starts when the common agent checks in with a federated agent. This behavior is designed to increase scalability and robustness. The polling processing is delegated out to the agents, which reduces the load on the federated agents.

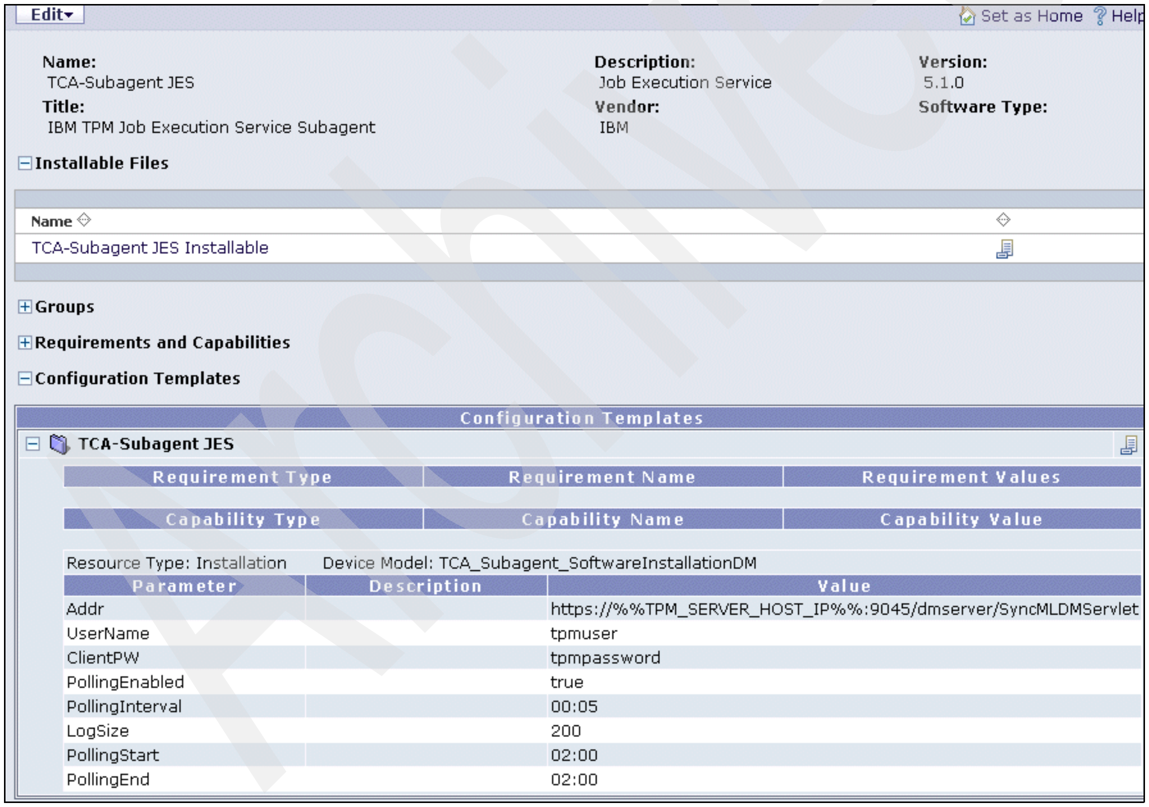
The polling interval of the JES subagent is set on the managed system by changing the value of PollingInterval in the file CA_HOME/jes.properties, as shown in Example 3-1. If you edit this or any other properties file, all special characters such as the colon in the time variables are quoted with a preceding “\”. You must restart the agent for the changes to take effect. You can change other polling variables to restrict the time of day that the agents poll.

Example 3-1 jes.properties file

```
#Tue Aug 29 13:01:36 BST 2006
PollingEnabled=true
PollingStart=02\:00
PollingEnd=02\:00
PollingInterval=01\:00
ClientPW=tpmpassword
Addr=https\://192.168.0.1\:9045/dmserver/SyncMLDMServlet
UserName=tpmuser
LogSize=200
```

Instead of editing the jes.properties file on the target system, you can use a workflow named *JES_Parameter_Set* to change the polling parameters of the JES subagent. The workflow takes three parameters: device ID, parameter name and parameter value. The only valid parameters are PollingEnabled, PollingEnd, PollingStart, and PollingInterval.

You can set the parameters centrally by modifying the Tivoli Common Agent subagent JES *before* the common agent is delivered and installed. You can do this from the Tivoli Provisioning Manager GUI by selecting **Software Management** → **Manage Software Catalog** → **Software Stacks** → **TCA_Stack**. This action shows a list of all modules in the TCA_Stack software stack. Selecting the "TCA-subagent JES" opens a window where you can change the software settings of the common agent JES subagent software product. Change the parameters by selecting the configuration template, as shown in Figure 3-2.



Name: TCA-Subagent JES
Title: IBM TPM Job Execution Service Subagent
Description: Job Execution Service
Vendor: IBM
Version: 5.1.0
Software Type:

Installable Files

Name
TCA-Subagent JES Installable

Groups

Requirements and Capabilities

Configuration Templates

Configuration Templates

TCA-Subagent JES

Requirement Type	Requirement Name	Requirement Values

Capability Type	Capability Name	Capability Value

Resource Type: Installation   Device Model: TCA_Subagent_SoftwareInstallationDM

Parameter	Description	Value
Addr		https://%%TPM_SERVER_HOST_IP%%:9045/dmserver/SyncMLDMServlet
UserName		tpmuser
ClientPW		tpmpassword
PollingEnabled		true
PollingInterval		00:05
LogSize		200
PollingStart		02:00
PollingEnd		02:00

Figure 3-2 JES subagent software configuration template

In our example in Figure 3-2 on page 53, the polling interval was changed to five minutes. In this way, new installations of common agents will use a polling interval of five minutes instead of the default one hour polling interval.

When the JES subagent executes the job, it returns the result to the federated agent. The federated agent periodically contacts the federator to return the result. This polling interval is controlled by the *instFederatedAgentPollFrequencyInMinutes* parameter in the DMSconfig.properties file, which is located in the directory DeviceManager\Config on the federated agent.

3.2.4 Dynamic content delivery service

The dynamic content delivery service (CDS) is a distributed, grid-like service that efficiently distributes large files (including software packages) around the network. The dynamic content delivery service allows you to upload of files from a *file repository* to a *depot server*. Files can be replicated to other depot servers located close to computers that need the files. The dynamic content delivery service client can download a file from multiple depot servers or peers, and it can simultaneously retrieve different segments of the file from different systems. The file segments are then reassembled on the client's local system.

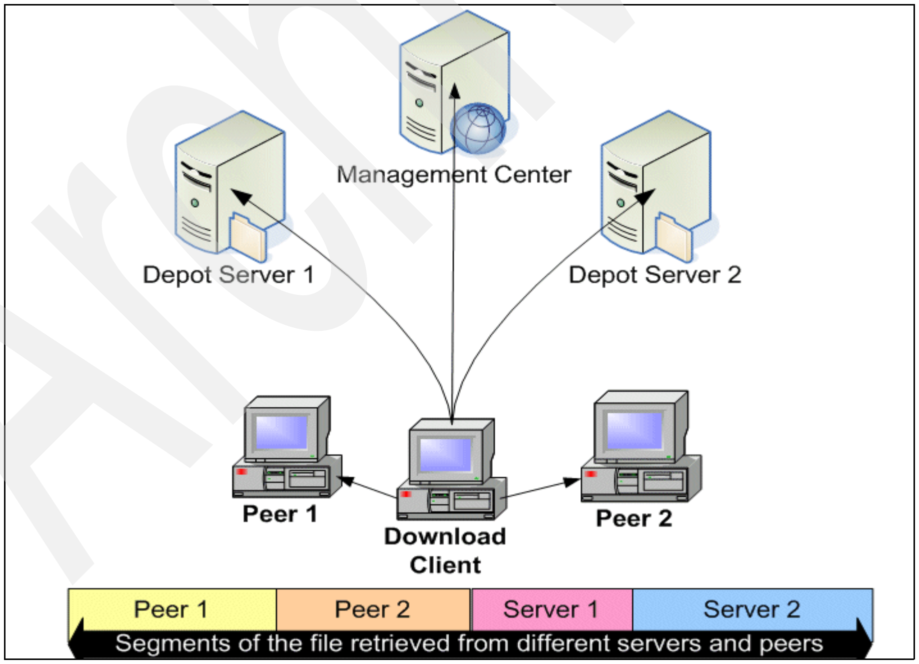


Figure 3-3 Dynamic content delivery service file download mechanism

With this design, multiple systems can quickly download files with minimal impact to the network infrastructure.

Dynamic content delivery service includes the following features:

- Scalable design: You can add depot servers to accommodate the needs of your network.
- Peer-to-peer file sharing: Clients can download a file from peer systems that have previously downloaded the same file (covered later in this chapter).
- Adaptive bandwidth control: You can throttle the transfer rate from both depot servers and peers to minimize the impact of downloads on existing network traffic from other applications.
- File encryption.
- Reports about download activity.

Dynamic content delivery service architecture

Dynamic content delivery service is made up of the following components:

- Management center: The management center provides centralized control of uploading, replicating, and downloading files.
- File repository: A file repository is a server used to store files.
- Depot servers: A depot server is a component that stores files and provides files to clients for download.
- Regions: Regions are used to group depot servers that are located near one another to optimize upload, replication, and download times.
- Zones: A zone is an IP range or domain and is used to logically group systems within regions.
- Clients: A client handles the communication and security between the system uploading or downloading a file and all of the other components. When peering is enabled, clients share downloaded files with other clients. The dynamic content delivery service client is implemented as a subagent of the Tivoli file repository.

Management center

The dynamic content delivery service management center is the central managing point of the dynamic content delivery service infrastructure. It provides the following functionality:

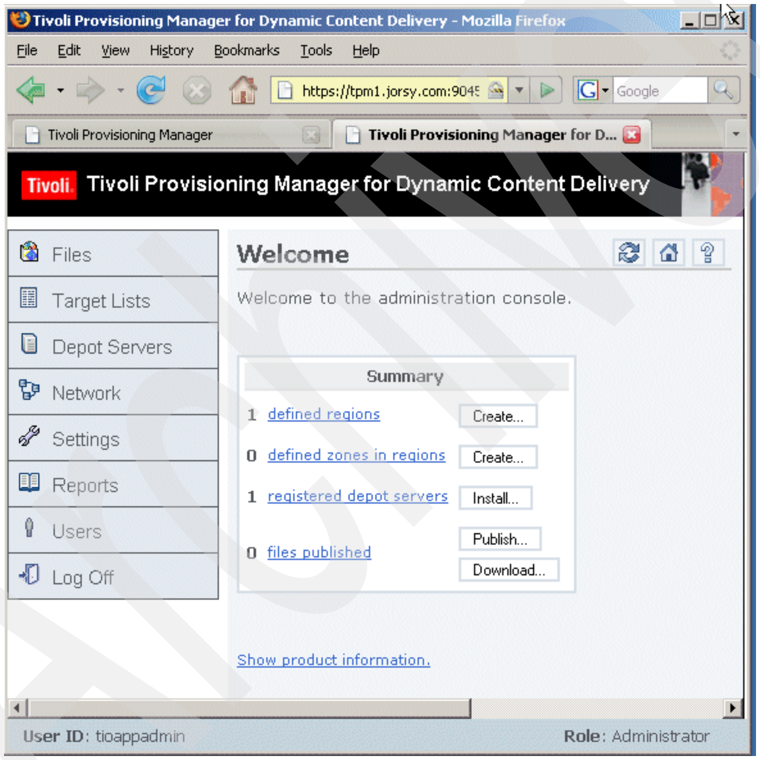
- Replicate published files to the depots' hierarchy.
- Generate and authenticate download plans when a client requests a file.

- Monitor the state of depot servers and automatically restart the depot server if it is stopped.
- Store file data and download statistics.
- Store configuration parameters for each depot, zone, and region.
- Maintain a catalog of files stored on depots and peers.

You can directly access the management center from the following Web site:

`https://tpm_server:9045/admin`

After you authenticate using tioappadmin, the Web interface in Figure 3-4 is displayed.


Tivoli Provisioning Manager for Dynamic Content Delivery - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://tpm1.jorsy.com:9045
Tivoli Provisioning Manager
Tivoli Provisioning Manager for D...
Tivoli Provisioning Manager for Dynamic Content Delivery
Files
Target Lists
Depot Servers
Network
Settings
Reports
Users
Log Off
Welcome
Create...
Create...
Install...
Publish...
Download...
Show product information.

Welcome to the administration console.

Summary	
1 defined regions	Create...
0 defined zones in regions	Create...
1 registered depot servers	Install...
0 files published	Publish... Download...

User ID: tioappadmin Role: Administrator

Figure 3-4 The dynamic content delivery service management center

Tivoli Provisioning Manager provides a front-end to the management center, the Tivoli Provisioning Manager Web User Interface, which eliminates the need to access the management center directly.

File repositories

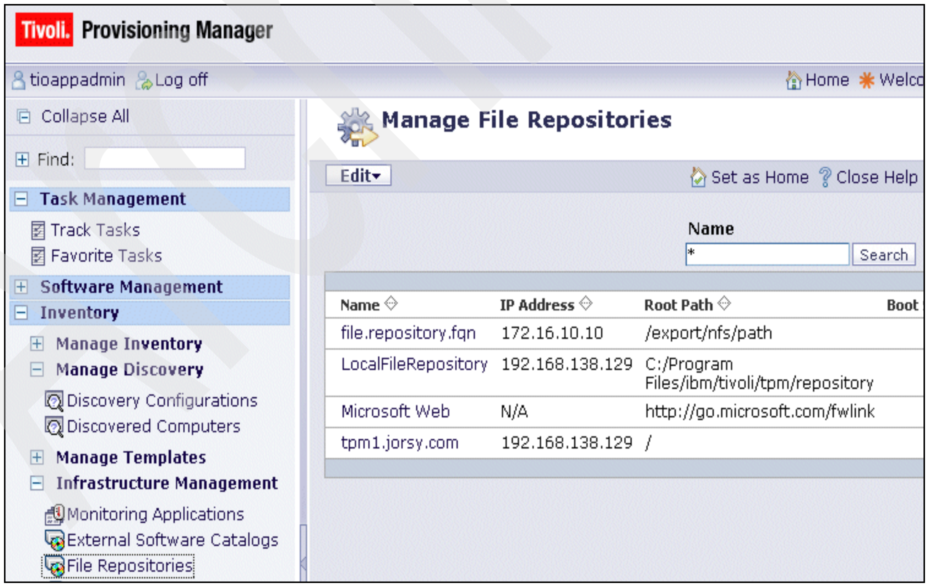
A file repository is a server that you use to centrally store software, images, scripts, application data, and other files that you want to install or run on target managed systems. A file repository called *LocalFileRepository* is created at installation time on the Tivoli Provisioning Manager server with the directory path set to *TIO_HOME*/repository.

You can use file repositories to:

- Add new software to the software catalog. You can select a file repository where you currently store the installation packages. For example, The Tivoli Software Package Editor can be used to save SoftwarePackageBlocks to a File Repository.
- Store a captured image.
- Create external software catalogs using specialized file repositories that provide software updates, such as Microsoft Windows Server® Update Services.

When a software distribution is initiated, the software package file is first uploaded from the file repository to the depot server.

File repositories are managed using the Tivoli Provisioning Manager GUI: **Inventory** → **Infrastructure Management** → **File Repositories**.



Name	IP Address	Root Path	Boot
file.repository.fqn	172.16.10.10	/export/nfs/path	
LocalFileRepository	192.168.138.129	C:/Program Files/ibm/tivoli/tpm/repository	
Microsoft Web	N/A	http://go.microsoft.com/fwlink	
tpm1.jorsy.com	192.168.138.129	/	

Figure 3-5 Managing file repositories

Depot servers

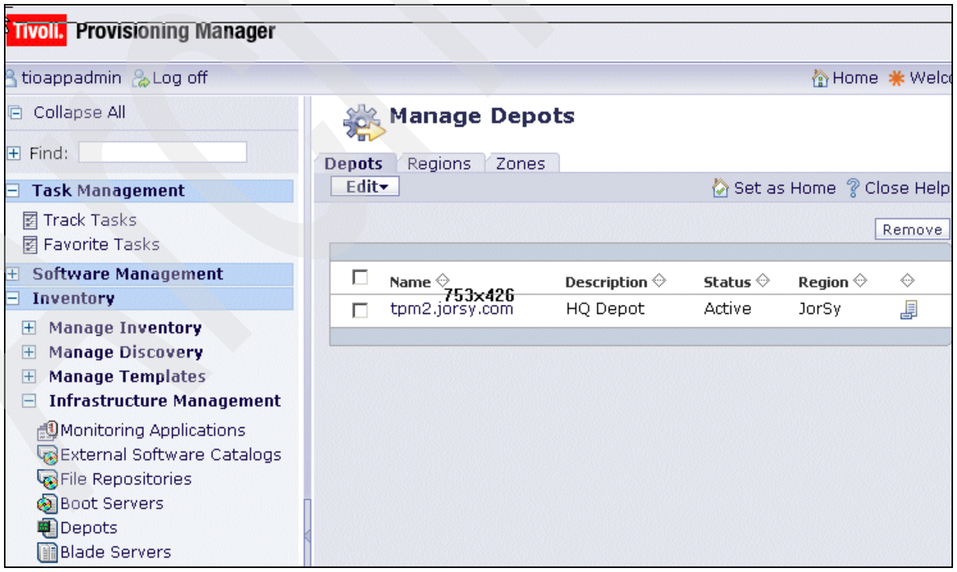
A depot server is a system that stores files for distribution. Distribution files are uploaded to a depot server using a client. Uploaded files are stored in a directory that you specify when the depot server is installed. Depot servers can replicate uploaded files to other depot servers to optimize the efficiency of network traffic. The target computers download the files from the depot servers.

Dynamic content delivery service uses intelligent routing to point a client to its nearest depots and peers. This is done by mapping clients to a position on the network defined through IP addresses and domain names.

Before you can define a depot server, you must create a *region*. Regions are used to *group depot servers* that are located near one another to optimize upload, replicate, and download times. When you create a depot server, you must associate it to a region. You can assign a depot server to *only one* region.

A *zone* can be used to define an IP range or a domain within a region. Regions can be generic, without a real network definition, or they can be specific, when zones are used to define IP domains within the region. Zones are logically group systems within regions.

Regions, zones, and depots are all managed using the Tivoli Provisioning Manager GUI: **Inventory** → **Infrastructure Management** → **Depots**. See Figure 3-6.



Name	Description	Status	Region
tpm2.jorsy.com	HQ Depot	Active	JorSy

753x426

Figure 3-6 Managing depots, regions, and zones

In the Tivoli Management Framework infrastructure, TMAs always receive their data from their assigned Tivoli gateways. Gateway assignment is accomplished using endpoint policy, which is a set of scripts that are executed upon TMA login.

Imagine the following scenario:

- A company with one datacenter and two geographical locations (East and West). Each location hosts plus-or-minus 500 workstations and 500 portables. The portables can roam between the two locations.
- Separate DNS domains for each location: east.jorsy.com and west.jorsy.com.
- A server for software distribution available at each site.

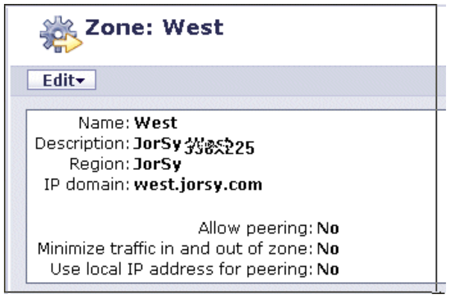
In the Tivoli Management Framework infrastructure, we would install two gateways at each location. In order to assign the endpoints to a correct gateway, we have to develop endpoint policy scripts. The endpoint login policy script has to check on the fully qualified host name of the PC and assign it to the correct location.

When a portable PC in the East region (connected to the East gateway) is switched off during a distribution and later connected in the West region the following occurs:

- The TMA starts in the West region and sends its login to the East gateway.
- The login policy script detects that the PC is connected to the “wrong” gateway and migrates the TMA to the West gateway.
- The TMA logs into the West gateway.
- The distribution roams from the East gateway to the West gateway and resumes from the previous checkpoint.

In a Tivoli Provisioning Manager scenario, we create two depots, two regions, and two zones. The West region links the West depot with the West zone, and similarly the East region links the East depot with the East zone. The zones are defined using the corresponding IP domain.

Figure 3-7 shows the setup of the West zone.



Name: **West**
Description: **JorSy West 358x225**
Region: **JorSy**
IP domain: **west.jorsy.com**

Allow peering: **No**
Minimize traffic in and out of zone: **No**
Use local IP address for peering: **No**

Figure 3-7 Definition of the West zone

Note: We cover the "peering" settings of the zone later on in this chapter.

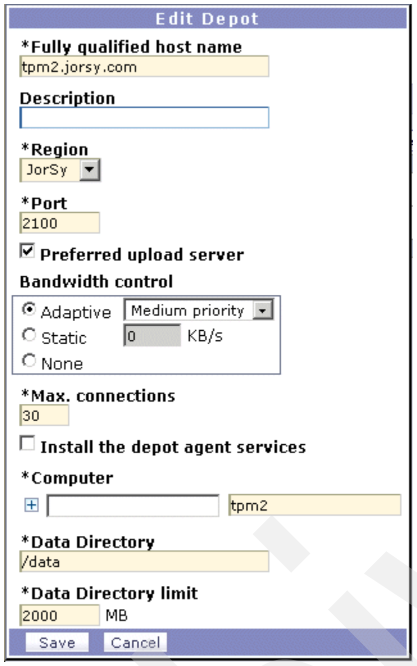
When a portable PC in the East region is switched off during a download from the East depot and it is later switched on in the West region, the following occurs:

- At startup, the common agent re-contacts the Management Center on the Tivoli Provisioning Manager server to request a new download plan.
- The Management Center calculates a new download plan using the common agent's IP domain.
- The common agent uses the new download plan, and starts downloading the missing segments from the West depot.

An important advantage of the Tivoli Provisioning Manager for Software infrastructure in this scenario is that no extra development is required to support roaming clients. The environment only requires the correct configuration of depots, regions, and zones.

Depot server settings

Each depot server has a number of tuning parameters that can be changed using the Tivoli Provisioning Manager User Interface. You can access the depot server settings by selecting **Inventory** → **Infrastructure Management** → **Depots**, and choosing **Properties** on the icon next to the depot server. See Figure 3-8 on page 61.


Edit Depot

*Fully qualified host name
tpm2.jorsy.com

Description

*Region
JorSy

*Port
2100

[✓] Preferred upload server

Bandwidth control
[● Adaptive] Medium priority
○ Static 0 KB/s
○ None

*Max. connections
30

☐ Install the depot agent services

*Computer
tpm2

*Data Directory
/data

*Data Directory limit
2000 MB

Figure 3-8 Depot server settings

If a depot servers is designated as a *preferred upload server*, files are sent to this depot server before any other depot server in this region. If no preferred upload server is available in a region, then a depot server is selected to receive the uploaded file. In this concept the term upload describes a software product installable file that is retrieved from a file repository and placed on a depot server.

You can configure the bandwidth that a depot server uses in three different ways:

- None

  The depot server does not implement any bandwidth control.

- Static

  (KB/s) The depot server does not use more bandwidth than the configured threshold. This setting is comparable with the MDist2 net_load setting.

- Adaptive

  As software distribution data is seldom the most important network data, dynamic content delivery service provides adaptive bandwidth control, which is a new technology that minimizes the impact of software deliveries on other network activities. Adaptive bandwidth control monitors the response time of packets sent to a target and slows down the distribution if the response time is slow because this indicates that the network is busy. Figure 3-9 shows how the adaptive bandwidth control is implemented.

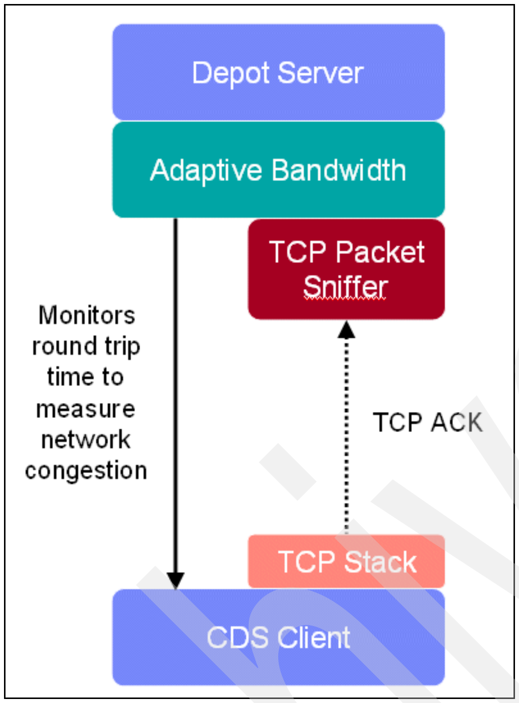


Figure 3-9 Dynamic content delivery service adaptive bandwidth control

Dynamic content delivery service's adaptive bandwidth control monitors all of the network traffic between the sender and the receiver. Some implementations only watch the local adapter of the machine to prevent interfering with the user of that machine. These implementations fail to ensure that multiple systems do not overload or flood the network. Note that this functionality is not available in Tivoli Management Framework's MDist2 distribution service.

The *max_connections* setting of a depot server configures the maximum number of simultaneous connections a depot server will accept.

> **Note:** For an MDist2 repeater, it is possible to define a maximum number of connections for each priority queue (low, medium, high). In dynamic content delivery service, the concept of priority queues is not available yet.

The *data directory* and *data directory limit* determine where the depot server stores the downloaded files and the maximum size in megabytes for this directory.

3.2.5 Tivoli Provisioning Manager Software Distribution

In this section we provide the steps for launching a software distribution to a target machine. In a scenario where a SoftwarePackageBlock is sent to a common agent using SDI, the following steps occur:

1. The Operator launches the software installation using the Tivoli Provisioning Manager User Interface by selecting the software product (SPB) and the target machines or common agents.
2. Tivoli Provisioning Manager automatically publishes the SoftwarePackageBlock to the depot systems:
   - Tivoli Provisioning Manager requires a publish to the dynamic content delivery service management center. The dynamic content delivery service management center just returns an upload depot. Tivoli Provisioning Manager *pre-loads the file to the upload depot and at this point Tivoli Provisioning Manager is free to submit the request to device management service* (create the device management service job).
   - In the meantime, dynamic content delivery service takes care of replicating the published file to every depot in a dynamic list that was built by dynamic content delivery service itself, considering which is the set of target computers that will need to download that file.
   - In this case the Management Center uses the Dynamic Depot Selection algorithm: given a list of targets (IP address and domains), dynamic content delivery service generates a list of depot servers that can be used to pre-cache the distribution based on their nearness to the targets and the number of targets.
3. A job is submitted to device management service:
   - The content of the job is an XML document, which gets processed by the JES subagent of the common agent.
4. The job is distributed by the central device management service task manager to any relevant device management service federated agent.
5. The task management system now waits for the common agents to connect.
6. The common agent's JES subagent polls the device management service federated agents and receives the details of the new job. As explained earlier, the polling interval is configurable per common agent (default 1 hour).

7. The SDI subagent contacts the management center and requests a download plan.
8. The management center generates a download plan that includes a list of depot servers and peers (we explain peering in the next section) from which the client can download the file. The management center selects these depot servers and peers that are pre-loaded and based on their location and past performance history.
9. The client starts the download:
   - The client opens connections to multiple depots and peers to retrieve the file.
   - After the download completes, the client notifies the management center that the download was successful and sends the file transfer rates achieved during downloads. The management center stores this data for use in creating future download plans.
10. The common agent SPBHandler subagent installs the SPB on the target machine.
11. The common agent notifies the device management service federated agent of the job completion.
12. The device management service federated agent periodically returns the results to the device management service federator.
13. The operator monitors the updated status using the Tivoli Provisioning Manager User Interface (track tasks).

3.2.6 Peering

Another important new feature of the SDI infrastructure, not available in Tivoli Management Framework's MDist2 service, is peering. When peering is enabled, common agents can act as miniature depot servers, meaning that they can hold copies of distributed files in a cache and act as sources for this file during downloads by their neighbors.

The cache on peers automatically retains files downloaded by that machine. When a preconfigurable amount of disk space is used, the peer deletes files based on their last access time. This allows peers to keep popular content, while automatically deleting seldom used files. The dynamic content delivery service management center maintains the list of files that a peer holds.

The transfer rate from peers is dynamically throttled to avoid interfering with other network traffic. It uses the same mechanism as the adaptive bandwidth control of depot servers.

When peering is enabled, the management center can assign multiple depot servers and peers to each download client for each download. The management center uses the client's IP address and TCP domain to find the best servers and peers. The peering functionality also provides an easy mechanism for managing branch offices—clients download from only peers in their branch office, which could be a huge saving on infrastructure components. The first client downloads the software from a depot outside the branch office, and then it acts as a peer for other clients in the branch office. Figure 3-10 shows a typical infrastructure that uses peering.

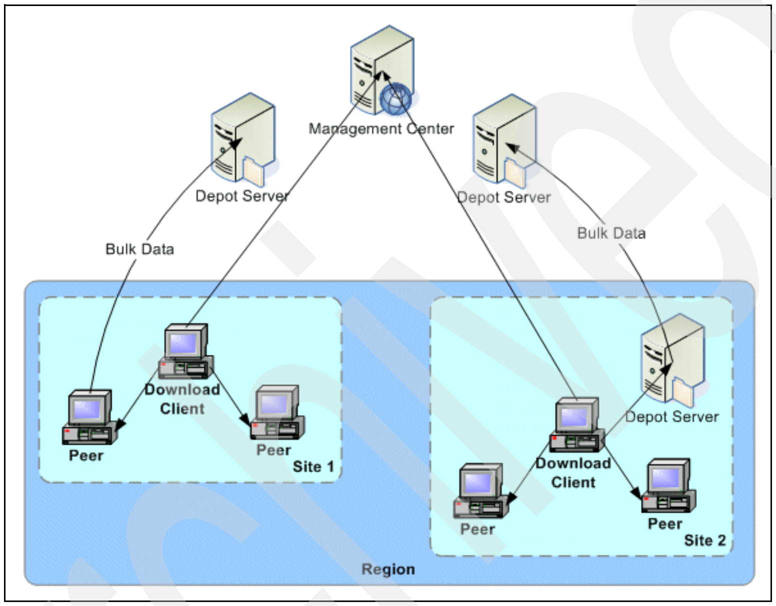
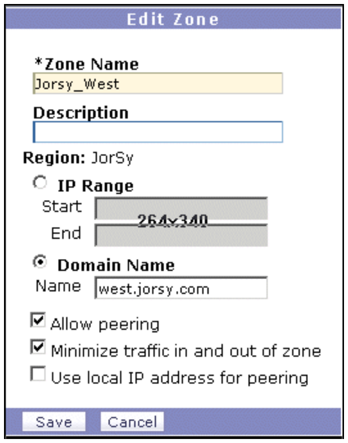


Figure 3-10 Dynamic content delivery service using depots and peering

Enabling peering

For peer-to-peer file sharing to become active, you must enable it in two places. First, set the zone, which contains the managed system, to allow peering. To set the zone, change the settings of the zone as shown in Figure 3-11 on page 66.



*Zone Name: Jorsy_West

Description:

Region: JorSy

( ) IP Range
Start
End

(●) Domain Name
Name: west.jorsy.com

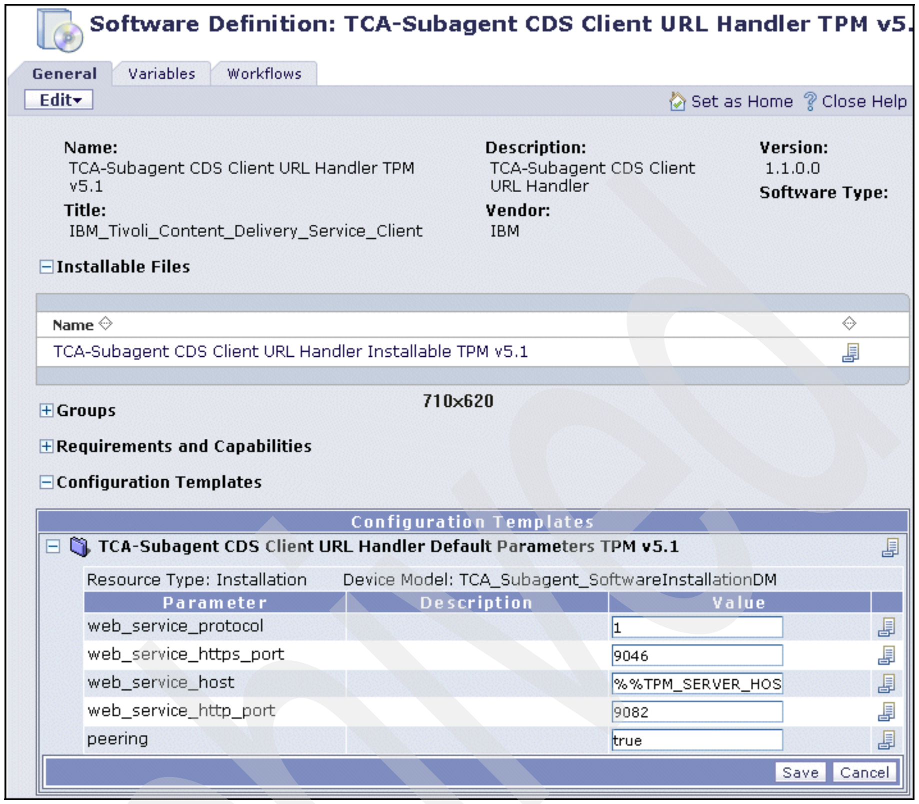
☑ Allow peering
☑ Minimize traffic in and out of zone
☐ Use local IP address for peering

Figure 3-11 Zone - Enabling peering

Select the Allow peering option to enable peering. The Minimize traffic in and out of zone setting prevents systems in other zones from using this zone for uploads and downloads, unless no other systems are available.

For Network Address Translation (NAT) environments, the management center stores both the local and remote TCP/IP address. For NAT environments, you must enable the Use local IP address for peering option.

The second step in configuring peering is to enable the peering option on the dynamic content delivery service subagent that provides dynamic content delivery services on the common agent. You can set this before the common agent is installed by editing the configuration template of the software package “TCA-Subagent CDS Client URL Handler TPM 5.1” and setting the peering parameter to true as shown in Figure 3-12 on page 67.



Software Definition: TCA-Subagent CDS Client URL Handler TPM v5.

General | Variables | Workflows

Edit▾ | Set as Home | Close Help

Name: TCA-Subagent CDS Client URL Handler TPM v5.1
Description: TCA-Subagent CDS Client URL Handler
Version: 1.1.0.0
Title: IBM_Tivoli_Content_Delivery_Service_Client
Vendor: IBM
Software Type:

⊟ Installable Files

Name
TCA-Subagent CDS Client URL Handler Installable TPM v5.1

710x620

⊞ Groups

⊞ Requirements and Capabilities

⊟ Configuration Templates

Configuration Templates

⊟ TCA-Subagent CDS Client URL Handler Default Parameters TPM v5.1

Resource Type: Installation   Device Model: TCA_Subagent_SoftwareInstallationDM

Parameter	Description	Value
web_service_protocol		1
web_service_https_port		9046
web_service_host		%%TPM_SERVER_HOS
web_service_http_port		9082
peering		true

Save | Cancel

Figure 3-12 Changing the peering option of the dynamic content delivery service subagent

If the dynamic content delivery service agent is already installed, you can enable peering by editing the configuration file of the dynamic content delivery service subagent. The configuration of the subagent on the common agent is held in the file CA_HOME\subagents\cdsclient.properties. Take care when you edit this file. We do not recommend that you alter any parameters except for changing peering from false to true. After you edit the file, stop and restart the common agent. After the next dynamic content delivery service distribution, the properties file is automatically updated with all the information required for it to act as a peer.

Example 3-2 shows a dynamic content delivery service subagent configuration file. Also notice that the peer is configured to use adaptive bandwidth control.

Example 3-2 Example cdsclient.properties file after peering enabled

```
web_service_http_port=9082
user_id=maldon.itsc.austin.ibm.com
web_service_protocol=1
subnetAddr=9.3.5.0
web_service_context_root=
adaptive_bandwidth_control=true
user_web_service_url_secure=https\://helsinki.itsc.austin.ibm.com\:9046
/DownloadGrid/services/DownloadGridUserServerSOAP
download_directory=C\:\\Program
Files\\tivoli\\ep\\runtime\\agent\\subagents\\cds\\downloads
web_service_host=helsinki.itsc.austin.ibm.com
cds_port=21235
user_web_service_url=https\://helsinki.itsc.austin.ibm.com\:9046/Downlo
adGrid/services/DownloadGridUserServerSOAP
peer_services_url=https\://helsinki.itsc.austin.ibm.com\:9046/DownloadG
rid/client/PeerServices
rmi_port=2111
cache_directory=C\:\\Program
Files\\tivoli\\ep\\runtime\\agent\\subagents\\cds\\cache
peering=true
uuid=43233B054A1A3DC79C87E181A91BF231
initial_login_done=true
user_ip=9.3.5.202
client_url=http\://helsinki.itsc.austin.ibm.com\:9082/DownloadGrid/clie
nt/ClientUpgrade
web_service_https_port=9046
config_setting=false
domain=itsc.austin.ibm.com
```

3.2.7 Security

Dynamic content delivery service provides industry leading security that is superior to the MDist2 security features. It uses public key technology to delegate file access privileges to clients, which reduces network traffic between depot servers and the Management Center.

After every download, file integrity is guaranteed by validating the file's MD5 checksum. The dynamic content delivery service automatically encrypts files before entering the system. Files remain encrypted during their lifetime, within the dynamic content delivery service distribution service, which includes when they reside on the disks of depots or peers. The files are decrypted only after the target client receives the files, which ensures that critical data or applications targeted at ATM or Point-of-Sale devices are securely stored until delivered or installed.

The encryption keys are not stored with the files. In the event that a depot or peer is compromised, the file contents will remain securely encrypted. This ensures that even within a company's internal firewall system, internal threats are contained.

Secure content

The *Advanced Encryption Standard (AES)* that Tivoli Provisioning Manager dynamic content delivery service utilizes, provides some key differentiating capabilities unparalleled today in Triple DES enabled competitive products and the MDist2 distribution service.

The Advanced Encryption Standard, the block cipher ratified as a standard by National Institute of Standards and Technology (NIST), was chosen using a process markedly more open and transparent than its predecessor, the ageing Data Encryption Standard (DES). This process won plaudits from the open cryptographic community and helped to increase confidence in the security of the winning algorithm from those who were suspicious of back doors in the predecessor, DES.

The Data Encryption Standard's relatively small 56-bit key, which was becoming vulnerable to brute force attacks, was the primary motivation for a new standard. Additionally, the DES was designed primarily for hardware and is relatively slow when implemented in software. While Triple-DES avoids the problem of a small key size, it is very slow in software, and also unsuitable for limited-resource platforms such as pervasive devices.

When it was found that a 56-bit key of DES was not enough to guard against brute force attacks, Triple-DES was chosen as a simple way to enlarge the key space without a need to switch to a new algorithm.

The principal drawback of Triple-DES is that the algorithm is relatively sluggish in software. The original DES was designed for a mid-1970s hardware implementation and does not produce efficient software code. Triple-DES, which has three times as many rounds of encryption as DES, is correspondingly slower. DES and Triple-DES use a 64-bit block length. To gain efficiency and security, a larger block length is desirable. Triple-DES suffers from extremely slow performance in software. On modern processors, AES tends, in some estimates, to be around six times faster.

Finally, AES offers markedly higher security margins: a larger block size, potentially longer keys, and (as of 2005) freedom from cryptanalytic attacks.

Data integrity—Did the data change anywhere?

The Tivoli Provisioning Manager client verifies the downloaded file's integrity using a Message-Digest algorithm 5 (MD5) checksum. In cryptography, MD5 is a widely-used cryptographic hash function. As an Internet standard (RFC 1321), MD5 is employed in a wide variety of security applications, and it is also commonly used to check the integrity of files. MD5 is widely used in the software world to provide assurance that a downloaded file was not altered. This ensures that bank currency data, register transaction data, or other sensitive content is unmodified during distribution.

Many vendors consider encryption on the wire to be enough security. Of course this is important, but it is not the only point-of-attack. Staging areas, even behind corporate and government firewalls, are subject to access by a company's or agency's employees. Many vendors leave this critical point-of-attack unprotected. Dynamic content delivery service ensures that data is encrypted from the time it is put into production on the management server, in transit on the network, and stored on depots, peers, or targeted clients, until, in the case of software distribution, it is installed or made available to the application on the endpoint.

3.2.8 Wake on LAN

Just like MDist2, the Scalable Distribution Infrastructure provides a Wake on LAN feature. Wake on LAN is a broadcast protocol that can remotely power on "sleeping" targets by sending special network data packets, which enables an administrator to remotely perform maintenance and send updates to systems that were powered off by their users. To use SDI's Wake on LAN, you must enable the common agent to receive and respond to the Wake on LAN data packets.

Using Wake on LAN with Tivoli Provisioning Manager

To enable each endpoint configured for Wake on LAN to respond to the packets, ensure that the value of the Wake on LAN option in the BIOS (usually under network settings) is set to Enabled. For PCI network cards, you must attach a Wake on LAN power cord to the motherboard (where applicable).

To set up and use the Wake on LAN protocol with Tivoli Provisioning Manager, we must know the IP address, the MAC address, and the subnet of all of the common agents that we want to "wake up". Use Tivoli Provisioning Manager's Inventory Discovery to collect these attributes.

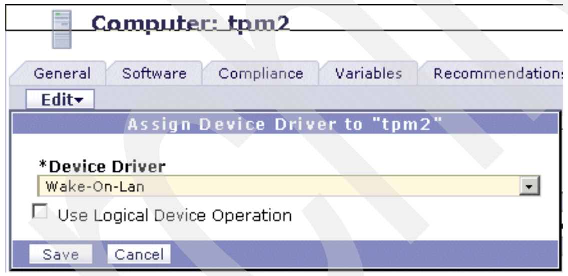
The Wake on LAN feature is provided with Tivoli Provisioning Manager version 5.1.0.1 (Fix Pack 1). A Wake on LAN .tcdriver automation package is automatically installed with the Fix Pack 1 version of Tivoli Provisioning Manager. This automation package provides a number of workflows that you can use to set up and use the Wake on LAN functionality for target common agents. Before using the Wake on LAN functionality we must assign the Wake on LAN device driver to the common agent.

Assigning the Wake On LAN device driver

To configure the target common agent for Wake on LAN using a set of predefined workflows, you must first assign the Wake on LAN device driver.

To assign the Wake on LAN device driver to a specified common agent, perform the following steps:

1. Using the Find box at the top of the navigation pane, find the common agent you want to work with, and click its name. The **General** tab is displayed.
2. Click the **Workflows** tab. Click **Edit** → **Assign Device Driver**. The Assign Device Driver to Computer dialog is displayed.
3. In the Device Driver list, select **Wake On Lan**, as shown in Figure 3-13, and click **Save**.



*Device Driver: Wake-On-Lan

☐ Use Logical Device Operation

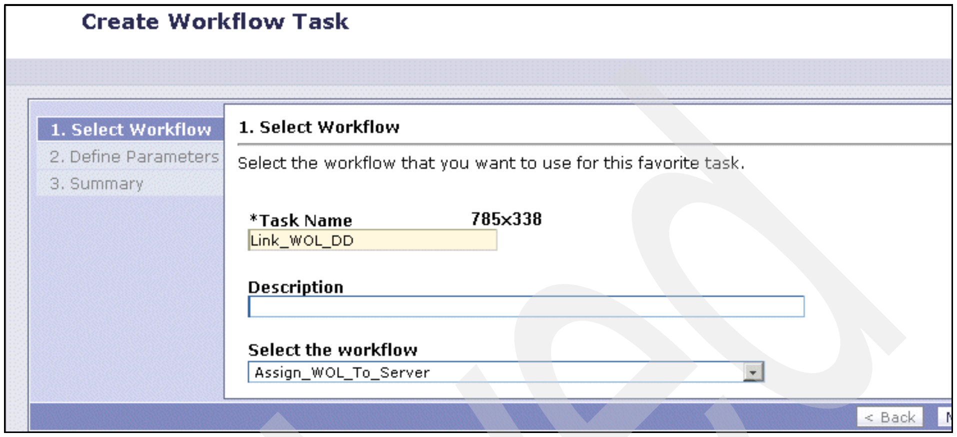
Figure 3-13 Assigning the Wake On LAN device driver

The Wake on LAN device driver is assigned to the specified common agent. The Wake on LAN-specific workflows, which are associated with this device driver, are now available to be used with the selected common agent.

You can also assign the Wake on LAN device driver to a group of common agents:

1. In the navigation pane, click **Task Management** → **Favorite Tasks**.
2. Click **Edit** → **Create Workflow Task**.

3. On the Select Workflow page (Figure 3-14), select the **Assign_WOL_To_Server.wkf** workflow, and specify a name and description for the task. Click **Next**.



Select the workflow that you want to use for this favorite task.

*Task Name 785x338
Link_WOL_DD

Description

Select the workflow
Assign_WOL_To_Server

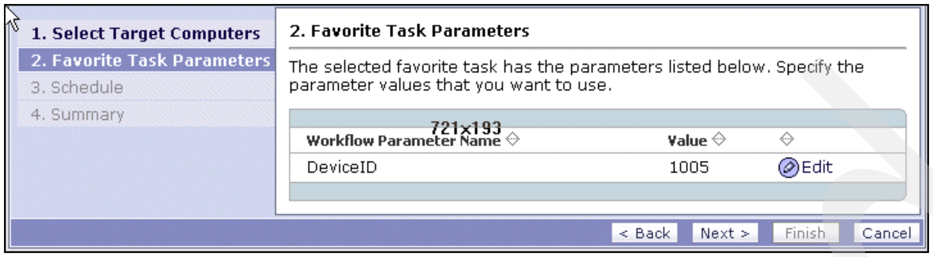
Figure 3-14 Wake on LAN—Creating a workflow task

4. On the Define Parameters page, do not select a workflow parameter for the object ID of the endpoint (this workflow does not have one). Click **Next**.
5. Verify your selections on the summary page, and then click **Finish**. The workflow task is created.
6. Run the workflow task on the group of common agents.

Note: You need the device ID of the target group to which you want the Wake on LAN device driver to be assigned.

1. Click **Task Management** → **Favorite Tasks**.
2. In the list of tasks, find the workflow task that you just created. In that row click **Run**.
3. On the Select Task page, select the task. Click **Next**.
4. On the Select Target Systems page, click **Next**. No target selection is required on this page.

5. On the Favorite Task Parameters page (Figure 3-15), specify the ID of the target group. To find the group ID, position the mouse pointer over the group name in the Inventory section. Click **Next**.


1. Select Target Computers
2. Favorite Task Parameters
3. Schedule
4. Summary
2. Favorite Task Parameters
< Back
Next >
Finish
Cancel

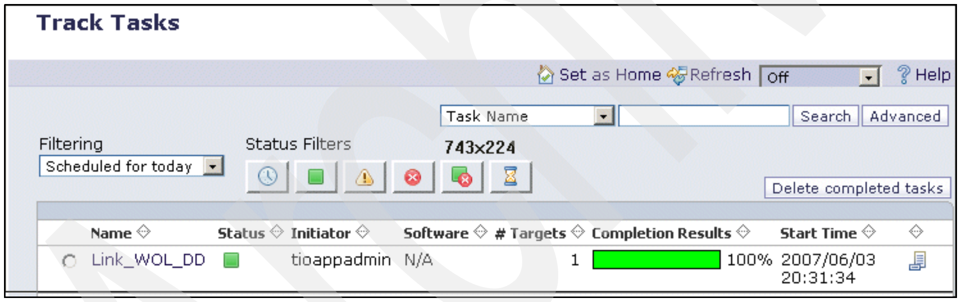
The selected favorite task has the parameters listed below. Specify the parameter values that you want to use.

Workflow Parameter Name	Value	
DeviceID	1005	Edit

Figure 3-15 Executing the Wake On LAN workflow

6. On the Schedule page, specify when you want to run the task.
7. Verify your selections on the summary page and then click **Finish**. The task runs and assigns the Wake on LAN device driver to all of the endpoints in the selected group.

   The **Track Tasks (**Figure 3-16) screen is displayed and shows the status of the task.



Name	Status	Initiator	Software	# Targets	Completion Results	Start Time	
Link_WOL_DD	🟩	tioappadmin	N/A	1	100%	2007/06/03 20:31:34	

Figure 3-16 Viewing the result of the Wake on LAN workflow

Using Wake on LAN with Tivoli Provisioning Manager

After configuring the common agents for Wake on LAN in Tivoli Provisioning Manager, there are several ways to power them on.

You can power on an individual common agent by using the **Management** button on the General tab of the computer. See Figure 3-17 on page 74.

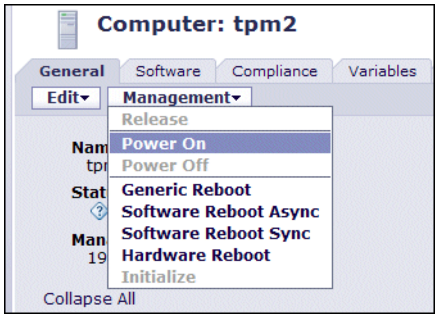


Figure 3-17 Waking up a single common agent

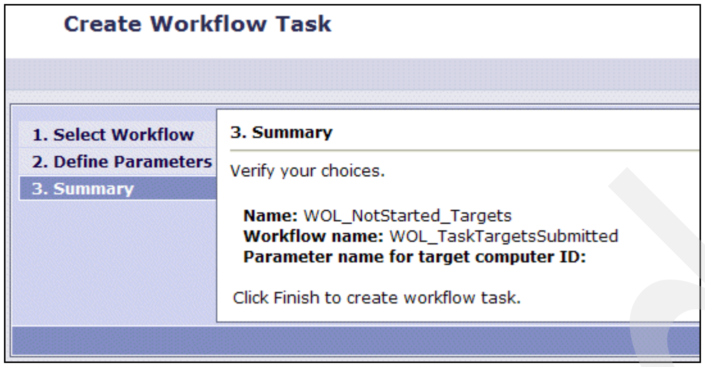
Similarly, you can power on a group of common agent endpoints by running a favorite task initiated by the *Device.PowerOn.wkf* workflow on the target group.

Finally we can power on a number of common agents in a group on which a software distribution task is run. The software distribution task might have completed successfully on most of the endpoints in the group, while some of the remaining endpoints may still be in progress. To check whether the endpoints that are still in progress are powered on or not, we can run a specific Wake On LAN workflow on the group.

We can define a favorite task using the *WOL_TaskTargetsSubmitted.wkf* workflow and run it on the endpoint task group. The workflow powers on the common agents that were shut down, which in turn allows the software distribution task to resume and complete successfully on the endpoints.

To set up the Wake On LAN favorite task:

1. In the navigation pane, click **Task Management** → **Favorite Tasks**.
2. On the Select Workflow page, select the WOL_TaskTargetsSubmitted.wkf workflow, and specify the name and description for the task. Click **Next**.
3. On the Define Parameters page, do not select a workflow parameter for the object ID of the target endpoint because this workflow does not have one. Click **Next**.


Create Workflow Task

1. Select Workflow
2. Define Parameters
3. Summary

3. Summary

Verify your choices.

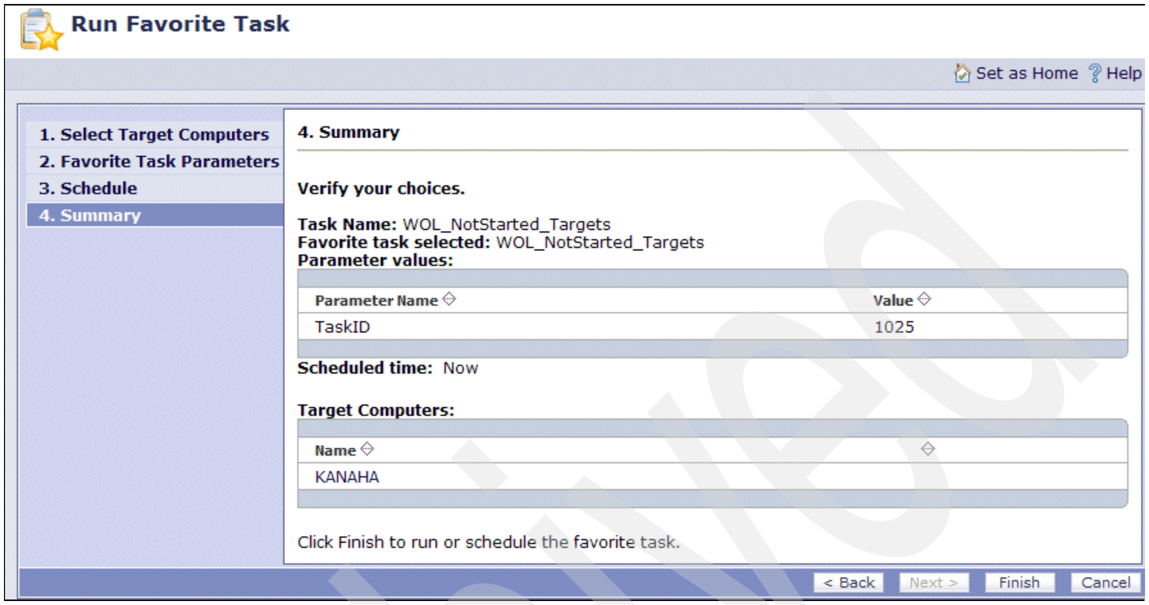
Name: WOL_NotStarted_Targets
Workflow name: WOL_TaskTargetsSubmitted
Parameter name for target computer ID:

Click Finish to create workflow task.

Figure 3-18 Creating the "WolTaskTargets" favorite task

4. Verify your selections on the summary page, and then click **Finish**. The workflow task is created.
5. We can now run the favorite task and provide it with the ID of a software distribution task that did not complete on a number of targets that are powered-off.
6. Click **Task Management** → **Favorite Tasks**.
7. In the list of tasks, find the workflow task that you just created. In that row, click **Run**.
8. On the Select Task page, select the task. Click **Next**.
9. On the Select Target Systems page, click **Next**. No target selection is required on this page.
10. On the Automation Task Parameters page, specify the Task ID value of the software distribution task, and then click **Next**. To obtain the task ID, position the mouse pointer over the task name on the Track Tasks page.
11. On the Schedule page, specify when you want to run the task.

12. Verify your selections on the Summary page (Figure 3-19), and then click **Finish**.



4. Summary

Verify your choices.

Task Name: WOL_NotStarted_Targets
Favorite task selected: WOL_NotStarted_Targets
Parameter values:

Parameter Name	Value
TaskID	1025

Scheduled time: Now

Target Computers:

Name
KANAHA

Click Finish to run or schedule the favorite task.

Figure 3-19 Waking up targets of a submitted job

The Wake On LAN task runs on the entire group and powers up the remaining endpoints. You can view the status of the task by clicking “Track Tasks”. After the targets are successfully started, the software distribution task starts automatically on these targets. The common agent checks with the device management service federated agent for applicable jobs.

3.3 Tivoli Provisioning Manager for Software Inventory collection

We explained in 3.1.4, “Scalable Collection Services” on page 46, that the Tivoli Management Framework’s SCS provides functionality to efficiently collect inventory data. In Tivoli Provisioning Manager for Software, the data collection algorithm is different, and we describe it in the next section.

When an Inventory scan completes on a common agent, the common agent sends the inventory data directly to the Tivoli Provisioning Manager server using an HTTPS post on port 9045. The inventory data is sent as a compressed XML file, a servlet on the Tivoli Provisioning Manager server receives. The receiver servlet does not process the files; instead, it simply saves the files in a queue into the data center model (DCM). Another thread on the Tivoli Provisioning Manager server, which is called the Integrator, is responsible for processing the reports and updating the related DCM resources.

If the common agent has no connection to the Tivoli Provisioning Manager server or it loses the connection to the Tivoli Provisioning Manager server, then it retries, with a maximum of 20 times, to resend the compressed XML file to the Tivoli Provisioning Manager server. Currently, you cannot customize this process.

4

Example scenarios

In this chapter, we provide a description of several real-life scenarios of migration from a Tivoli Configuration Manager-based environment to a Tivoli Provisioning Manager for Software-based environment. We describe several customer environments, and then we discuss the suggested approach for moving to a Tivoli Provisioning Manager for Software environment. We also discuss how the two environments can coexist during the migration process.

We discuss the following topics in this chapter:

4.1 Centrally managed hub and spoke environment

A centrally managed hub and spoke environment is commonly seen in Tivoli Management Framework configurations, with a hub Tivoli Management Region for all management operations and several spoke Tivoli Management Regions to handle the endpoints.

4.1.1 Environment description

There is one hub Tivoli Management Region and several spoke Tivoli Management Regions. The hub Tivoli Management Region has all the management profiles, such as software packages and inventory profiles, defined on it. There is one inventory database.

All the managed endpoints are defined on the spokes. The number of spokes is dictated by the total number of endpoints to be managed. This environment, which we illustrate in Figure 4-1, is a reasonably large customer with five spokes and approximately 65,000 endpoints.

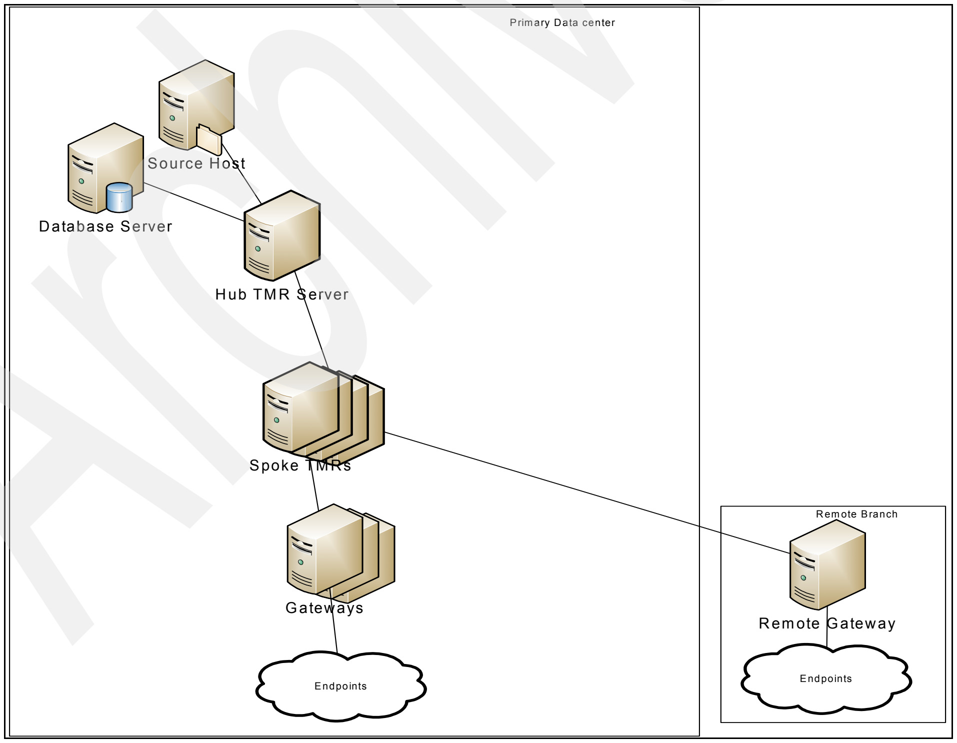


Figure 4-1 Hub and spoke environment

4.1.2 Suggested migration approach and architecture

In this section, we describe the suggested migration approach and architecture for the example scenario.

Phase 1: Tivoli Provisioning Manager for Software installation and configuration

Phase 1 involves installing and configuring Tivoli Provisioning Manager for Software:

- Install a single Tivoli Provisioning Manager for Software server to manage all the existing Tivoli Management Regions.
- Set up the coexistence environment and replicate the Tivoli Management Framework and the Tivoli Configuration Manager data, which we describe in Chapter 5, "Implementing coexistence" on page 93.
- Migrate the required user information from Tivoli Configuration Manager to Tivoli Provisioning Manager for Software.

Now, you can use the Tivoli Provisioning Manager for Software console to manage the TMA endpoints through the existing Tivoli Configuration Manager environment.

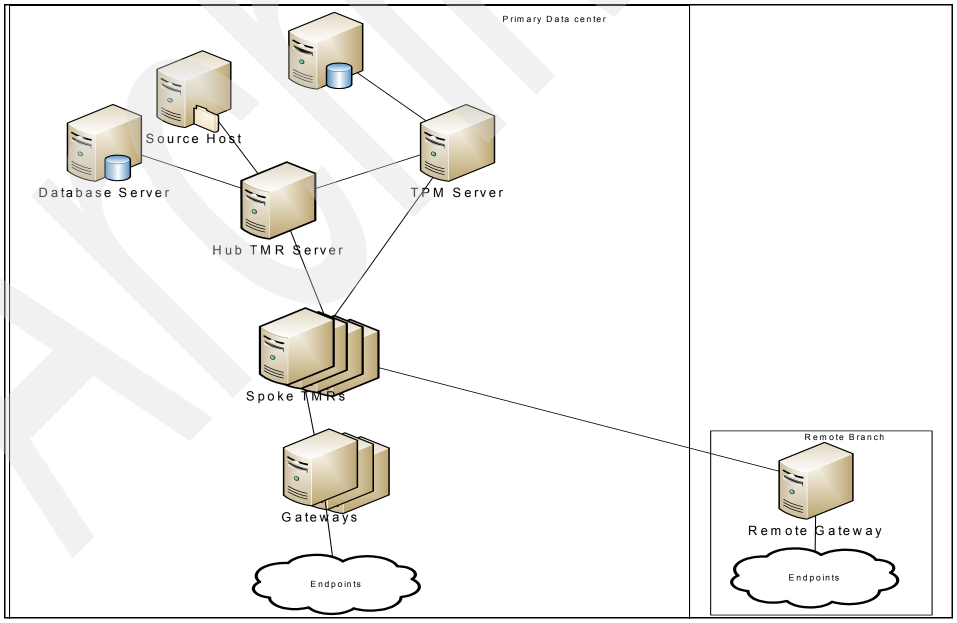


Figure 4-2 Hub and spoke environment migration phase 1

Phase 2: Set up Tivoli Provisioning Manager for Software infrastructure

Phase 2 involves setting up the physical components of Tivoli Provisioning Manager for Software:

- Deploy the SOA server components, if you did not complete this as part of the first phase.
- Deploy one or more depots using the Tivoli Provisioning Manager for Software console. You can install the depots on existing gateways.
- Deploy the common agent to some endpoints using the Tivoli Provisioning Manager for Software console, and activate the common agents. (If there are any problems, switch the common agent device back to using the TMA.)

At this point, you started using the new Tivoli Provisioning Manager for Software SOA environment. After you install and activate a common agent, Tivoli Provisioning Manager for Software automatically uses the SOA environment for Software Distribution and Inventory (Discovery) actions.

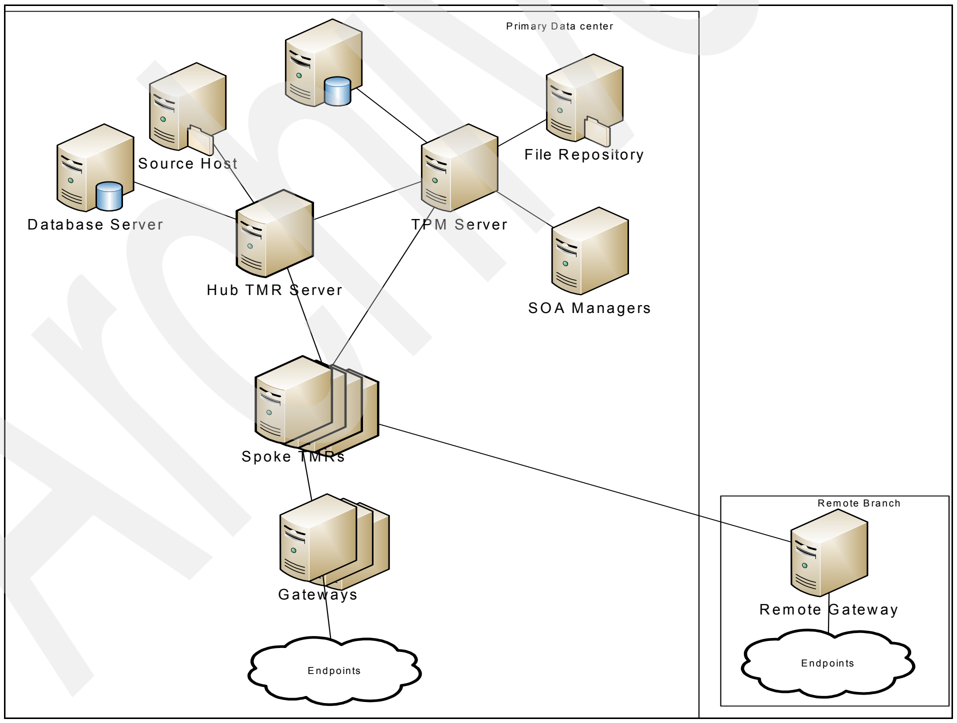


Figure 4-3 Hub and spoke environment migration phase 2, SOA components added

Phase 3: Complete common agent deployment and configuration

This phase involves moving all of the remaining infrastructure over to CDS.

- Migrate the remaining endpoints over to common agents at what ever pace is convenient. Where required, enable peering on some common agents, which reduces the number of depots required.
- Deploy the CDS infrastructure, which involves:
  - Installing the required depots. Generally, but especially if peering is enabled, you are unlikely to need as many depots as you have gateways.
  - Defining CDS regions and zones. You can consider making each Tivoli Management Region a region if the logical design of the Tivoli Management Regions makes this appropriate. Mapping physical locations such as branches to zones is a good strategy.

At this point, the spoke infrastructure is replaced by the new CDS components. Management operations no longer go through the Tivoli Configuration Manager environment; therefore, you can retire it, which leaves the environment as shown in Figure 4-4.

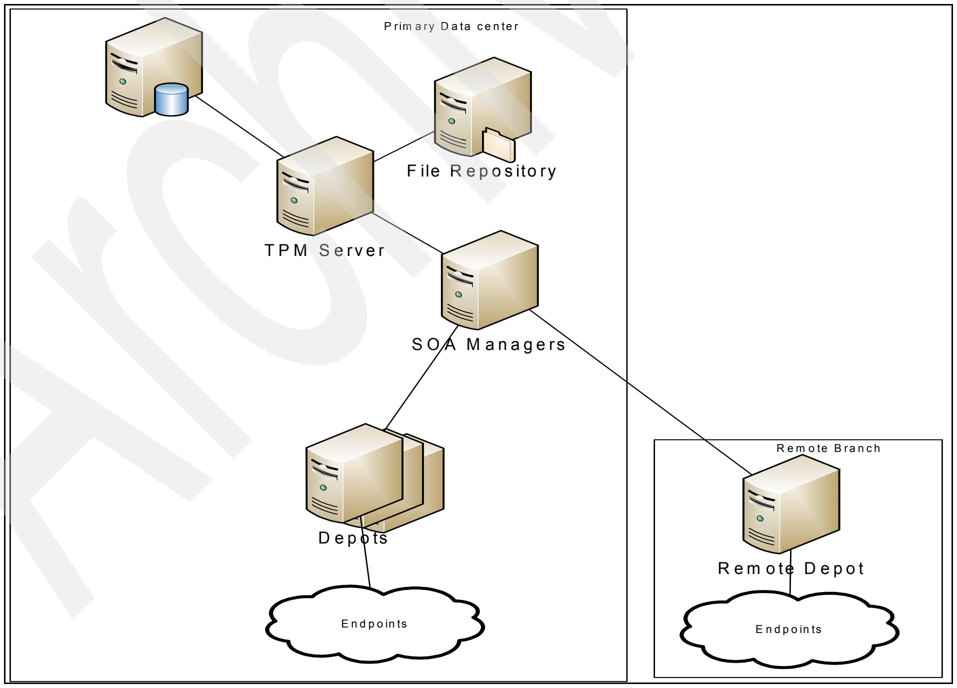


Figure 4-4 Hub and spoke environment—migration complete

4.1.3 Hub only database

Some customers with the environment of a single hub and several spokes, as shown in Figure 4-1 on page 80, perform all of their management operations and the monitoring and reporting of these operations through the hub Tivoli Management Region. In this case there is no requirement for the spoke Tivoli Management Regions to access a database. The spoke Tivoli Management Regions in these environments have no RIMs defined and no database clients installed. There are two possible approaches for this type of environment:

- Install the database client on each spoke Tivoli Management Region server, and add the inventory RIMs. Then proceed as before, linking all the spokes through the central database.
- Link the hub Tivoli Management Region server to allow the import of resources, such as software packages, profile managers, and so on. In this case, you cannot use the Tivoli Provisioning Manager for Software environment to drive the Tivoli Management Framework environment, which has a few disadvantages:
  - A period of parallel operation is required where the same distributions have to be initiated from both the Tivoli Management Framework and the Tivoli Provisioning Manager for Software environments.
  - You cannot use the Tivoli Provisioning Manager for Software provided method of migrating endpoints from Tivoli management agents to common agents. Instead, do one of the following:
    - Discover the endpoint targets using SMB or secure shell (ssh), and then install the common agent using the Deployment Engine.
    - Create a common agent software package in the Tivoli Management Framework environment, and distribute that package to install the common agent. Next, discover the common agents from the Tivoli Provisioning Manager for Software environment.

4.2 Large global environment without coexistence

This section describes a large environment that is distributed across several countries. The large environment uses a mixture of local and central management operations, where the customer does not want to use coexistence.

4.2.1 Environment description

The environment manages approximately 100,000 endpoints distributed across nine countries. Each country has two data centers (a primary and a backup). All the users' workstations are located in the primary center. The backup is a lights-out location used for testing and disaster recovery.

The headquarters site hosts a hub Tivoli Management Region with spoke Tivoli Management Regions in each of the other countries. Each Tivoli Management Region has its own database. The headquarters site manages 20,000 local endpoints. The hub can perform distribution operations to any endpoint in the enterprise. The spoke Tivoli Management Regions are also perform distribution operations to their countries' local endpoints using a subset of the software packages that are used at the hub.

The customer does not want to run a coexisting Tivoli Provisioning Manager for Software and Tivoli Configuration Manager environment, which means the approach involves building a new Tivoli Provisioning Manager for Software environment. Then, after the new environment is up, we retire the Tivoli Management Framework environment.

The customer wants the new environment to provide the current hub and spoke management capability so that local administrators and central administrators can perform management operations.

4.2.2 Suggested migration approach and architecture

This section covers the suggested migration approach and architecture for a large global environment.

Architecture

Deploy two Tivoli Provisioning Manager for Software management systems in each country, one primary and one backup. Size each system for the number of endpoints managed in that country.

With this configuration you have nine independent Tivoli Provisioning Manager for Software installations.

The problem to solve with this configuration is that Tivoli Provisioning Manager for Software cannot provide the facility for the headquarters to perform operations on all the remote endpoints through the regional Tivoli Provisioning Manager for Software servers. There are several solutions to this problem:

- Use the GUI and SOAP commands to remotely connect to each Tivoli Provisioning Manager for Software installation individually, and perform operations on the devices.
- The Tivoli Provisioning Manager for Software in the headquarters can be a sort of hub, which is used to hold all the definitions of all the software packages, with replication of required software packages to remote sites using custom workflows to replicate the software definitions and images (software package blocks).
- It might be possible to automate running remote SOAP commands on the regional Tivoli Provisioning Manager for Software servers using custom workflows, which gives a configuration similar to Figure 4-5.

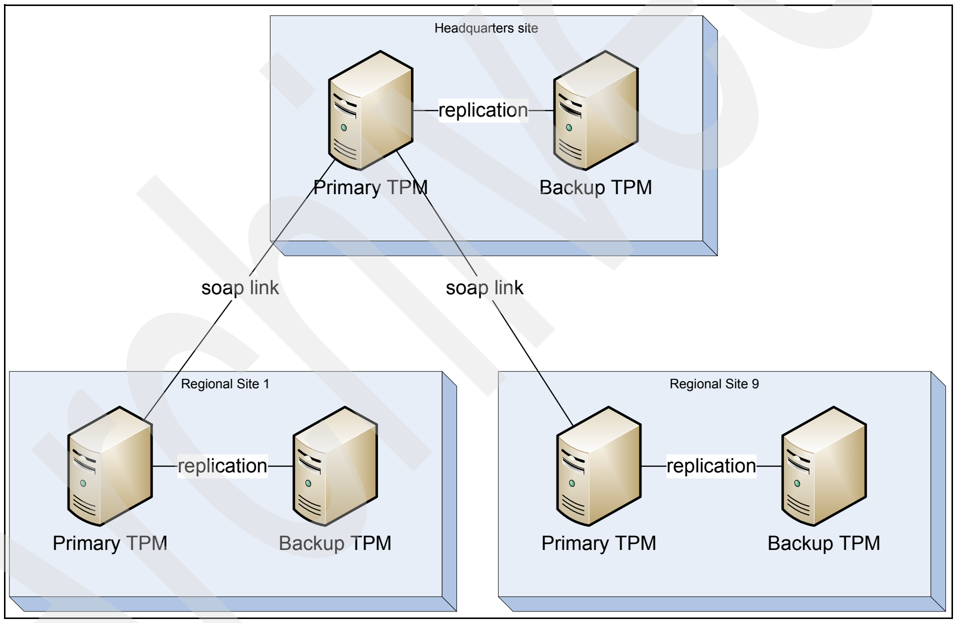


Figure 4-5 Global environment

Deploying common agents

There are two options for deploying the common agent:

- Use the standard Tivoli Provisioning Manager for Software method from the GUI.

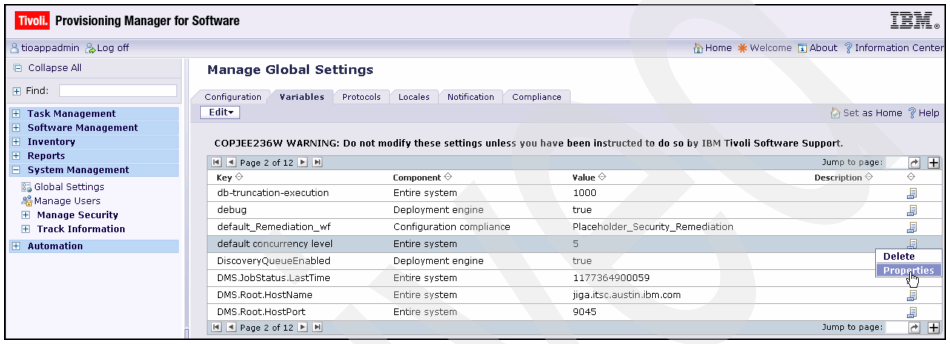
  This allows you to address a small group of targets at once because the Deployment Engine handles the deployment. By default the *concurrency* *level* is set to five, which you can increase to manage more targets. Do not increase the concurrency level above a maximum of 30.

To change the concurrency level:

a. Select **System Management** → **Global Settings** → **Variables**.

b. Find the value **default_concurrecy_level**.

c. Select **Properties**, as shown in Figure 4-6, change the value, and then click **Save**.



COPJEE236W WARNING: Do not modify these settings unless you have been instructed to do so by IBM Tivoli Software Support.

Key	Component	Value	Description
db-truncation-execution	Entire system	1000	
debug	Deployment engine	true	
default_Remediation_wf	Configuration compliance	Placeholder_Security_Remediation	
default concurrency level	Entire system	5	
DiscoveryQueueEnabled	Deployment engine	true	
DMS.JobStatus.LastTime	Entire system	1177364900059	
DMS.Root.HostName	Entire system	jiga.itsc.austin.ibm.com	
DMS.Root.HostPort	Entire system	9045	

Figure 4-6 Selecting properties for the default_concurrency_level

- Use the existing common agent architecture.

  To use the Tivoli Provisioning Manager for Software console to install the common agent through the Tivoli Configuration Manager environment requires that you temporarily set up the coexistence environment. Run a light, one-off replication that replicates only the software packages and the TMAs.

  An alternative is to import the common agent SPB into the Tivoli Configuration Manager environment, and use the standard Tivoli Configuration Manager mechanisms to distribute it. Next, run a common agent discovery to import the devices into Tivoli Provisioning Manager for Software.

4.3 Slow links and branch offices

In this section, we describe an environment where many branches, which are at the end of slow links, are managed.

4.3.1 Environment description

There are approximately 10,000 endpoints. 2,000 of these are in a corporate headquarters, and the remaining are in 450 branch offices. Each branch is behind a firewall, and the endpoints are connected over slow links. There is one Tivoli Management Region and 50 regional gateways located throughout the US to handle the endpoints in the branch offices. Distributions require traversal of multiple firewalls.

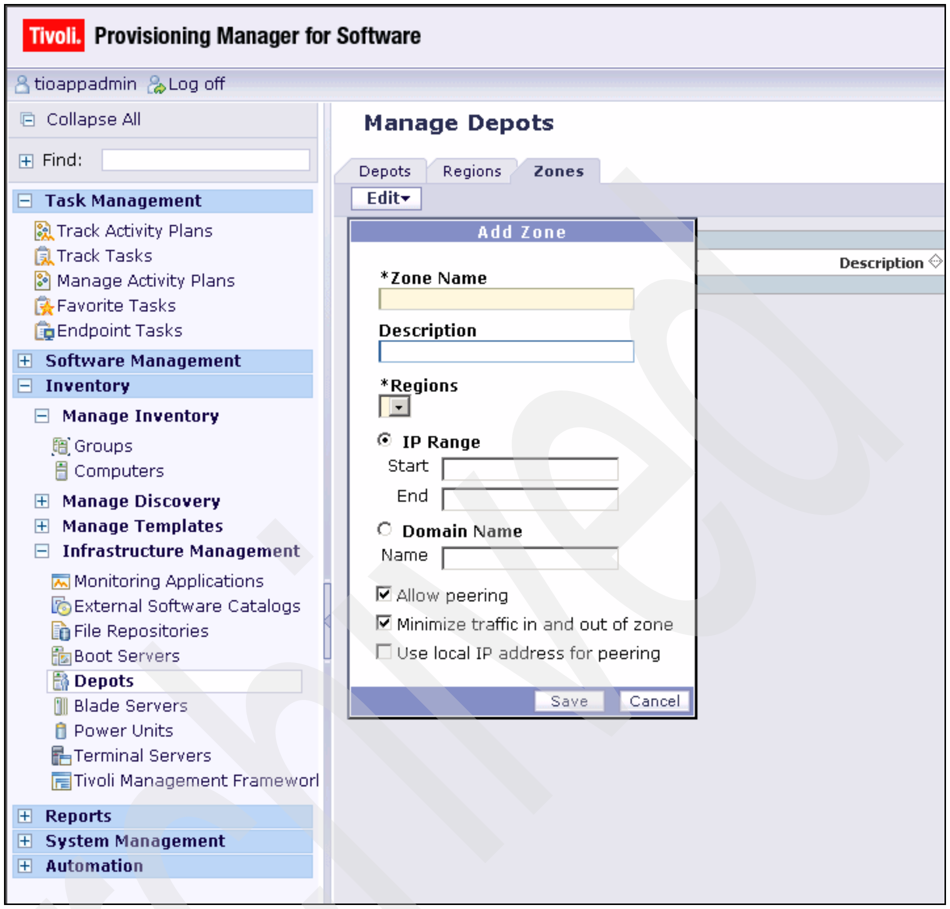
4.3.2 Suggested migration approach and architecture

Install a single Tivoli Provisioning Manager for Software server to replace the Tivoli Management Region server. The Tivoli Provisioning Manager for Software server will then manage all the branches. The approach is similar to the approach we described for 4.1, “Centrally managed hub and spoke environment” on page 80; however, this environment provides two additional problems that we did not encounter in the previous environment: the slow links to the branch and the firewalls.

Branch specific recommendations

For the specific problem of managing slow links, we recommend that you enable peering. Define each branch as a zone, and in the add zone dialog, select the **Allow peering** and **Minimize traffic in and out of zone** options, as shown in Figure 4-7 on page 89.

Note: Turning on the Allow peering and the Minimize traffic in and out of zone options will restrict the deployment of software into a zone so that the software only passes into the zone once. Only one peer downloads the software into the zone. All other devices wait for the first peer (this is a random selection from the available peers) to download the software. All other devices then load the software from the peer (or any other peer that already loaded the software from the first peer).


Tivoli. Provisioning Manager for Software

tioappadmin Log off

Manage Depots

Depots | Regions | Zones

Add Zone

*Zone Name

Description

*Regions

(●) IP Range
Start
End

( ) Domain Name
Name

[✓] Allow peering
[✓] Minimize traffic in and out of zone
[ ] Use local IP address for peering

Figure 4-7   Zone settings for a branch

Firewall specific recommendations

Tivoli Provisioning Manager for Software only uses six-to-eight ports, depending on the features used. All communications are encrypted and the ports are fixed. With this in mind our recommended solution for communication through a firewall is that you arrange for these ports to be opened. For more details on the default ports used, see Chapter 10, “Working in a firewall environment” on page 291.

For a DMZ type installation, consider putting the Tivoli Provisioning Manager for Software server and core databases in the core zone and having the SOA components in the dematerialized zone (DMZ). In this way the endpoints only need to traverse one firewall into the DMZ, and they never initiate any communications into the core zone, as shown in Figure 4-8 on page 90.

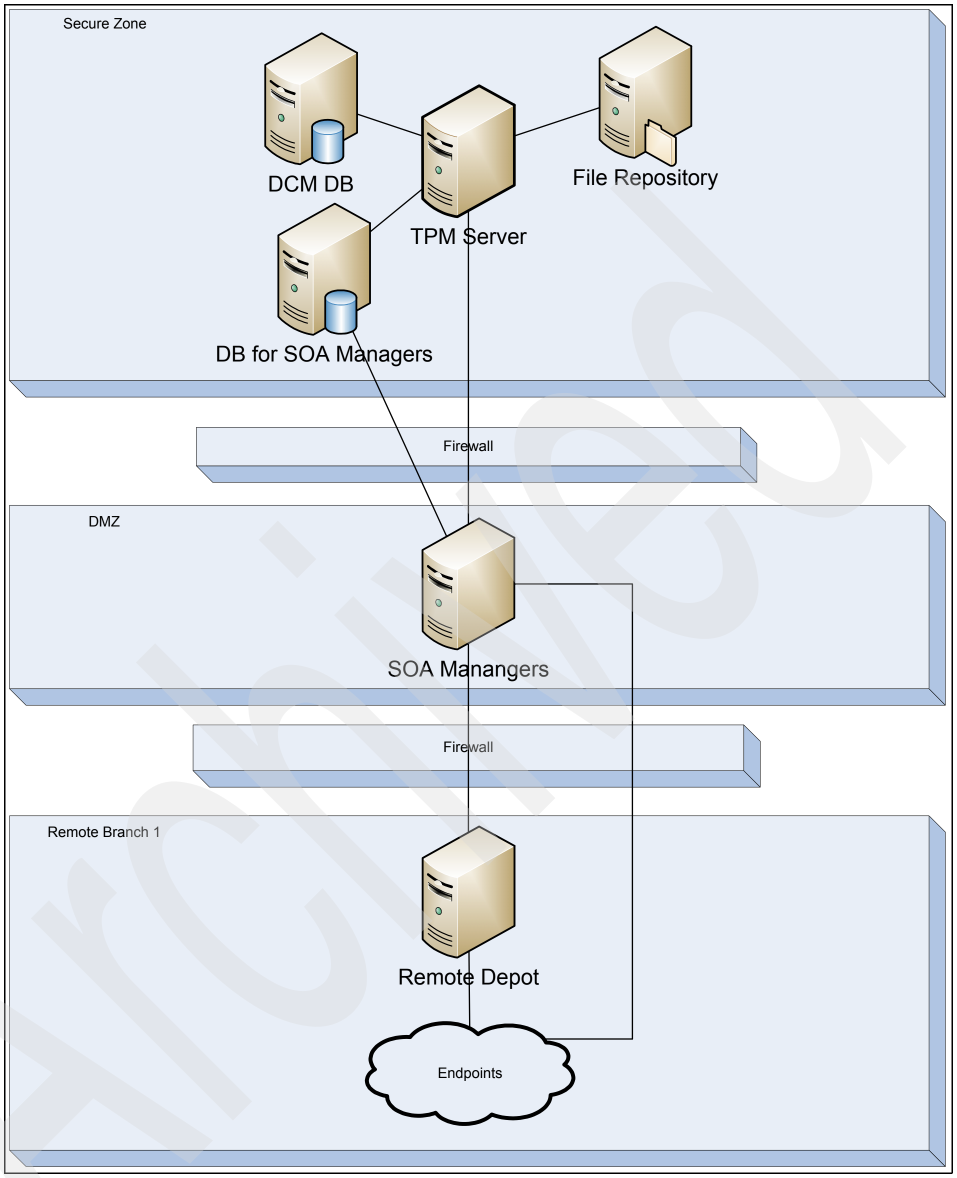


Figure 4-8 Firewall environment

The alternative, which can also be used for environments where several firewalls must be traversed, is the Tivoli Provisioning Manager firewall proxy solution, which is similar to the Tivoli Management Framework solution and can address any requirement for all communications to be initiated from one side of the firewall. For more details see Chapter 10, “Working in a firewall environment” on page 291.

4.4 Data center

This section describes a simple data center environment where the customer has some custom requirements.

4.4.1 Environment description

There are approximately 500 servers in a data center and about 100 servers in a back-up data center. There is a total of roughly 2000 endpoints. A single Tivoli Management Region, with a couple of gateways, manages the endpoints.

The customer performs weekly inventory scans using custom Management Information Formats (MIF) to collect additional inventory data which is utilized to update the assets management system. The customer has plans to also begin population of a new IBM Tivoli Change and Configuration Management Database (CCMDB).

4.4.2 Suggested migration approach and architecture

Install a single Tivoli Provisioning Manager server, which replaces the Tivoli Management Region and manages the data center.

Use the approach we provided in 4.1, “Centrally managed hub and spoke environment” on page 80. Tivoli Provisioning Manager for Software provides support of custom MIF files for collecting additional inventory data. Inventory Discovery is being enhanced to support database extensions, and a command-line tool is being provided to define them. The Inventory Discovery feature imports your MIFs and automatically creates the needed DCM extensions. You are responsible for distributing custom scripts that run on endpoints to collect data. Then you can specify extensions to run when you create Inventory Discovery configurations.

Tivoli Provisioning Manager for Software also provides discovery configurations for exporting data to CCMDB.

4.5 Environment with unsupported database configuration

This section describes the approach you can take when the Tivoli Management Framework environment uses a database configuration that Tivoli Provisioning Manager for Software does not support, which prevents the linking of the two environments.

4.5.1 Environment description

The existing Tivoli Configuration Manager environment has a database and operating system configuration that is not supported in Tivoli Provisioning Manager for Software, such as Oracle® on AIX or MS SQL Server™ on Windows.

4.5.2 Suggested migration approach and architecture

If possible, consider using an alternative operating system that is supported. For example, you can use an Oracle on Solaris™ or Oracle on Linux® database to link to a Tivoli Configuration Manager Oracle on AIX database. If you cannot use an alternative operating system, then it is possible to request support as a feature or enhancement.

The alternative is to perform a parallel running migration. You can import software package blocks (SPBs) from the Tivoli Configuration Manager environment into the Tivoli Provisioning Manager for Software environment. During the parallel running, send deployments to TMAs using the Tivoli Configuration Manager environment, and send deployments to the common agents using the new Tivoli Provisioning Manager for Software environment.

Deploying common agents

In a parallel running migration, you can create and import an SPB for the new common agent into the Tivoli Configuration Manager environment, so that you can use the Tivoli Configuration Manager environment to distribute the agent to the endpoints. Then use common agent discovery to activate the devices in Tivoli Provisioning Manager for Software. Alternatively, you could use the Distribution Engine to deploy the common agent to the devices.

In a parallel running environment, make sure that distributions to devices did not occur using both the Tivoli Provisioning Manager for Software and the Tivoli Configuration Manager environment.

5

Implementing coexistence

In this chapter, we describe the actions you must take to synchronize Tivoli Provisioning Manager for Software V5.1 and Tivoli Configuration Manager V4.2.3, which means making Tivoli Management Region and Tivoli Configuration Manager objects available in the Tivoli Provisioning Manager environment.

We cover the following topics in this chapter:

5.1 Creating a coexistence environment

Tivoli Management Framework and Tivoli Provisioning Manager environments are based on different infrastructures; however, they both can coexist because Tivoli Provisioning Manager provides resources to import and handle Tivoli Management Framework and Tivoli Configuration Manager objects.

The mechanism is a variation of an Extract, Transform, and Load (ETL) that extracts data from outside sources (frameworks and Relational Databases), Transforms it to fit business needs, and Loads it into the Tivoli Provisioning Manager for Software Data Center Model or DCM. You can extract the data from one or many Tivoli Management Regions and one or many Tivoli databases.

Data Replication brings the following types of information into the DCM:

- Organizational and hierarchical structures (such as policy regions and profile managers) as Data Center Model Groups.
- Operational artifacts (such as software packages and endpoints) as Tivoli Provisioning Manager for Software and computers.

If you want to implement Tivoli Provisioning Manager for Software version 5.1 and also keep working on Tivoli Configuration Manager version 4.2.3, you must create a link between them and replicate data from Tivoli Management Framework and Tivoli Configuration Manager databases into Tivoli Provisioning Manager Data Center Model or DCM.

To configure coexistence, first you must set up your environment using the following steps:

1. Install Tivoli Provisioning Manager for Software V5.1
   - Because the installation of Tivoli Provisioning Manager V5.1 and Tivoli Provisioning Manager for Software V5.1 is identical, we do not repeat the installation steps here. You can refer to *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261*,* Chapter 3 for detailed information about the installation steps.
   - Installing Tivoli Provisioning Manager for Software V5.1 is almost the same as installing the classic Tivoli Provisioning Manager V5.1; however, there are some differences between both, for example:
     - Some of the functionality that comes with Tivoli Provisioning Manager V5.1 is not available to Tivoli Provisioning Manager for Software V5.1.
     - The bridging functionality between Tivoli Management Framework and Tivoli Provisioning Manager for Software V5.1 only comes with the Tivoli Provisioning Manager for Software V5.1 product, but IBM plans

to incorporate this feature into Tivoli Provisioning Manager V5.1 in Fix Pack 2.

Important: The bridging or coexistence functionality only applies to a regular installation of Tivoli Provisioning Manager for Software V5.1, Enterprise installation. The coexistence is not available for Tivoli Provisioning Manager for Software V5.1, Fast Start installation.

– You must install Tivoli Provisioning Manager for Software V5.1 using the Topology Installer Launcher on Windows platforms. The Tivoli Management Framework database should be the same vendor as Tivoli Provisioning Manager for Software database for a coexistent environment. Depending on the operating systems, ensure that you have the following databases installed:

Table 5-1 Operating systems and supported databases

Operating system	Database
► Windows 2003 Server (Standard Edition Service Pack 1/Enterprise Edition Service Pack 1) ► AIX 5.2 ML7/AIX 5.3 ML1, ML3, ML5 ► Linux RHEL AS4 Update 3 ► Linux SLES9 Enterprise Edition SP3 ► Solaris 9 only ► Linux RHEL4 Update 3/SLES9 Enterprise Edition SP3 (iSeries® or pSeries®) ► zSeries® 32 or 64bit - SLES9 Enterprise Edition SP3	IBM DB2
► Linux RHEL AS4 Update 3 ► Solaris 9 or 10	Oracle

Note: The only operating systems that support Oracle as the relational database are Sun™ and Linux RH4. You can integrate the Tivoli Provisioning Manager for Software V5.1 on one of these environments running with Oracle and a Tivoli Management Framework environment that uses an Oracle database for storing the data (not necessarily the same OS). A mix of IBM DB2 and Oracle for Tivoli Configuration Manager and Tivoli Provisioning Manager for Software databases is not supported.

2. Install Tivoli Provisioning Manager for Software V5.1 Fix Pack 2, as described in 5.1.1, “Preparing the Tivoli Provisioning Manager for Software environment” on page 96.

    Note: You do not need to install the Tivoli Configuration Manager automation server. However, if it is already installed, you can dismiss the Tivoli Configuration Manager automation server if you are working in a Tivoli Configuration Manager version 4.2.3 coexistence environment with Tivoli Provisioning Manager for Software V5.1 Fix Pack 2.

3. Install Tivoli Management Framework 4.1.1 Fix Pack 6 and also Tivoli Configuration Manager V4.2.3 Fix Pack 4, as described in 5.1.2, “Preparing the Tivoli Management Framework environment” on page 97.
4. Replicate Tivoli Management Framework data as described in 5.3, “Replicating the Tivoli Management Framework data” on page 111.
    - This step replicates the existing Tivoli Management Framework objects into the Tivoli Provisioning Manager for Software DCM database.
    - The Tivoli Management Framework databases remain the same after the replica.
5. Configure user security settings, as described in 5.6, “Configuring user security settings” on page 130.

At this point you can manage the Tivoli Management Framework objects from the Tivoli Provisioning Manager for Software Web User Interface.

5.1.1 Preparing the Tivoli Provisioning Manager for Software environment

Before you proceed with the Tivoli Provisioning Manager for Software V5.1 Fix Pack V5.1.0.2 or Fix Pack 2 installation, thoroughly review the list of prerequisites in the README file.

Remember that to take advantage of the support provided for the Tivoli Management Framework setup and data replication to Tivoli Provisioning Manager for Software, ensure that your Tivoli Configuration Manager environment is at the required level and that it has the required compatibility interim fixes or fix pack 3 installed.

We recommend that you install Tivoli Provisioning Manager for Software Fix Pack 2 for the improvements and new functionalities that this version provides. This IBM Redbooks publication is based on Tivoli Provisioning Manager for Software V5.1 Fix Pack 2.

Important: The Fix Pack 2 only applies to a regular installation of Tivoli Provisioning Manager for Software (an installation with WebSphere Application Server as the application server). This means that Tivoli Provisioning Manager for Software V5.1 Fix Pack 2 is not available for a Fast Start installation.

5.1.2 Preparing the Tivoli Management Framework environment

To implement a Tivoli Provisioning Manager for Software and Tivoli Configuration Manager coexistence, you need two updates on the Tivoli Management Framework side:

- Tivoli Management Framework 4.1.1 Fix Pack 6
- Tivoli Configuration Manager 4.2.3 Fix Pack 4

Installing the Tivoli Management Framework V4.1.1 Fix Pack 6

Before you proceed with the Tivoli Management Framework 4.1.1 Fix Pack 6 installation, thoroughly read the instructions provided in the README files.

Tivoli Management Framework 4.1.1 Fix Pack 6 contains:

- 4.1.1-LCF-0042, which you install on the Tivoli gateways
- 4.1.1-TMF-0061, which you install on the managed nodes with JRIM and JCF components installed
- 4.1.1-TMF-0080, which you install on Tivoli servers, managed nodes, and gateways

You can download Tivoli Management Framework 4.1.1 and Fix Pack 6 from the following Web address:

`ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP06`

Installing the Tivoli Configuration Manager V4.2.3 Fix Pack 4

If you have an existing installation of Tivoli Configuration Manager V4.2.3, then the Tivoli Provisioning Manager for Software V5.1 requires Tivoli Configuration Manager V4.2.3 Fix Pack 04. You can download Tivoli Configuration Manager V4.2.3 Fix Pack 04 from the following Web address:

`ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.2.3/4.2.3-TIV-TCM-FP0004`

Before you proceed with the Tivoli Configuration Manager V4.2.3 Fix Pack 04 installation, get further instructions from the *4.2.3-TIV-TCM-FP0004.pdf* publication, which you can download from the following Web address:

`ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.2.3/4.2.3-TIV-TCM-FP0004/4.2.3-TIV-TCM-FP0004_docs.tar`

Important: Thoroughly review the section *“Updating the Inventory Schema” on 4.2.3-TIV-TCM-FP0004.pdf,* under *“Installation”.*

A new resource called ReportManager is included in the Software Distribution product. You need ReportManager to enable the Tivoli Configuration Manager and Tivoli Provisioning Manager coexistence. ReportManager provides the reports to Tivoli Provisioning Manager when distributions are initiated by Tivoli Provisioning Manager itself.

To enable the integration, execute the $BINDIR/TME/ReportManager/rep_<*DB_Vendor*>.sql script to update the inventory database. Supported database vendors are DB2 and Oracle.

The rep_*vendor*.sql script

The scripts are located in the following paths:

- $BINDIR/TME/ReportManager for UNIX
- %BINDIR%\TME\ReportManager for Windows

Example 5-1 shows the execution of the `rep_db2.sql` script on the AIX 5.3 system called milan, which is also our Tivoli Management Region server in this test environment.

Example 5-1 rep_db2.sql script execution

```
[root@milan][/]-> . /etc/Tivoli/setup_env.sh

Run the “wgetrim inv_query” command to find the information regarding
the RDBMS User and the Database ID:

[root@milan][/etc/Tivoli]-> wgetrim inv_query
RIM Host:        milan
RDBMS User:      db2inst1
RDBMS Vendor:    DB2
Database ID:     inv_db
Database Home:   /home/db2inst1/sqllib
Server ID:       tcpip
Instance Home:   ~db2inst1
Instance Name:   db2inst1
```

```
You can now use these data to connect to the inv_db database after su
as db2inst1.

[root@milan][/]-> su - db2inst1
[db2inst1@milan][/home/db2inst1]-> db2 connect to inv_db user db2inst1
Enter current password for db2inst1:

   Database Connection Information

 Database server        = DB2/6000 8.2.4
 SQL authorization ID   = DB2INST1
 Local database alias   = INV_DB

The rep_db2.sql script is located in $BINDIR/TME/ReportManager
directory on the TMR server machine.

[db2inst1@milan][/Tivoli/usr/local/Tivoli/bin/aix4-r1/TME/ReportManager
]-> db2 -t -f rep_db2.sql
DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.
```

For an Oracle database, you execute the script with the following command:

```
sqlplus “sys/<sys_password>@server as sysdba” @rep_ora.sql
```

The command logs on as sysdba database user to update the database with the tables required by the ReportManager.

The *sys_password* represents the password of the sys database user, and *server* is the name of the Oracle server.

Refer to 5.7, “Report Manager” on page 135, for more details about this new component that Tivoli Management Framework Fix Pack 6 introduced.

Tivoli Management Framework Fix Pack 6 provides further functionality on the source hosts: the implementation of the FileRepository.GetFile and FileRepository.PutFile logical device operations (LDO) on the Tivoli Management Framework source hosts. The following list contains the scenarios where these functions are used:

- Software Package Editor download of an SPB from a migrated Tivoli Management Framework file repository (Source Host) → getFile

- Software Package Editor upload of a new SPB to a migrated Tivoli Management Framework file repository (Source Host) → putFile
- Deployment of a migrated SPB to OAMPi targets → getFile
- Deployment of a new SPB (created on Tivoli Provisioning Manager) to Tivoli Management Framework endpoints → putFile

Example 5-2 shows how you can confirm that Fix Pack 04 is installed in the Inventory and Software Distribution components.

Example 5-2 Checking for installation of Tivoli Configuration Manager V4.2.3 Fix Pack 04

```
Log on to the system and set the tivoli environment:
[root@milan][/]-> . /etc/Tivoli/setup_env.sh

Check the output of the command "wlsinst -P | grep 4.2.3".
If the Fix Pack 04 is installed you will see an output like the
following:
[root@milan][/]-> wlsinst -P | grep 4.2.3
Activity Planner, Version 4.2.3, Fix Pack 4.2.3-APM-FP02 (U806551 - 2006/06)
Change Manager, Version 4.2.3,  Fix Pack 4.2.3-CCM-FP02 (U806551 - 2006/06)
Scalable Collection Service, Version 4.2.3, Fix Pack 4.2.3-CLL-FP02 (U806551 -
2006/06)
Inventory, Version 4.2.3, Interim Fix 4.2.3-INV-F2P1
Inventory, Version 4.2.3, Fix Pack 4.2.3-INV-FP02 (U806551 - 2006/06)
Inventory Gateway, Version 4.2.3, Fix Pack 4.2.3-INVGW-FP02 (U806551 - 2006/06)
Software Distribution Gateway, Version 4.2.3, Fix Pack 4.2.3-SWDGW-FP02
(U806551 - 2006/06)
Software Distribution Software Package Editor, Version 4.2.3,  Fix Pack
4.2.3-SWDJPS-FP02 (U806551 - 2006/06)
Software Distribution, Version 4.2.3, Interim Fix 4.2.3-SWDSRV-F2P1
Software Distribution, Version 4.2.3, Fix Pack 4.2.3-SWDSRV-FP02 (U806551 -
2006/06)
Scalable Collection Service, Version 4.2.3, Fix Pack 4.2.3-TIV-CLL-FP0004
(U8109381 - 2007/03)
Inventory, Version 4.2.3, backward compatibility patch Fix Pack
4.2.3-TIV-INV-COMP-FP0004 (U810931 - 2007/03)
Inventory, Version 4.2.3, Fix Pack 4.2.3-TIV-INV-FP0004 (U810931 - 2007/03)
Inventory Gateway, Version 4.2.3, Fix Pack 4.2.3-TIV-INVGW-FP0004 (U810931 -
2007/03)
Software Distribution Gateway, Version 4.2.3, Fix Pack 4.2.3-TIV-SWDGW-FP0004
(U810931 - 2007/03)
Software Distribution, Version 4.2.3, Fix Pack 4.2.3-TIV-SWDSRV-FP0004 (U810931
- 2007/03)
Scalable Collection Service, Version 4.2.3
```

```
Check the output of the command “wlookup -ar PatchInfo | grep 4.2.3”.
If the Fix Pack 04 is installed you will see an output like the
following:

[root@milan][/]-> wlookup -ar PatchInfo | grep 4.2.3
4.2.3-APM-FP02  1536006190.1.899#TMF_Install::PatchInfo#
4.2.3-CCM-FP02  1536006190.1.900#TMF_Install::PatchInfo#
4.2.3-CLL-FP02  1536006190.1.922#TMF_Install::PatchInfo#
4.2.3-INV-F2P1  1536006190.1.1068#TMF_Install::PatchInfo#
4.2.3-INV-FP02  1536006190.1.901#TMF_Install::PatchInfo#
4.2.3-INVGW-FP02        1536006190.1.921#TMF_Install::PatchInfo#
4.2.3-SWDGW-FP02        1536006190.1.897#TMF_Install::PatchInfo#
4.2.3-SWDJPS-FP02       1536006190.1.898#TMF_Install::PatchInfo#
4.2.3-SWDSRV-F2P1       1536006190.1.1062#TMF_Install::PatchInfo#
4.2.3-SWDSRV-FP02       1536006190.1.894#TMF_Install::PatchInfo#
4.2.3-TIV-CLL-FP0004    1536006190.1.1113#TMF_Install::PatchInfo#
4.2.3-TIV-INV-COMP-FP0004       1536006190.1.1109#TMF_Install::PatchInfo#
4.2.3-TIV-INV-FP0004    1536006190.1.1088#TMF_Install::PatchInfo#
4.2.3-TIV-INVGW-FP0004  1536006190.1.1108#TMF_Install::PatchInfo#
4.2.3-TIV-SWDGW-FP0004  1536006190.1.1118#TMF_Install::PatchInfo#
4.2.3-TIV-SWDSRV-FP0004 1536006190.1.1114#TMF_Install::PatchInfo#
```

Validating inventory scripts

After you confirm that Tivoli Configuration Manager V4.2.3 Fix Pack 04 is installed, you must confirm that the inventory schema scripts, which are included with Fix Pack 04, were executed.

You run the inventory schema scripts to:

- Create and update Tivoli Configuration Manager V4.2.3 infrastructure database tables.
- Work properly with Tivoli Provisioning Manager for Software.

You have two options for validating that the fix pack 04 inventory schema scripts were run:

- Use the Tivoli Management Framework Query Facility in the Tivoli Desktop. You can create a new Query to consult the current information about the SCHEMA_VERS table under the INVENTORY_QUERY Query_Library. Refer to *Tivoli Management Framework User's Guide V4.1.1,* GC32-0805 Chapter 7, if you are not familiar with Query creation.
- Perform a DB2 or Oracle SQL query on the inventory database.

In Example 5-3 we use a DB2 SQL query against the inv_db database.

Example 5-3 How to double check schema version running a query against the inv_db

```
Log on to the system and set the tivoli environment:
[root@milan][/]-> . /etc/Tivoli/setup_env.sh

Run the "wgetrim inv_query" command to find the information regarding
the RDBMS User and the Database ID:

[root@milan][/]-> wgetrim invdh_1
RIM Host:       milan
RDBMS User:     db2inst1
RDBMS Vendor:   DB2
Database ID:    inv_db
Database Home:  /home/db2inst1/sqllib
Server ID:      tcpip
Instance Home:  ~db2inst1
Instance Name:  db2inst1

You can now use these data to connect to the inv_db database after su
as db2inst1.

[db2inst1@milan][/home/db2inst1]-> db2 connect to inv_db

Database Connection Information

 Database server        = DB2/6000 8.2.4
 SQL authorization ID   = DB2INST1
 Local database alias   = INV_DB

[db2inst1@milan][/home/db2inst1]-> db2 "select * from schema_vers"

You get an output like the following:

[db2inst1@milan][/home/db2inst1]-> db2 "select * from schema_vers"
```

```
DATABASE_VERS                    SCHEMA_CHG_TIME            SCHEMA_CHG_TYPE
CHG_FILE_NAME                                                    CHG_DESC
-------------------------------- -------------------------- --------------------------------
CM 4.2.3                         2006-08-22-16.26.16.270186 INSTALL
inv_db2_schema.sql                                               New 4.2.3 installation, GA level
CM 4.2.3                         2006-08-30-18.47.34.452586 PATCH
inv_db2_schema_423_FP01.sql                                      FP01 applied to existing
installation
CM 4.2.3                         2006-08-30-18.54.00.410347 PATCH
inv_db2_schema_423_FP02.sql                                      FP02 applied to existing
installation
CM 4.2.3                         2006-08-30-18.54.51.554885 PATCH
h_inv_db2_schema_423_FP01.sql                                    FP01 applied to existing
installation, enable history
CM 4.2.3                         2006-08-30-18.55.00.333925 PATCH
h_inv_db2_schema_423_FP02.sql                                    FP02 applied to existing
installation, enable history
CM 4.2.3                         2007-04-25-17.14.54.336098 PATCH
inv_db2_schema_423_FP03.sql                                      FP03 applied to existing
installation
CM 4.2.3                         2007-04-25-17.15.06.479247 PATCH
h_inv_db2_schema_423_FP03.sql                                    FP03 applied to existing
installation, enable history
CM 4.2.3                         2007-04-25-17.15.15.418459 PATCH
inv_db2_schema_423_FP04.sql                                      FP04 applied to existing
installation
CM 4.2.3                         2007-04-25-17.15.22.596554 PATCH
h_inv_db2_schema_423_FP04.sql                                    FP04 applied to existing
installation, enable history

  9 record(s) selected.
```

5.2 Mapping to the Tivoli Management Framework

To manage the Tivoli Management Framework objects from the Tivoli Provisioning Manager for Software Web User Interface, you need to add the required Tivoli Management Framework infrastructure to Tivoli Provisioning Manager for Software. Then, replicate the existing Tivoli Management Framework objects into the Tivoli Provisioning Manager for Software DCM.

5.2.1 Adding the infrastructure

Using the Tivoli Provisioning Manager for Software Web User Interface, point to the hubs, spokes, and databases for your Tivoli Management Framework infrastructure.

Note: The minimum configuration required is a Tivoli Configuration Manager inventory database and a single hub.

Adding the hubs, spokes, and databases from the existing Tivoli Management Region Framework to the Tivoli Provisioning Manager for Software DCM creates, on-the-fly, the following XML files:

- *%TIO_HOME%*\config\tcm.xml on the Windows platforms
- *$TIO_HOME*/config/tcm.xml on the UNIX platforms

In the example described in this Redbook, we added a Tivoli Management Environment® infrastructure based on an AIX Tivoli Management Region server using an IBM DB2 relational database to store inventory and software distribution data.

Use the following instructions to add the Tivoli Management Framework infrastructure.

Adding an inventory database

The first step is to add a database definition on the Tivoli Provisioning Manager for Software DCM that points to the existing `inv_db` database on the Tivoli Management Environment infrastructure.

1. In the navigation pane, click **Inventory** → **Infrastructure Management** → **Tivoli Management Framework**. The Manage Tivoli Management Framework page is displayed, as shown in Figure 5-1.

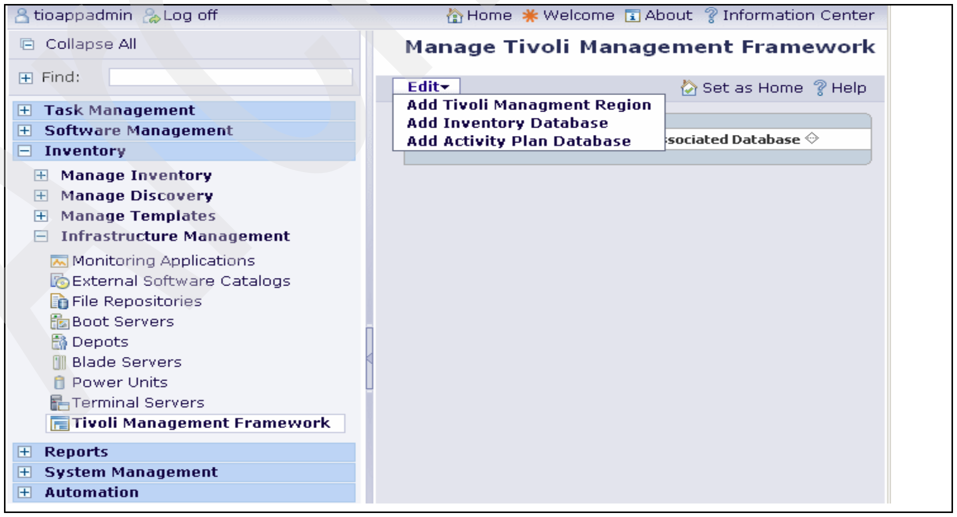
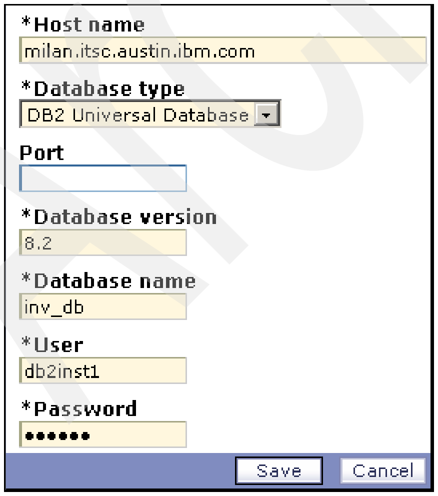


Figure 5-1 Edit the Manage Tivoli Management Framework dialog

Note: If your environment is on the Tivoli Provisioning Manager for Software V5.1 Fix Pack 1 level, then the Add Database Dialog contains different options, such as Add Hub and Add Spoke databases. We recommend that you install Tivoli Provisioning Manager for Software Fix Pack 2 to take advantage of improvements and new functionalities

2. Click **Edit** → **Add Inventory Database**. The Add Inventory Database dialog is displayed.
3. In the Add Inventory Database dialog, specify the following information:
   a. In the Host Name field, type the fully qualified host name of the database server.
   b. In the Database type list, select **DB2 Universal Database™**. If the database for your infrastructure is Oracle, select **Oracle Database**.
   c. The port field is optional and is in a different color. The default port IBM DB2 uses for a single instance database is 50000. If you want to know which port was used, refer to“Checking the default DB2 port” on page 106. For this database we did not specify any value.
   d. In the Database version field, type the database version, for example, 8.2.
   e. In the Database name field, type the inventory database name, for example `inv_db`.
   f. In the User and Password fields, type the database user name and password, and click **Save**.


*Host name
milan.itsc.austin.ibm.com
*Database type
DB2 Universal Database
Port

*Database version
8.2
*Database name
inv_db
*User
db2inst1
*Password
••••••

Figure 5-2 Add Inventory Database dialog

4. Repeat step 3., “In the Add Inventory Database dialog, specify the following information:” on page 105, including all of its substeps, for each Tivoli Management Region that includes a configured Tivoli Configuration Manager Inventory database.

Note: If the Inventory database is shared among different Tivoli Management Regions, then create one unique record for the Inventory database, and relate it to every associated Tivoli Management Region.

Checking the default DB2 port

If you do not specify a different port at installation time, the IBM DB2 uses the default of 50000.

If you have multiple instances on your IBM DB2 server, you must define different ports.

To check the port used during your IBM DB2 installation, you can refer to Example 5-4.

Example 5-4 Checking the default port that IBM DB2 used

```
Switch to your instance owner user, in our example db2inst1:

[root@milan][/]-> su - db2inst1

Run the below command lo locate the TCP/IP service name associated to
IBM DB2:

[db2inst1@milan][/home/db2inst1]-> db2 get dbm cfg | grep -i svc
 TCP/IP Service name                          (SVCENAME) =
db2c_db2inst1

Grep for this service name in the /etc/services file:

[root@milan][/etc]-> grep db2c_db2inst1 services
db2c_db2inst1   50000/tcp

The file on Windows Operating Systems is the following:

%SystemRoot%\system32\drivers\etc\services
```

Adding a Tivoli Management Region

The next step is to add a hub definition on the Tivoli Provisioning Manager for Software DCM that points to the existing hub on the Tivoli Management Framework infrastructure.

1. In the navigation pane, click **Inventory** → **Infrastructure Management** → **Tivoli Management Framework**. The Manage Tivoli Management Framework page is displayed.
2. Click **Edit** → **Add Tivoli Management Region**. The Add Tivoli Management Region dialog is displayed.
3. In the Add Tivoli Management Region, specify the following information:
   a. In the Host Name field, type the fully qualified host name of the Tivoli Management Region.
   b. In the User and Password fields, type the user name that has administrative rights to the Tivoli Management Region server, and then type the password.

      Note: The user name for the Tivoli Management Region is case-sensitive.

   c. Select the Associated database for the Tivoli Management Region, and click **Save**.

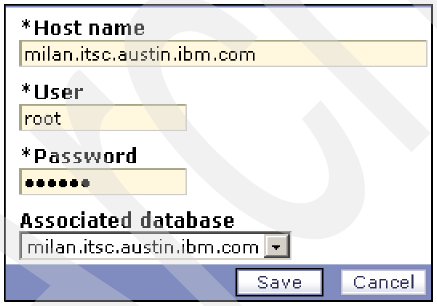
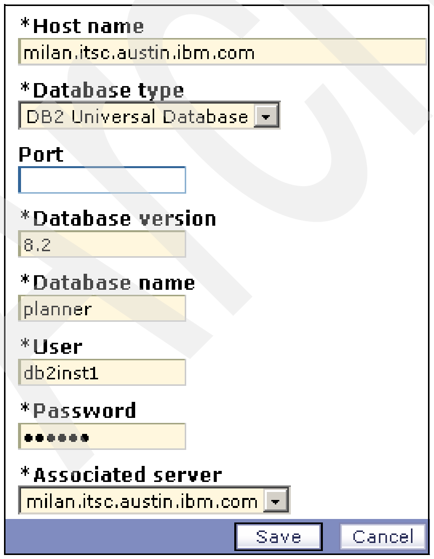

*Host name
milan.itsc.austin.ibm.com
*User
root
*Password
●●●●●●
Associated database
milan.itsc.austin.ibm.com

Figure 5-3 Add Tivoli Management Region dialog

Adding an activity plan database

The last step is to add an activity plan database definition on the Tivoli Provisioning Manager for Software DCM that points to the existing `planner` database on the Tivoli Management Environment infrastructure.

1. In the navigation pane, click **Inventory** → **Infrastructure Management** → **Tivoli Management Framework**. The Manage Tivoli Management Framework page is displayed.
2. Click **Edit** → **Add Activity Plan Database**. The Add Activity Plan Database dialog is displayed.
3. In the Add Activity Plan Database dialog, specify the following information:
   a. In the Host Name field, type the fully qualified host name of the database server.
   b. In the Database type list, select **DB2 Universal Database**. If your infrastructure's database is Oracle, then select **Oracle Database**.
   c. The port field is optional and in a different color. The default port IBM DB2 uses for a single instance database is 50000. If you want to know which port was used, refer to "Checking the default DB2 port" on page 106. For this database, we specified a value of 50000.
   d. In the Database version field, type the database version, for example, `8.2`.
   e. In the Database name field, type the inventory database name, for example `planner`.
   f. In the User and Password fields, type the database user name and password.
   g. Select the Associated server for the activity plan database, and click **Save**.


*Host name
milan.itsc.austin.ibm.com
*Database type
DB2 Universal Database
Port

*Database version
8.2
*Database name
planner
*User
db2inst1
*Password
●●●●●●
*Associated server
milan.itsc.austin.ibm.com

Figure 5-4 Add Activity Plan Database dialog

4. Repeat step 3 on page 108, and all of its substeps, for each Tivoli Management Region that includes a configured Tivoli Configuration Manager activity plan database.

Note: If the activity plan database is shared among different Tivoli Management Regions, create one unique record for the activity plan database, and relate it to every associated Tivoli Management Region.

5.2.2 The tcm.xml file

The actions from the previous sections result in the creation of an XML file called tcm.xml. This file is maintained in the following path on the Tivoli Provisioning Manager server:

- *$TIO_HOME*/config/tcm.xml on UNIX
- *%TIO_HOME%*\config\tcm.xml on Windows

Example 5-5 shows the file we used for our environment.

Example 5-5 Sample tcm.xml file

```
This file adds the following resources to the Tivoli Provisioning
Manager database:


1. TMR called milan.itsc.austin.ibm.com

2. Database pointing to a Tivoli Configuration Manager Inventory
   database called inv_db

3. Activity Plan Database pointing to a Tivoli Configuration Manager
   Activity Plan database called planner


<?xml version="1.0"?>
<config>
  <tmr>
    <server>milan.itsc.austin.ibm.com</server>
    <username>root</username>
    <password>cZyAQ8477wk=</password>
    <database>milan.itsc.austin.ibm.com</database>
  </tmr>
  <databases>
    <database>
      <server>milan.itsc.austin.ibm.com</server>
      <type>db2</type>
      <version>8.2</version>
      <name>inv_db</name>
      <username>db2inst1</username>
      <password>cZyAQ8477wk=</password>
    </database>
  </databases>
  <apmdatabases>
    <apmdatabase>
      <identifier>milan.itsc.austin.ibm.com</identifier>
      <server>milan.itsc.austin.ibm.com</server>
      <type>db2</type>
      <version>8.2</version>
      <name>planner</name>
      <port>50000</port>
      <username>db2inst1</username>
      <password>cZyAQ8477wk=</password>
    </apmdatabase>
  </apmdatabases>
</config>
```

If you need to add multiple Tivoli Management Region servers, but you do not want to repeat every step through the Web interface, you can add entries in the tcm.xml file (following the correct XML syntax according to the previous one). Then use the command **xmlimport** to import those entries into the DCM.

Example 5-6 XML import command

```
After you made the modifications to the file, follow these steps to
import it into the DCM database:


1. Login as tioadmin

2. Ensure the database is running

3. Open a command window and run the following command which enable you
   to import properly formed XML file to populate the DCM:
   $TIO_HOME/tools/xmlimport.sh file:$TIO_HOME/xml/tcm.xml for
   UNIX
   or
   %TIO_HOME%\tools\xmlimport.cmd file:%TIO_HOME%\xml\tcm.xml for
   Windows.

The DCM is now populated with the data related to your xml file.
```

5.3 Replicating the Tivoli Management Framework data

After mapping the Tivoli Management Framework databases to Tivoli Provisioning Manager for Software, as described in 5.2.1, “Adding the infrastructure” on page 103, you can proceed with the synchronization.

Tivoli Provisioning Manager for Software V5.1 provides two script files, tcmLink.cmd/sh and tcmReplicate.cmd/sh that you must run in the specified order to replicate the existing Tivoli Management Region Framework objects into the Tivoli Provisioning Manager for Software DCM.

5.3.1 The tcmLink script file

The tcmLink script file creates links between the databases and sets up a number of views on the Tivoli Provisioning Manager for Software server. You can regard these views as links between the Tivoli Management Region Framework and the DCM databases.

Use the following steps to run the tcmLink script:

1. Stop the Tivoli Provisioning Manager for Software server.
   - Allow two or three minutes before proceeding to the next step to ensure that any databases associated with Tivoli Provisioning Manager for Software server also stopped.
   - Make sure that no connections to the tc database are pending running the database command to list them.
     In our scenario we are using an IBM DB2 database that uses the **`db2 list application`** command to list the active connections to the database.
2. Ensure that the Tivoli Management Framework databases and the managed node process on the Tivoli Management Region servers (oserv) are running.
3. Open a command prompt (Windows) or a shell window (Oracle), and perform the following actions to create links between the Tivoli Management Framework and the Tivoli Provisioning Manager for Software databases:
   - Windows: Change the directory to*%TIO_HOME%*\tools, and run the **`tcmLink.cmd`** script file.
   - UNIX: Change the directory to *$TIO_HOME/*tools, and run the **`tcmLink.sh`** script file.

   Until tcmLink finishes successfully, do not attempt replication.

4. Restart the Tivoli Provisioning Manager for Software server.

   For troubleshooting, a `tcmLink_timestamp.log` file is created in the *%TIO_LOGS%* directory. The default value for *`%TIO_LOGS%`* is C:\Program Files\ibm\tivoli\common\COP\logs.

   In case you are running your Tivoli Provisioning Manager server on a UNIX machine, the script to use is called **`tcmLink.sh`** and the *$TIO_LOGS* is defaulted to /usr/ibm/tivoli/common/COP/logs.

   Note: If the script ends with errors, the database part of the bridge is not created. In this case you need to analyze the log files before attempting to execute the script again. You can also refer to 11.4.1, “Troubleshooting tcmLink” on page 369 for more information.

   Note: After you upgrade to Tivoli Provisioning Manager for Software version 5.1 Fix Pack 2, and before you perform any coexistence operations, rerun the tcmLink script file to pass to the Tivoli Provisioning Manager for Software database the new Inventory database tables and views, which are related to the patch management functionality and required to successfully run any operation.

The tcmLink execution details for an IBM DB2 database

Following are the main operations that the tcmLink script executes for an IBM DB2 database:

- Catalogs a local node called tcm_1:

  ```
  catalog tcpip node tcm_1 remote MILAN.ITSC.AUSTIN.IBM.COM server
  50000;
  ```

- Catalogs a local database with the alias name TCM1 that points to the `inv_db` database on the Tivoli Management Region server:

  ```
  catalog database inv_db as TCM1 at node tcm_1;
  ```

- Connects to local tc database on the Tivoli Provisioning Manager server:

  ```
  Database server        = DB2/NT 8.2.4
  SQL authorization ID   = DB2INST1
  Local database alias   = TC;
  ```

- Enables the federation to the database manager:

  ```
  UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES;
  ```

- Stops and starts the database for the configuration to take effect.

- Connects to the remote Tivoli Management Region database using the new TCM1 alias:

```
Database server        = DB2/6000 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TCM1;
```

- Creates the following views:

  HARDWARE_VIEW
  STORAGE_DEVICE_VIEW
  CPU_VIEW;

- Connects to the local tc database on the Tivoli Provisioning Manager server:

```
Database server        = DB2/NT 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TC;
```

- Creates the following wrapper:

  TCM_WRAPPER1;

- Creates the following nicknames:

```
db2inst1.COMPUTER_VIEW1
db2inst1.COMPUTER_MEM_VIEW1
db2inst1.CPU_VIEW1
db2inst1.HARDWARE_VIEW1
db2inst1.INVENTORY_SIG1
db2inst1.MEM_MODULES_VIEW1
db2inst1.MODEM_VIEW1
db2inst1.MOUSE_VIEW1
db2inst1.PARTITION_VIEW1
db2inst1.PRINTER_VIEW1
db2inst1.SD_INST1
db2inst1.SD_PACKAGES1
db2inst1.SIGNATURE1
db2inst1.SMBIOS_DATA_VIEW1
db2inst1.STORAGE_DEVICE_VIEW1
db2inst1.USB_DEV_VIEW1
db2inst1.VID_CARD_VIEW1;
```

- Connects to the local tc database on the Tivoli Provisioning Manager server:

```
Database server        = DB2/NT 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TC;
```

- Creates a view for all the nicknames with the following SQL commands for each view:

```
create COMPUTER_VIEW and return to COMPUTER_VIEW
CREATE VIEW COMPUTER_VIEW as SELECT * from COMPUTER_VIEW1;
```

- Catalogs a local node called apm_1:

```
catalog tcpip node apm_1 remote MILAN.ITSC.AUSTIN.IBM.COM server
50000;
```

- Catalogs a local database with the alias name APM1 that points to the planner database on the Tivoli Management Region server:

```
catalog database planner as APM1 at node apm_1
```

- Connects to the local tc database on the Tivoli Provisioning Manager server:

```
Database server        = DB2/NT 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TC;
```

- Enables the federation to the database manager:

```
UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES;
```

- Stops and starts the database for the configuration to take effect.
- Connects to the remote Tivoli Management Region database using the new APM1 alias:

```
Database server        = DB2/6000 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = APM1;
```

- Creates the following view:

```
TCM_ACTIVITY_PLAN_VIEW;
```

- Connects to the local tc database on the Tivoli Provisioning Manager server:

```
Database server        = DB2/NT 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TC;
```

- Creates the following wrapper:

```
APM_WRAPPER1;
```

- Creates the following nicknames:

```
db2inst1.TCM_ACTIVITY_PLAN_VIEW1
db2inst1.TCM_ACTIVITY1
db2inst1.TCM_ACT_TARGET_SPEC1
db2inst1.TCM_ACTIVITY_TARGET1
db2inst1.TCM_PLAN_TARGET_SPEC1
```

```
db2inst1.TCM_PLAN_TARGET1
db2inst1.TCM_EXECUTION_WINDOW1
db2inst1.TCM_ACT_PARAMETER1
db2inst1.TCM_VARIABLE1;
```

- Connects to local tc database on the Tivoli Provisioning Manager server:

```
Database server        = DB2/NT 8.2.4
SQL authorization ID   = DB2INST1
Local database alias   = TC;
```

- Creates a view for all the nicknames with the following SQL commands for each view:

```
Create view TCM_ACTIVITY_PLAN_VIEW and return to
TCM_ACTIVITY_PLAN_VIEW
CREATE VIEW TCM_ACTIVITY_PLAN_VIEW as SELECT * from
TCM_ACTIVITY_PLAN_VIEW1
```

5.3.2 The tcmReplicate script file

The **`tcmReplicate`** script file populates the Tivoli Provisioning Manager for Software DCM using the links that running the tcmLink.cmd/sh script created. You can perform this task using the Web interface, the **`tcmReplicate`** command, or the **`tcmReplicate.cmd/sh`** script.

Summary of the replication settings

Table 5-2 on page 116 provides the replication settings you can specify to run a partial or a full replication from the Web User Interface or from the command line.

[1] On the Discovery parameters list, for the Custom Software Signature replication, you can set only one of the following values: All or None

Table 5-2 Replication settings summary

Replicating from	Partial replication	Full replication	No replication
Web User Interface	For each replicating type set one of the following values: ► All [1] ► New ► Existing ► None	Set all replication types to All	Set all replication types to None
Command line (**`tcmReplicate.cmd`** or **`tcmReplicate.sh`**)	For each replication type, specify one of the following values: ► NEW ► ALL ► The names of the item to be replicated: name# and region	No parameters	Do not run **`tcmReplicate.cmd/.sh`**

Running replication using the tcmReplicate script

Use the following steps to run the tcmreplicate.cmd/sh script:

1. Open a command prompt (Windows) or a shell window (UNIX).
2. Open a command prompt (Windows) or a shell window (Oracle), and perform the following actions to populate the Tivoli Provisioning Manager for Software DCM database with the Tivoli Management Framework data:
   - Windows: Change the directory to *%TIO_HOME%*\tools, and run the **`tcmReplicate.cmd`** script file.
   - UNIX: Change the directory to *$TIO_HOME/*tools, and run the **`tcmReplicate.sh`** script file.

Use the following command to get help with replication switches.

Select the execution directory:

```
cd %TIO_HOME%\tools:
tcmReplicate -h
```

The `tcmReplicate -h` script produces the following output:

[1] On the Discovery parameters list, for the Custom Software Signature replication, you can set only one of the following values: All or None

```
COPTMF007E Usage: tcmReplicate [parameters] where parameters are:

[-a TRUE]|[-i TRUE|FALSE]|[ -<option>
[ALL|NEW|[<name>#<region>[,<name>#<region>]*]]
```

In the displayed output, *option* can be any of the following:

-p Activity Plans
-e Endpoints
-g Groups: Profile Managers, Policy Regions, Task Library
-s Software Packages
-r File Repositories
-d Inventory Profiles
-m Managed Nodes
-w Gateways
-h Help

Consider the following examples for replication:

- Full environment replication

  ```
  tcmReplicate.cmd
  ```

- New endpoints replication

  ```
  tcmReplicate.cmd –e NEW
  ```

- Replicate PolicyRegion named “sample-pr” in TMR1 having top region named “top-tmr1”

  ```
  tcmReplicate.cmd –g sample-pr#top-tmr1
  ```

- Replicate all gateways

  ```
  tcmReplicate.cmd –w ALL
  ```

Important: Before running **`tcm replicate`**, execute a **`wchkdb -ux`** to make sure that the objects in the Tivoli Management Framework are up-to-date according to the last modifications.

Running replication using the Web interface

You can also run replication using through Web interface using a provided discovery configuration called Tivoli Configuration Manager Discovery.

Use the following steps to execute the replication using the Tivoli Configuration Manager Discovery:

1. Log on to the Tivoli Provisioning Manager for Software Web User Interface. The default user ID and the default password are both tioappadmin.
2. In the navigation pane, expand Inventory.

3. Click **Manage Discovery** → **Discovery Configurations**.
   The Configure Discovery window is displayed.
4. Click **Edit** → **Add Discovery Configurations**. The Discovery window is displayed.
5. Enter the name and the description of the discovery.
   a. Under Discovery, select Other as the type of data to discover.
   b. Under Discovery Method, select Tivoli Configuration Manager Discovery, and click **Next**. The Discovery Parameters window is displayed.
6. You can specify a full or partial replication by choosing to replicate all or some of the predefined types; however, the first time you replicate, you must perform a full replication by accepting the default values.

   For each type, the selection for replication can be:

   – All

     Replicate all new and existing items in the DCM. Choose this value the first time you replicate all of the Tivoli Management Framework objects into the Tivoli Provisioning Manager for Software DCM database.

   – None

     Do not replicate any items into the DCM.

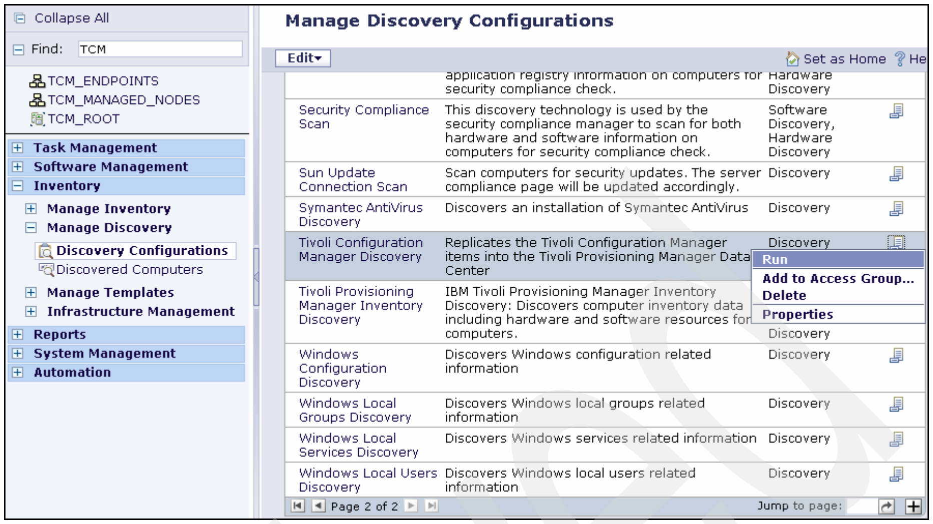
   – New

     Replicate only the new items that are not currently defined in the DCM.

   – Existing

     Allow partial replication of existing items defined in the DCM. When you select Existing, you have the possibility to choose which existing items to replicate from a list.

   Click **Finish**.
7. On the Manage Discovery Configurations page, select the Tivoli Configuration Manager Discovery Configuration, right-click, and then click **Run**. The Initiate Discovery page is displayed, as shown in Figure 5-6 on page 119.


Collapse All

Find: TCM

- TCM_ENDPOINTS
- TCM_MANAGED_NODES
- TCM_ROOT

- Task Management
- Software Management
- Inventory
  - Manage Inventory
  - Manage Discovery
    - Discovery Configurations
    - Discovered Computers
  - Manage Templates
  - Infrastructure Management
- Reports
- System Management
- Automation

Manage Discovery Configurations

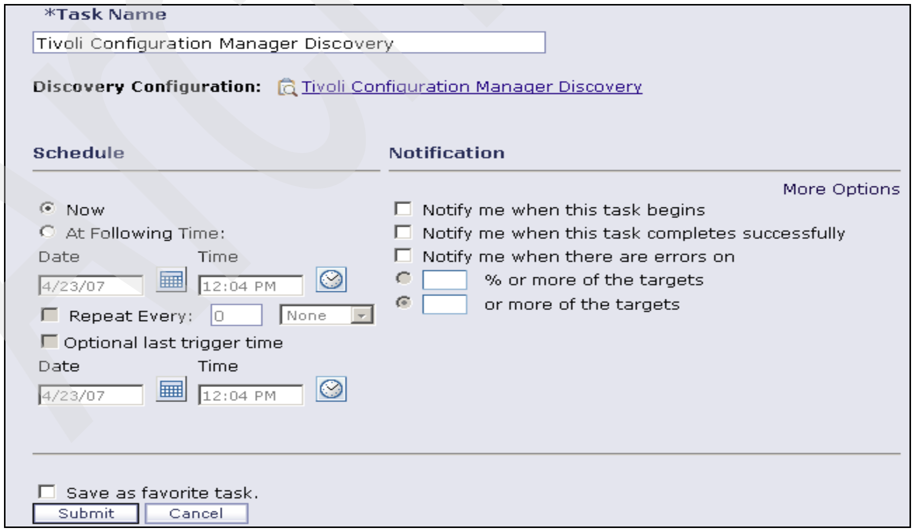
Edit▾ Set as Home ? He

	application registry information on computers for security compliance check.	Hardware Discovery
Security Compliance Scan	This discovery technology is used by the security compliance manager to scan for both hardware and software information on computers for security compliance check.	Software Discovery, Hardware Discovery
Sun Update Connection Scan	Scan computers for security updates. The server compliance page will be updated accordingly.	Discovery
Symantec AntiVirus Discovery	Discovers an installation of Symantec AntiVirus	Discovery
Tivoli Configuration Manager Discovery	Replicates the Tivoli Configuration Manager items into the Tivoli Provisioning Manager Data Center	Discovery
Tivoli Provisioning Manager Inventory Discovery	IBM Tivoli Provisioning Manager Inventory Discovery: Discovers computer inventory data including hardware and software resources for computers.	Discovery
Windows Configuration Discovery	Discovers Windows configuration related information	Discovery
Windows Local Groups Discovery	Discovers Windows local groups related information	Discovery
Windows Local Services Discovery	Discovers Windows services related information	Discovery
Windows Local Users Discovery	Discovers Windows local users related information	Discovery

Page 2 of 2 Jump to page:

Figure 5-5   Manage Discovery Configurations page

8. On the Initiate Discovery page, perform the following actions:
   a. Enter a Task Name or accept the default value.
   b. Under Schedule, select **Now** to run the discovery task immediately, or schedule it to run at a specified date and time.
   c. Select the appropriate notification options, and click **Submit**.


*Task Name: Tivoli Configuration Manager Discovery

Discovery Configuration: Tivoli Configuration Manager Discovery

Schedule

[✓ Now]
At Following Time:
Date 4/23/07 Time 12:04 PM
Repeat Every: 0 None
Optional last trigger time
Date 4/23/07 Time 12:04 PM

Notification

More Options
Notify me when this task begins
Notify me when this task completes successfully
Notify me when there are errors on
% or more of the targets
or more of the targets

Save as favorite task.

Figure 5-6   Example of a Tivoli Configuration Manager Discovery dialog.

The tcmReplicate execution details

A full replication migrates the following Tivoli Management Framework data into the Tivoli Provisioning Manager for Software DCM:

- All of the target computers, source hosts, managed nodes, gateways, profile managers, policy regions, queries, query libraries, task libraries, and tasks that are defined in the Tivoli Management Framework.

  Note: The root path of the replicated source host becomes empty and is interpreted as */* on UNIX systems and *<drive:>* on Windows systems. You cannot modify it.

- All of the SoftwarePackage profiles and software package blocks that are defined in the Tivoli Management Framework, which are converted to Software Products in Tivoli Provisioning Manager for Software. However, the nesting of software packages is not applicable.
- All of the InventoryConfig profiles that are defined in the Tivoli Management Framework, which are converted to Discovery Configurations in Tivoli Provisioning Manager for Software.
- All of the activity plans that are defined in the Tivoli Management Framework environment using the old Activity Plan Editor, which are converted to new activity plans in Tivoli Provisioning Manager for Software.
- All of the task libraries and tasks that are defined in the Tivoli Management Framework.
- All of the queries that are defined in the Tivoli Management Framework, which are converted to new reports in Tivoli Provisioning Manager for Software.

During the replication, a TCM_ROOT group is created that contains all of the group members that you replicated.

Note: Only the resources are replicated, not the scheduled procedures, such as inventory scans or scheduled tasks. You must manually redefine the inventory scans or scheduled tasks after the replication. We recommend that you do not replicate Tivoli Management Framework data until you have no pending scheduled activities.

5.4 Data mapping behind the synchronization scenes

This section provides details and internal information about the components and the flow of the integration with the Tivoli Management Framework.

5.4.1 Database synchronization process

In a medium-large business enterprise managed by Tivoli Configuration Manager over Tivoli Management Framework, the number of endpoints can reach tens of thousands. Management operations over such enterprises are made of hundreds of software distribution profiles, inventory data retrieval profiles, and general purposes task execution.

With the objective of simplifying and helping the administrator to get this huge number of resources under control, Tivoli Management Framework allows you to organize the resources into hierarchical groups. You can logically organize the enterprise into policy regions that contain several profile managers, where each policy region addresses software distribution and inventory profiles over a list of target endpoints. Additionally, this organization can belong to a multi Tivoli Management Region Servers environment that is structured in a typical hub-spoke architecture.

In order for Tivoli Provisioning Manager for Software to coexist with an environment where Tivoli Configuration Manager is deployed, a subset of the resources handled by Tivoli Configuration Manager is required to be available from the Tivoli Provisioning Manager for Software server to allow the enterprise system operator to perform management operations on these resources directly from the Tivoli Provisioning Manager for Software Web User Interface. A replication strategy is implemented to make those resources available in Tivoli Provisioning Manager for Software DCM.

Coexistence databases architecture

The synchronization process analyzes several types of data from the following databases:

- Inventory
- Activity plan
- Tivoli Management Framework object database

Figure 5-7 shows the connections between the Tivoli Provisioning Manager for Software and the Tivoli Management Framework environment.

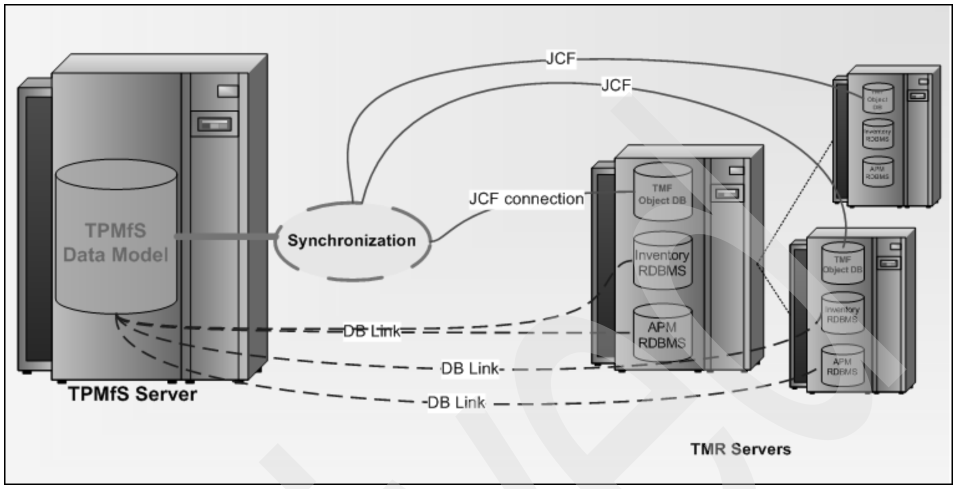


Figure 5-7 Connections generic architecture

Typical Tivoli Management Framework resources that belong to each Tivoli Management Region, such as endpoints, profile managers, profiles, policy regions, queries, task libraries, and tasks along with their relationships, are stored into the object database that is registered on that Tivoli Management Region. Accessing these resources from outside of the Tivoli Management Framework environment is only possible using the Java™ Client Framework (JCF), which is a kind of Java development kit that the Tivoli Management Framework ships exactly for these purposes. After the right credentials are used by an external process to create a connection to a given Tivoli Management Region through the Java Client Framework, that process can perform any kind of management operations for which the used credential is authorized on that Tivoli Management Region.

In Tivoli Provisioning Manager for Software, a process called **`tcmReplicate`** is intended to perform the database synchronization, and in this context it uses JCF connections (one connection for each Tivoli Management Region required to be migrated) to retrieve the information stored inside the Object Databases.

Database link

From a Tivoli Configuration Manager standpoint, other storage systems are used to contain data specific to inventory and configuration management. In other words, Tivoli Configuration Manager-specific data along with their relationships with Tivoli Management Framework resources are stored in Relational Data Base Management Systems (RDBMS) connected to the Tivoli Management

Framework environment. Each Tivoli Management Region server holds its own inventory database, even though in particular configurations the same inventory database can be shared between several Tivoli Management Region servers. In this case, all of the Tivoli Management Region spoke servers store the data on a central database connected to the hub Tivoli Management Region server.

In the same way, the Activity Plan Manager (APM) component needs to have its own database too, and, in general, any Tivoli Management Region server can own one APM database.

A lot of information that belongs to both Tivoli Configuration Manager databases (Inventory and APM) gets retrieved by the **tcmReplicate** process during database synchronization. This implies that additional connections to the databases in the Tivoli Configuration Manager world need to be maintained by Tivoli Provisioning Manager for Software in order for the replication process to happen.

To activate these connections, you must establish a link between each Tivoli Configuration Manager database and the database used by Tivoli Provisioning Manager for Software as depicted in Figure 5-8 on page 124.

The database link (also known as federation) is a technique where a remote and catalogued database, inside a wrapper, are created on the local instance of another database. Inside this wrapper, a set of nicknames is used to emulate Tables that belong to the remote database. In this way, whatever process (either CLI or running inside an application) can connect to the local instance of the database and retrieve data from a nickname even though the physical data is stored in the related linked table on the remote database.

In the context of Tivoli Provisioning Manager for Software, after the federation occurs, the replication process (**tcmReplicate**) needs only the Tivoli Provisioning Manager for Software database connection credentials to access data belonging to any linked database on the Tivoli Configuration Manager side.

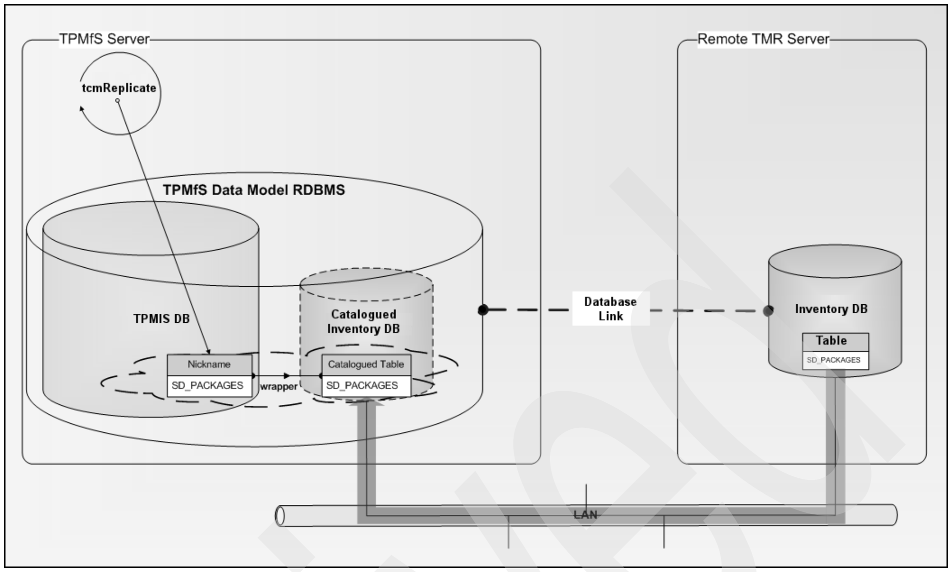


Figure 5-8 Database link diagram

Database replication

In the previous sections, we discussed the objects that are replicated in a coexistence environment. We can now go into the details of what objects these replicated objects are mapped to in Tivoli Provisioning Manager for Software DCM.

Considering that Tivoli Provisioning Manager for Software is intended to manage configuration and provisioning of heterogeneous environments (for example data centers, enterprises with servers and workstations, down to single mobile computers) we need to standardize its DCM to address this heterogeneity. From this point-of-view, Tivoli Management Framework concepts that are designed to address logical grouping and hierarchical organizations, such as policy regions and profile managers, do not have a specific mapping on the new Tivoli Provisioning Manager for Software; therefore, any kind of Tivoli Management Framework grouping is mapped into the group generic object.

Tivoli Provisioning Manager for Software groups can be:

- Typed

  They can contain only objects of the specified type, for example, computer, software product, Discovery Configuration, and so on.

- Untyped

  They can contain any kind of objects, regardless of their type.

The task of reporting, inside Tivoli Provisioning Manager for Software DCM, one single resource along with its relations to other objects stored in Tivoli Management Framework and Tivoli Configuration Manager databases is a large time consuming task. The overall process can take a large time frame to complete depending on how many objects and how many relations they have inside the original enterprise.

The Tivoli management agent endpoint is the resource that requires the highest number of database accesses in read-mode from the Tivoli Management Framework and Tivoli Configuration Manager side and in write-mode on the Tivoli Provisioning Manager for Software side. Figure 5-9 shows the connections that are maintained for any single-migrated endpoint.

Considering the number of groups that are created or updated after an endpoint is migrated, it is easy to understand that the `tcmReplicate` process is time consuming, and its duration is strictly related to the number of endpoints that belong to the Tivoli Management Framework enterprise.

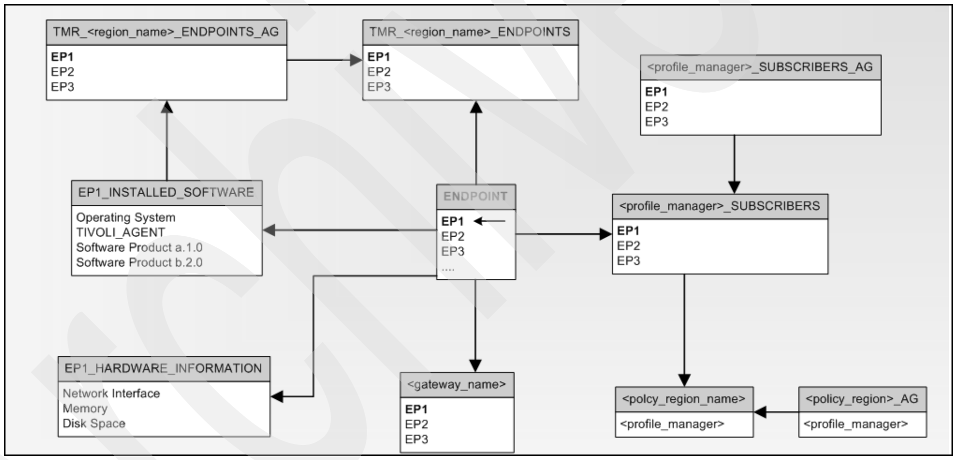


Figure 5-9 Connections between endpoint and other resources inside Tivoli Provisioning Manager for Software DCM

Best practices

For the previously mentioned considerations about the **`tcmReplicate`** process duration, it is possible to smoothly synchronize the Tivoli Management Framework and the Tivoli Provisioning Manager for Software environments using one of the following methods:

- Overnight synchronization

  To avoid the initial replication process, you can schedule the synchronization by night to avoid a slow down of any Tivoli Provisioning Manager for Software business related operations.

- Selective replication

  The **`tcmReplicate`** process allows you to select a subset of resource types or resource groups to be replicated at a time.

5.5 Data mapping between Tivoli Provisioning Manager for Software and Tivoli Configuration Manager

Table 5-3 through Table 5-7 on page 130 describe commonly used objects and resources and their location in Tivoli Configuration Manager with the Tivoli Provisioning Manager for Software equivalent and its new location.

Table 5-3 General objects

Object in Tivoli Configuration Manager terminology	**Current location**	**Object in Tivoli Provisioning Manager for Software terminology**	**New location**
Software Distribution Profile	Tivoli Object Database	Software Definition The definition of an application in terms of generic properties (name, version), also known as Software Module.	Data model (a.k.a. DCM) This name identifies the Tivoli Provisioning Manager for Software relational database, where every Tivoli Provisioning Manager for Software resource is defined.

Object in Tivoli Configuration Manager terminology	Current location	Object in Tivoli Provisioning Manager for Software terminology	New location
Software Package Block (SPB)	Tivoli Configuration Manager Source Host	Software installable The installable (binary) file associated to a software definition, also known as Software Product. A Software Definition might have more than one installable.	File Repository (FR) Device dedicated to store files. A valid FR must have a root path and a valid IP configuration. Each file has an associated file repository and its path is related to the FR root path.
Tivoli Management Framework task	Tivoli Object Database	Endpoint task New concept introduced in Tivoli Provisioning Manager for Software to provide the ability to execute a command on a remote machine leveraging the infrastructure capabilities.	Data model
Tivoli Management Framework queries	Tivoli Object Database	Reports or Dynamic groups A report is an object used to perform queries on the DCM contents. It is a wrapper of a SQL statement and can be used to define dynamic groups, which are groups whose members are defined by the execution of a query on the database.	Data model
Inventory profile	Tivoli Object Database	Discovery Configuration An instance of a configuration for a specific discovery method (device model). Tivoli Provisioning Manager for Software release provides two discovery methods: the one based on the Common Inventory Technology (CIT) v2.2 and the one based on the Security Compliance Manager (SCM) for compliance checks (desired state management).	Data model
Activity plan	APM RDBMS	Activity plan	Data model

Table 5-4 Hardware

Object in Tivoli Configuration Manager terminology	Current location	Object in Tivoli Provisioning Manager for Software terminology	New location
Inventory data (Hardware and software discovered information)	Tivoli Config-uration Manager RDBMS	Properties of computer Any software installed on a computer is mapped into the concept of Software Installation, which represents the association between a Software Installable and a device. The hardware properties discovered during a scan are mapped into the computer's properties.	Data model

Table 5-5 Logical grouping

Object in Tivoli Configuration Manager terminology	Current location	Object in Tivoli Provisioning Manager for Software terminology	New location
Policy region	Tivoli Object Database	Untyped group A group is a DCM object that represents a collection of resources. It can be either typed (every member having the same type) or untyped (members having mixed types).	Data model
Profile manager	Tivoli Object Database	Untyped group	Data model
Task library	Tivoli Object Database	End Point Task group	Data model

Table 5-6 Topology

Object in Tivoli Configuration Manager terminology	Current location	Object in Tivoli Provisioning Manager for Software terminology	New location
Tivoli Management Region	Tivoli Object Database	Computer A computer is a DCM object that represents a system in the enterprise, regardless of its role (server, endpoint).	Data model
Managed node	Tivoli Object Database	Computer	Data model
SourceHost	Tivoli Object Database	File Repository	Data model
Gateway	Tivoli Object Database	Computer group A computer group is a typed collection of computers.	Data model
Repeater	Tivoli Object Database	Tivoli Configuration Manager Depot A Tivoli Configuration Manager Depot is a DCM object that represents a system where software resources are loaded to allow a faster deployment of such resources. It is the only allowed target of a publish operation and is used to represent Tivoli Management Framework repeaters as well as CDS depots.	Data model
Endpoint (TMA object)	Tivoli Object Database	Computer	Data model

Table 5-7 Security

Object in Tivoli Configuration Manager terminology	Current location	Object in Tivoli Provisioning Manager for Software terminology	New location
Administrator	Tivoli Object Database	User Group Collections of users that can be used to assign the same level of security permissions	Data model
Login (OS users)	Tivoli Object Database	LDAP users A user's profile is defined for each of them, which includes: logon information, personal information, security role, access permission groups to which the user belongs, user's password.	LDAP server
Policy Region	Tivoli Object Database	Access Group Logical organizations of DCM Objects over which a user is granted access.	Data model
Roles	Tivoli Object Database	Access permission Group Set of access privileges that identify the security requirements needed to perform Logical Device Operations. A Logical Device Operation (LDO) is an abstraction of a real operation, which can be implemented in different ways depending on the involved resources.	Data model

5.6 Configuring user security settings

After migrating the existing Tivoli Management Framework data into the Tivoli Provisioning Manager for Software DCM database, you can migrate the existing Tivoli Management Framework users into Tivoli Provisioning Manager for Software.

Based on the assumption that you may choose to configure Tivoli Provisioning Manager for Software with a read-only replica of the LDAP server, migrating the existing Tivoli Management Framework data inserts the migrated resources only into the Tivoli Provisioning Manager for Software DCM database. After you migrate the existing Tivoli Management Framework data, there are additional steps you must perform to complete the migration process.

The tmrname.xml file

After you populate the Tivoli Provisioning Manager for Software DCM database with Tivoli Management Framework data, a file named *tmrname.xml* is created on the Tivoli Provisioning Manager for Software for each Tivoli region, where tmrname is the variable associated to the Tivoli region name.

Example 5-7 shows the root administrator for the Tivoli region Root_milan-region.

Example 5-7 Sample milan.xml file

```
The hostname of the Tivoli Management Region server used for this
chapter is milan.
Follow the content of the milan.xml file:

<Tmr>
  <Administrators>
    <Administrator name="Root_milan-region">
      <Login name="root@milan" />
      <Login name="Administrator" />
      <Policy-Region name="global">
        <Role name="backup" />
        <Role name="restore" />
        <Role name="Query_view" />
        <Role name="Query_execute" />
        <Role name="Query_edit" />
        <Role name="RIM_view" />
        <Role name="RIM_update" />
        <Role name="HTTP_Control" />
        <Role name="Dist_control" />
        <Role name="Inventory_view" />
        <Role name="Inventory_scan" />
        <Role name="Inventory_edit" />
        <Role name="Inventory_end_user" />
        <Role name="CCM_Manage" />
        <Role name="CCM_Edit" />
        <Role name="CCM_Admin" />
        <Role name="CCM_View" />
        <Role name="APM_Manage" />
        <Role name="APM_Edit" />
        <Role name="APM_Admin" />
        <Role name="APM_View" />
        <Role name="policy" />
        <Role name="install_product" />
        <Role name="install_client" />
        <Role name="user" />
        <Role name="admin" />
        <Role name="senior" />
        <Role name="super" />
        <Role name="set_role_role" />
      </Policy-Region>
      <Policy-Region name="security_group_any_admin">
        <Role name="user" />
      </Policy-Region>
      <Policy-Region name="Root_milan-region#milan-region">
        <Role name="admin" />
        <Role name="user" />
        <Role name="rconnect" />
      </Policy-Region>
    </Administrator>
  </Administrators>
  <InterRegions>
    <LocalRegion HostName="milan" />
  </InterRegions>
</Tmr>
```

On the tmrname.xml file, for each user the following data is reported:

- The login names that identify an operating system user
- The policy region name
- The related Tivoli Management Framework security role for each policy region

To migrate the existing users' data into Tivoli Provisioning Manager for Software, perform the following steps:

1. Review the information contained in the tmrname.xml file.
2. Choose which users you intend to keep, and delete the other login names from the file.

   Note: Remove the login names that have the following naming convention: hostname\username.

3. Create the users on the LDAP server. The following options are available:

   a. Use any LDAP tool.

   b. Use the Tivoli Provisioning Manager for Software Web User Interface, if Tivoli Provisioning Manager for Software is configured with write permissions on the LDAP server.

      For more details on how to perform this, select **System Management** → **Manage Users** → **Edit** → **Add User** action from the Web User Interface, and refer to the Tivoli Provisioning Manager for Software Information Center.

   c. Create the admin.ldif file, and customize its content by using the Example 5-8 as a reference.

      The mandatory fields in this file are:

      - Specify the fn, cn, roleA, and userPassword values for each Login name in the tmrname.xml file.
      - The roleA keyword must contain a valid Tivoli Provisioning Manager for Software role.

   d. After you customize the .ldif file, from the Tivoli Provisioning Manager for Software server command line, run the following command to create the LDAP users:

      ```
      ldapmodify -a -D cn=root -w pass -h hostname -i admin.ldif
      ```

      In the command, note the following explanation:

      - *pass* is the password of the LDAP user.
      - *hostname* is the host name of the LDAP server.
      - *admin.ldif* is the name of the .ldif file you created.

Example 5-8 admin.ldif example file

```
dn: cn=tioappadmin, dc=ibm,dc=com
pwdMustChange: false
sn: tioappadmin
roleA: SuperUser
roleA: SystemAdministrator
userPassword:: e1NIQX1GbElwQWdoS1NGclZ4UnlnalZOQWVueGhQVDg9
businessTelephoneNumber: (555)555-5555
notificationMethod: SMTP
mail: user@company
objectClass: thinkControlUser
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

```
postalAddress: 1234 Street
fn: tioappadmin
cn: tioappadmin
locale:en
```

4. From the Tivoli Provisioning Manager for Software Web User Interface perform the following steps:
   a. Click **System Management**.
   b. Click **Manage Users**. A list of all of the LDAP users is displayed.
   c. Verify that the users displayed in the list are the same users defined on the tmrname.xml file.
5. From the Tivoli Provisioning Manager for Software server command line, run the following command to associate to each LDAP user the appropriate Access Permission Group, which is based on the information contained in the tmrname.xml file `tcmreplicate -a true`.

Customizing access permissions

When an IBM Tivoli Management Framework user is migrated to a Tivoli Provisioning Manager for Software user, the user is associated to a default access group. The objects related to this user are organized in this default access group.

An access permission group is also automatically assigned to that default access group, which contains the individual permissions that apply to individual objects. Depending on the access permission group, the user can perform particular actions on a group of resources.

To keep the same IBM Tivoli Management Framework permissions for a migrated user, you must either change the default access group associated to the user or define a new access permission group for the default access group.

Roles

Table 5-8 provides a mapping between the security roles of Tivoli Configuration Manager and Tivoli Provisioning Manager for Software.

Table 5-8 Security roles mapping

Tivoli Configuration Manager roles	Tivoli Provisioning Manager for Software roles	Tivoli Provisioning Manager for Software tasks
admin	Configuration Operator	The user cannot create or edit DCM objects. They can only work with existing objects.

Tivoli Configuration Manager roles	Tivoli Provisioning Manager for Software roles	Tivoli Provisioning Manager for Software tasks
super, senior	Configuration Administrator	The user can work with existing objects. The user can create DCM objects in the DefaultAccessGroup.
root, super	System Administrator	The user can work with existing objects. The user can create DCM objects in the DefaultAccessGroup. The user can place the objects in any existing AccessGroup.

If you are using Tivoli Provisioning Manager for Software roles, such as Inventory Specialist or Software Operator, you can perform only specific discovery operations or software management operations.

5.7 Report Manager

You install the Report Manager component on the Tivoli Management Region server using *either* of the following methods:

- Install the following interim fixes for the Tivoli Configuration Manager Software Distribution server component:
  - Software Distribution Interim Fix 4.2.3.2-TIV-SWDSRV-IF0002
  - Inventory Interim Fix 4.2.3.2-TIV-INV-IF0002
- Install Tivoli Configuration Manager 4.2.3 Fix Pack 4 (that also includes the Interim Fixes mentioned in the previous bullet).

Report Manager acts as a report consumer for the Inventory and Software Distribution operations. Its behavior is similar to the old Activity Planner Engine, where the notification is sent at the end of the operation.

The Report Manager is responsible for:

- Managing reports coming from Tivoli Configuration Manager, and providing a querying interface to the Tivoli Provisioning Manager for Software server.
- Storing reports coming from the underlying applications into the inventory database inv_db.

The Tivoli Provisioning Manager for Software reporting engine queries the Report Manager at regular intervals to retrieve the results of completed actions, and then stores the results into the DCM repository.

Two separate threads are used to manage software related reports and discovery related reports. The discovery thread triggers endpoint selective replication.

Report Manager internals

Figure 5-10 shows an internal representation of the Report Manager component.

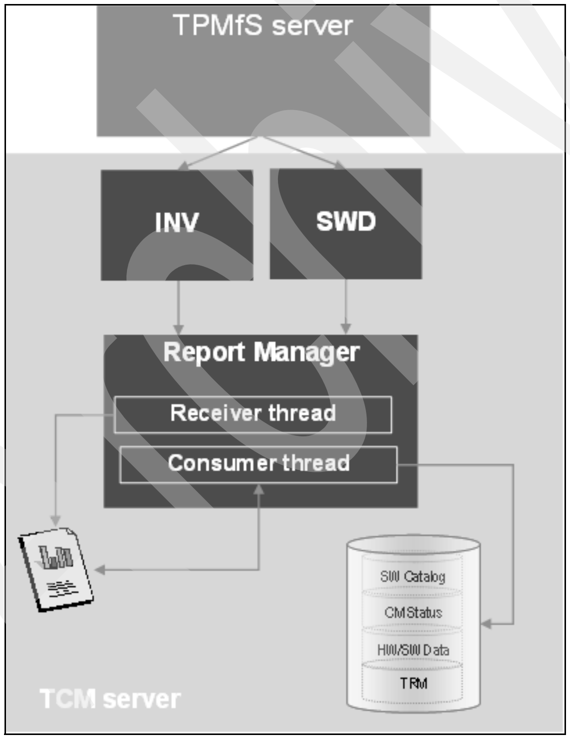


Figure 5-10 Report Manager - internals

The Report Manager is responsible for the following actions:

- Every request from the Tivoli Provisioning Manager for Software server to the Tivoli Configuration Manager applications registers the Report Manager as a callback object on the receive_report method.
- When the Tivoli Configuration Manager application completes, it notifies the Report Manager about the result of the operation on each target, using the registered callback method.
- The Report Manager's receive_report method stores each report on a temporary file in the local file system.
- A consumer thread in the Report Manager is responsible for consuming the enqueued reports and storing them into the inventory database.

As shown in Figure 5-11, the Report Manager also executes the following actions:

- The Reporting engine on Tivoli Provisioning Manager for Software server queries the infrastructures for the status of pending tasks.
- The Tivoli Management Framework infrastructure invokes the query_report method on the Report Manager component, and specifies a starting date and time, as well as the max number of reports.
- The Report Manager retrieves from its database all the reports starting from the specified date and time.
- The Reporting engine on the server side updates the status of the pending tasks in the DCM repository.

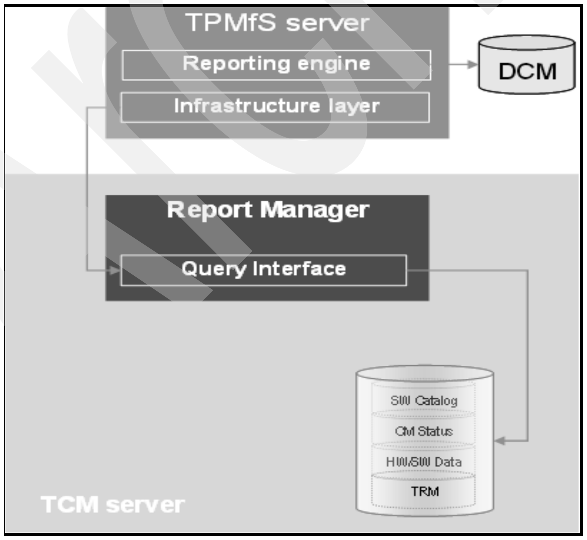


Figure 5-11 Report Manager—reports handling

REP_MANAGER table

The REP_MANAGER table is created on the inv_db database by the execution of the rep_*vendor*.sql script, which we described in "The rep_vendor.sql script" on page 98.

Table 5-9 describes the REP_MANAGER table schema.

Table 5-9 REP_MANAGER table schema

Column name	Type	Description
SERVER_ID	String	The Tivoli Provisioning Manager for Software server identifier. This information identifies which Tivoli Provisioning Manager for Software server originated the request on Tivoli Configuration Manager.
DIST_ID	String	The MDist2 distribution ID that identifies the Tivoli Provisioning Manager for Software task, which originated the request on Tivoli Management Framework.
STATUS	Integer	The MDist2 status of the operation (final status).
TARGET	String	The Tivoli Management Environment object ID used to identify each target TMA endpoint.
RECORD_MODE	String	The date and time information for each report. The Report Manager uses this information when querying the results.
INST_ID	String	A string containing a reference to the installable that the Tivoli Provisioning Manager for Software server uses when originating the request. This is valid only for software distribution operations.
APP_DATA	String	The application that originated the report
REGION_ID	String	The Tivoli Management Region identifier that identifies the Tivoli Region where the report was generated (when multiple Tivoli Management Regions share the same inventory database).
MESSAGES	String	The message associated to the report.

Configuration

When Tivoli Provisioning Manager for Software consumes the reports in the database, the Report Manager removes the entries related to those reports in order to keep the table size under control. You can disable this behavior by setting a specific idl attribute of the Report Manager component (delete_old_rows).

Example 5-9 shows some commands you can use to check the actual status of the delete_old_rows attribute and eventually change its value.

Example 5-9 Report Manager configuration

```
The Report Manager is registered on the Tivoli Management Region
server, as a distinguished resource called ReportManager.
To locate its OID run the following command:

[root@milan][/]-> wlookup ReportManager
1536006190.1.1067#RMEngine::ReportManager#

To get the value of the delete_old_rows attribute run the following
command:

[root@milan][/]-> idlcall 1536006190.1.1067 _get_delete_old_rows
TRUE

By default the output is set to TRUE. To disable the report entries
removal, use this command to set the attribute to FALSE:

[root@milan][/]-> idlcall 1536006190.1.1067 _set_delete_old_rows FALSE
```

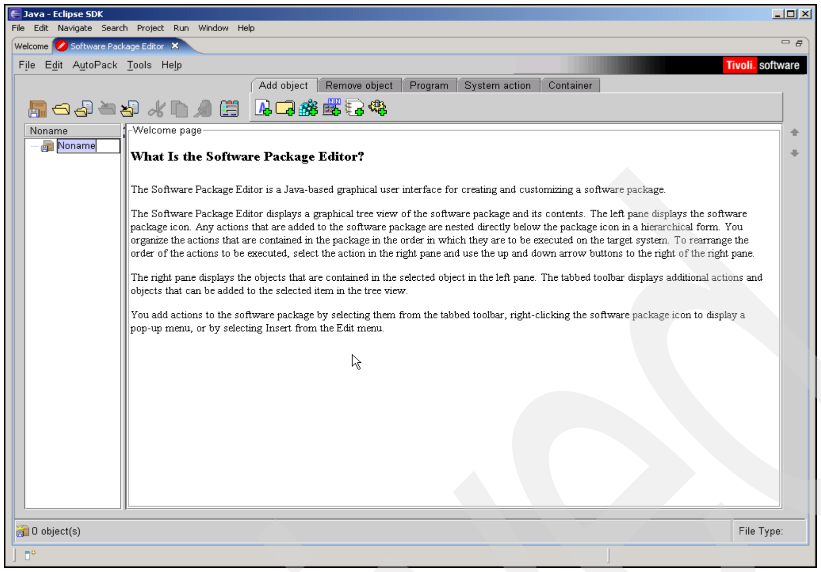
5.8 The new Software Package Editor

The Software Package Editor (SPE) is now available in two forms. You can launch it within Eclipse, as it was in the earlier version, or you can use Java Web Start to launch it.

5.8.1 Software Package Editor in Eclipse

To launch SPE in Eclipse, you must first install Eclipse version 3.1.1 or later, and then set up the Software Package Editor by unzipping the file spe.zip on the machine on which you want to run SPE.

Full details of the procedure for launching SPE in Eclipse is described in *Section 5.4* of the book *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1,* SG24-7261-00*.* Figure 5-12 on page 140 shows the SPE in Eclipse.



What Is the Software Package Editor?

The Software Package Editor is a Java-based graphical user interface for creating and customizing a software package.

The Software Package Editor displays a graphical tree view of the software package and its contents. The left pane displays the software package icon. Any actions that are added to the software package are nested directly below the package icon in a hierarchical form. You organize the actions that are contained in the package in the order in which they are to be executed on the target system. To rearrange the order of the actions to be executed, select the action in the right pane and use the up and down arrow buttons to the right of the right pane.

The right pane displays the objects that are contained in the selected object in the left pane. The tabbed toolbar displays additional actions and objects that can be added to the selected item in the tree view.

You add actions to the software package by selecting them from the tabbed toolbar, right-clicking the software package icon to display a pop-up menu, or by selecting Insert from the Edit menu.

Figure 5-12 The SPE with Eclipse

5.8.2 Software Package Editor from the Java Web Start

A new alternative is to set up the SPE within the Java Web Start. However, you must still set up the certificate and the path entry for Java as described in section 5.4 of the *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261.

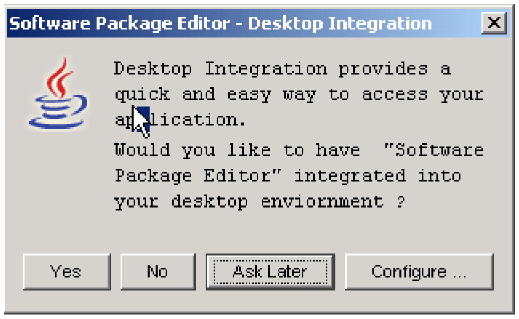
1. After you set up the SPE within the Java Web start, launch a Web browser, such as Internet Explorer® or Mozilla Firefox. Type the following into your browser's address field:

   ```
   https://<tpm host>:<port>/SPEWebStart/spe.jnlp
   ```

   By default, the port is 9045. Java Web Start downloads and launches the Software Package Editor.

   When SPE is launched, a notification window from Desktop Integration is displayed, as shown in Figure 5-13 on page 141.

2. Click **Yes** on the Desktop Integration message to have shortcuts added to your desktop and program files.


Software Package Editor - Desktop Integration

Desktop Integration provides a quick and easy way to access your application.

Would you like to have "Software Package Editor" integrated into your desktop enviornment ?

Figure 5-13   Desktop integration

Figure 5-14 shows the shortcuts that you can have added to your desktop. You can use these shortcuts to directly launch SPE.

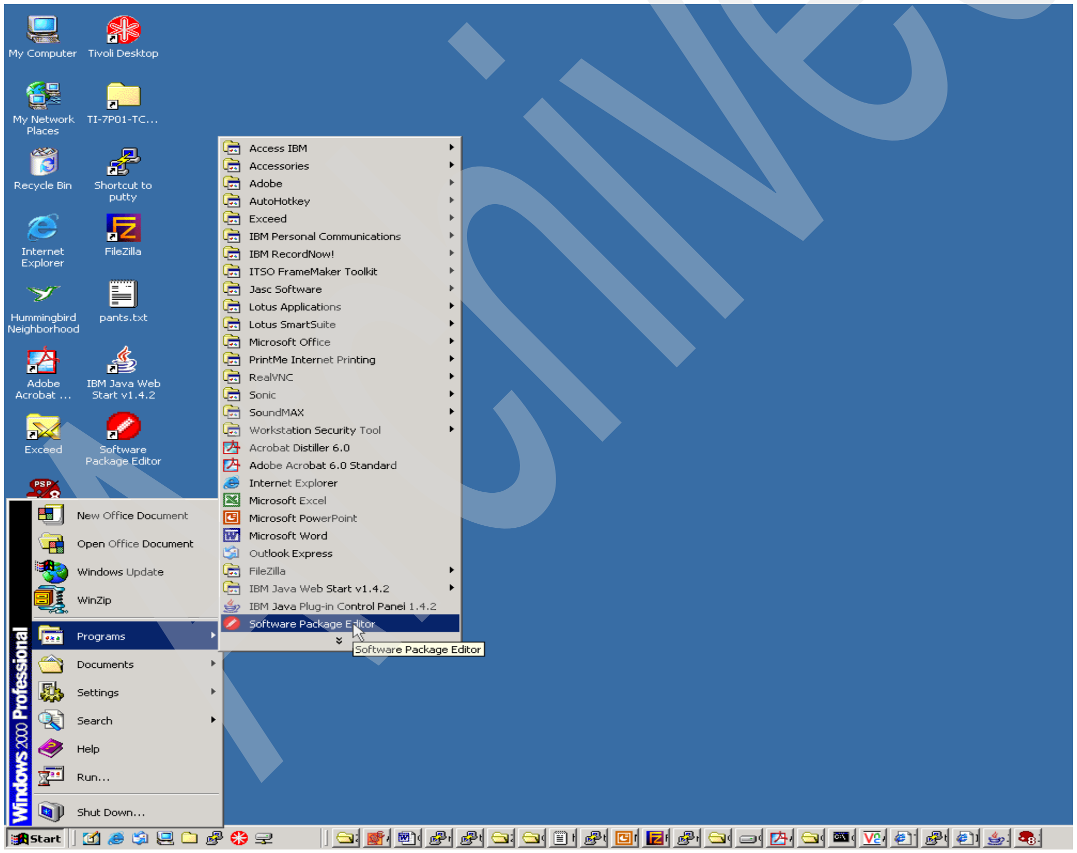


Figure 5-14   SPE shortcuts

The SPE launched in this manner is the same as the Eclipse launched version except that is does not appear in an Eclipse window, as shown in Figure 5-15.

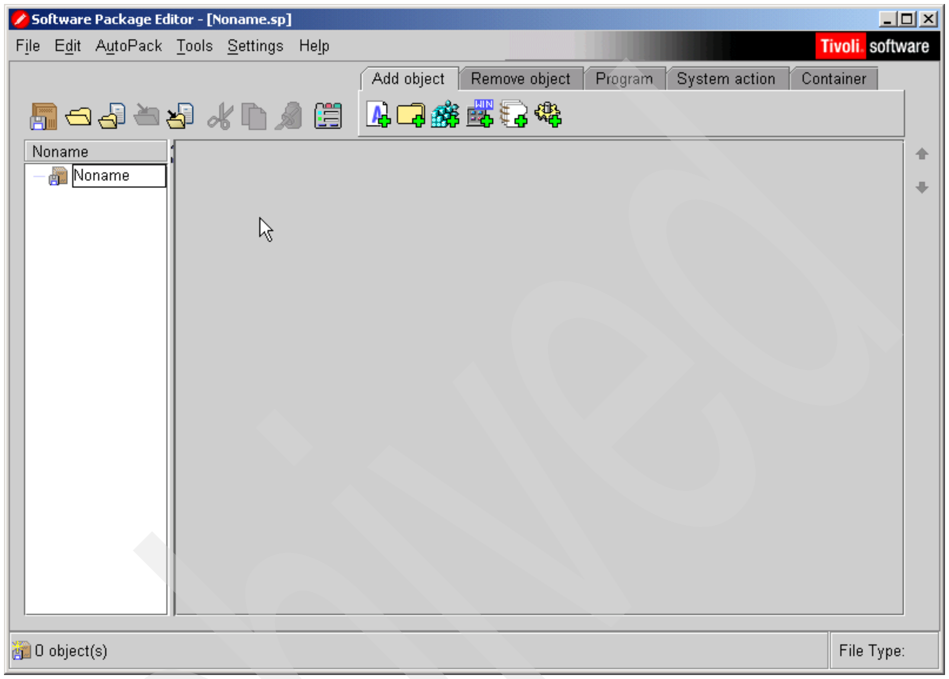
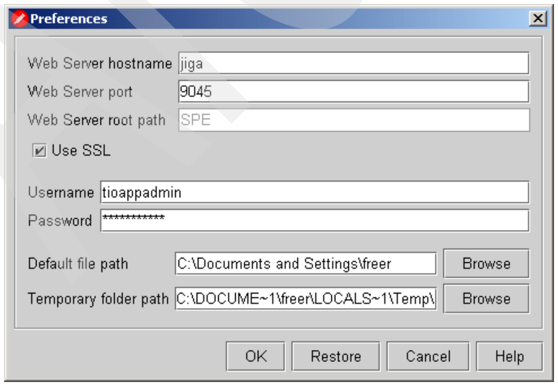


Figure 5-15 SPE launched using IBM Java Web Start

3. After SPE starts, configure it to connect to the Tivoli Provisioning Manager for Software server by selecting **Settings** → **Preferences**. A dialog appears similar to Figure 5-16.



Preferences

Web Server hostname jiga
Web Server port 9045
Web Server root path SPE
☑ Use SSL
Username tioappadmin
Password ************
Default file path C:\Documents and Settings\freer
Temporary folder path C:\DOCUME~1\freer\LOCALS~1\Temp\

Figure 5-16 SPE Preferences

5.9 The tc drivers from Tivoli Provisioning Manager for Software Fix Pack 2

Two new tc drivers are included with fix pack 2.

5.9.1 SWDDiscCLI tc driver

SWDDiscCLI.tcdriver is a new automation package that provides disconnected command line interface (CLI) commands on the Tivoli Provisioning Manager for Software server. The tar and zip file for each platform type is installed in $TIO_HOME/sie/bundles. The appropriate archive for the Tivoli Provisioning Manager for Software server platform is unzipped or untarred to $TIO_HOME/sie.

SWDDiscCLI.tcdriver also provides a workflow, *Build_SPB_from_SPD,* to build software package blocks (SPB) starting from the software package definition (SPD) files. The workflow requires the full path of the SPD that is to read and the full path of the SPB that will be generated.

5.9.2 New Tivoli common agent tc driver (automation package)

This driver is improved from earlier versions and now provides Tivoli common agent (TCA) and JRE™ images for all supported platforms. After it is installed, it creates a Tivoli common agent stack and a draft activity plan, TCA_PLAN_FOR_TMA.

Migrating a Tivoli management agent to a common agent

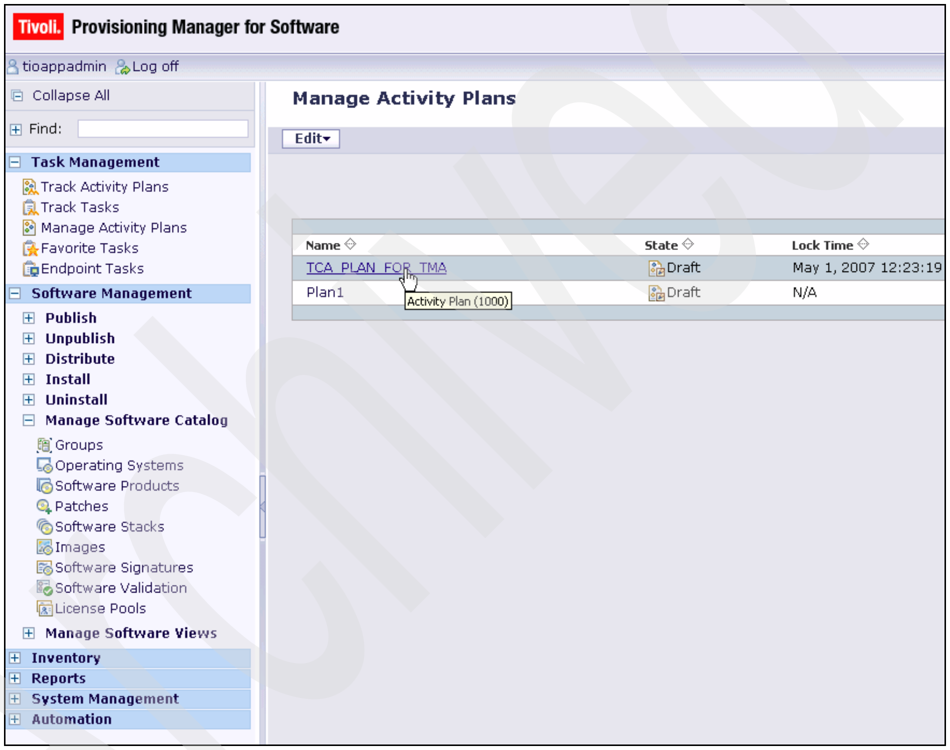
To use this new functionality, you must first enable coexistence and synchronize data so that the Tivoli management agents (TMAs) are available in Tivoli Provisioning Manager for Software, as described in 5.1, “Creating a coexistence environment” on page 94 and 5.3, “Replicating the Tivoli Management Framework data” on page 111.

Before you can use the TCA to Tivoli Management Framework functionality, Tivoli Provisioning Manager for Software must know the operating system of the devices. To set this, run an inventory scan against each device in Tivoli Management Framework, and then run the replication. Alternately, you can run a discovery from Tivoli Provisioning Manager for Software using the Tivoli Provisioning Manager Inventory Discovery discovery method.

Note: It is possible to set the operating system manually for a device; however, you should avoid doing this. An inventory scan at a later date adds a second entry for the operating system, which causes distribution problems.

After making the adjustments that we mentioned in the previous paragraphs, perform the following steps:

1. Select **Manage Activity plans**, select TCA_PLAN_FOR_TMA from the list, and then edit TCA_PLAN_FOR_TMA, as shown in Figure 5-17.



Name	State	Lock Time
TCA_PLAN_FOR_TMA	Draft	May 1, 2007 12:23:19
Plan1	Draft	N/A

Figure 5-17 Selecting TCA_PLAN_FOR_TMA

2. Select **View** → **Plan Properties**. Select the **Target** tab, and add a list of TMA targets.
3. Save the plan.
4. Run the plan.

5.10 Considerations of data coexistence

This section describes additional considerations when running Tivoli Provisioning Manager for Software and Tivoli Configuration Manager coexistence.

5.10.1 Selecting Tivoli Management Regions

You can incrementally add Tivoli Management Regions to the Discovery Configuration; however, every time the discovery runs, it scans all the Tivoli Management Regions listed in the configuration file. There is no way in the current implementation to select which TMR to discover if you configure a connection to more than one TMR. However, after you run a discovery once, the subsequent scans only return the differences.

5.10.2 Grouping

Tivoli Management Framework logical groups (policy regions and profile managers) are always replicated to DCM, even when you run a selective replica of either profiles or endpoints. The current implementation does not allow you to replicate only the resource instances without their containers.

The Tivoli Management Framework discovery spends most of its time managing the grouping of resources. The number of nested policy regions and profile managers has a big impact on the time required to replicate a Tivoli Management Framework environment.

5.10.3 Software discovery

The current implementation detects only software that was installed using software packages. Information about other types of software entries are neither collected nor reported into the DCM.

5.10.4 Tivoli Provisioning Manager for Software results

After you are using Tivoli Provisioning Manager for Software and Tivoli Configuration Manager coexistence, the changes caused by management operations in Tivoli Configuration Manager are not automatically updated in Tivoli Provisioning Manager for Software; instead, they are only updated following a Tivoli Configuration Manager discovery.

So, for example, if a package is installed on a device using Tivoli Configuration Manager rather than Tivoli Provisioning Manager for Software, that package appears to the Tivoli Provisioning Manager for Software server as though it is not installed, so it is possible to send the package again using Tivoli Provisioning Manager for Software, which causes it to error.

5.10.5 Migrating Tivoli Management Framework queries

As soon as you perform the federation step on Tivoli Provisioning Manager for Software, some views are created in the Tivoli Provisioning Manager for Software database to present data in a Tivoli Configuration Manager-like format. Also, the Tivoli Provisioning Manager for Software Report Framework is extended to query the new views.

During the replication step, a subset of the queries existing in Tivoli Management Framework is migrated. A query is migrated if both the "where" and the "group by" clauses contain all valid fields in the Tivoli Provisioning Manager for Software database. See Table 5-10 for further details.

Table 5-10 Migrating Tivoli Management Framework queries

Query	**State**
Subscription queries	Completely migrated.
Inventory queries	Partially migrated.
Custom queries	Partially migrated.
Historical queries	Not migrated at all.
Pervasive devices queries	Not migrated at all.
TCM Web User Interface queries	Not migrated at all.

For each migrated query, both a Report and a Dynamic Computer Group are created in Tivoli Provisioning Manager for Software.

The standalone management operations can address Dynamic Computer Groups coming from the Tivoli Management Framework migrated queries, while activities that belong to activity plans can contain targets that are defined as either Dynamic Groups or Reports.

Mapping Tivoli Configuration Manager to Tivoli Provisioning Manager for Software

We divided this chapter into two sections. The first part provides a feature-by-feature comparison between the common framework and Tivoli Configuration Manager operations and their equivalent in Tivoli Provisioning Manager.

The second part of the chapter covers the operational flow of several management operations in more detail.

This chapter has the following sections:

- ► "Operation comparison" on page 148
- ► "Tivoli Provisioning Manager for Software and Tivoli Configuration Manager operations in detail" on page 163
- ► "Driving Tivoli Management Framework from Tivoli Provisioning Manager for Software" on page 181

6.1 Operation comparison

In this section we listed many of the common operations that are regularly performed using Tivoli Configuration Manager and Tivoli Management Framework, along side the equivalent operation in Tivoli Provisioning Manager for Software.

When listing the Tivoli Provisioning Manager for Software functions, if the function is introduced in a fix pack, the fix pack number is highlighted.

The features are broken down by Tivoli Configuration Manager component and include:

- Software package preparation
- Software distribution operation
- Inventory
- Activity Planner
- Change Manager
- The Tivoli Configuration Manager Automated Patch Management Solution
- Tivoli Configuration Manager Web Interface
- Data Moving
- Pristine Manager
- Managing pervasive devices
- Publish
- Unpublish
- Distribute
- Install
- Uninstall
- Discovery
- Endpoint tasks

6.1.1 Tivoli Management Framework and Tivoli Configuration Manager configuration

We cannot describe every single configuration option possible with Tivoli Configuration Manager and the Tivoli Management Framework, so in this section Table 6-1 on page 149 focuses on some of the most important items that affect the operation of Tivoli Configuration Manager.

Table 6-1 Tivoli Management Framework and Tivoli Configuration Manager configuration

Operation or function	**Tivoli Configuration Manager**	**Tivoli Provisioning Manager**
Set network bandwidth utilization during distributions.	In Tivoli Management Framework, this is set on a per-gateway basis using the **`wmdist`** command. You can set bandwidth usage to a maximum figure per endpoint, with an overall ceiling figure.	Tivoli Provisioning Manager for Software allows you to set the overall bandwidth utilization of a depot in kilobytes per second. You can also specify depot bandwidth usage as Adaptive, which means that when network traffic is heavy, the transfer rate decreases, and when network traffic is light, the transfer rate increases again.
Maintain central record of what software was installed on targets.	Tivoli Configuration Manager provides Software Distribution and Inventory integration function. The inventory record of each target is maintained with the status of the software distribution.	The Data Center Model (DCM) maintains a record of what software is installed and where the software is installed. Various reports are available to show this information, and it is also available on the Computer/ Software tab of the GUI. A “Software Installation” object is created when an install is submitted. This has a status of “pending” during the install, which changes to “active” after the action is complete. The object is deleted if the install fails, or the package is removed.
Alerting, Tivoli Enterprise™ Console (TEC) integration.	You can configure Tivoli Configuration Manager to send alerts to the Tivoli Enterprise Console® (TEC) when various events occur. These events include the start, completion, and failure of SWD operations.	Tivoli Provisioning Manager for Software does not provide TEC integration. Tivoli Provisioning Manager for Software can send SNMP e-mail notifications regarding the status of specific tasks or workflows. Notifications can be sent about reports, installing, distributing and publishing software, discovery activities, compliance and favorite tasks.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Assign endpoints to the correct gateways.	The endpoint policy select_gateway_policy assigns a primary gateway and list of alternates to each new endpoint. You must manually code the policy script.	Depots are the nearest equivalent in Tivoli Provisioning Manager for Software to gateways, but they are only involved in dynamic content delivery service distributions and not for other communication, such as Deployment Engine workflows or device management service job delivery. Tivoli Provisioning Manager for Software does not enforce a static relationship between common agent and depot. During dynamic content delivery service distributions, the most appropriate depot or list of depots is chosen based upon the common agent TCP/IP address and the zone and region definitions.
Move target to alternate gateway.	Lightweight Client Framework (LCF) endpoints automatically use a predefined alternate gateway if their primary is unavailable. Endpoints can be manually moved to alternative gateways from the CLI if required.	There is not a fixed relationship between the common agent and the depot. The dynamic content delivery service manager assigns depots dynamically when a download plan is generated.
Restrict which endpoints can register with the Tivoli Management Region.	You can control which new LCF endpoints can connect to a Tivoli Management Region with the allow_install_policy endpoint policy.	There is no equivalent policy in Tivoli Provisioning Manager. However a common agent must have the registration password before it can register with the agent manager.
Ensure common endpoint naming policy.	Use the **`wepmgr`** command to enforce endpoint label formats. Use the **`wep`** command to change endpoint labels.	Managed systems are always labeled with their host name.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Configure agent parameters from central server.	You can set LCF parameters with the **wep** command.	There is no common way to modify parameters for the common agent sub-agents. One commonly tuned set of parameters control how the agent polls device management services for jobs. You can change the following parameters with the workflow JES_Parameter_Set: PollingEnabled, PollingEnd, PollingInterval, PollingStart, enableFirewallSupport and logSize. You can create workflows to change other parameters, such as to enable peer-to-peer operation.

6.1.2 Managed system management operations

Table 6-2 covers the type of operations that are routinely performed on or by managed systems.

Table 6-2 Managed System Management operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Investigate current managed system status.	You can query the current status of individual endpoints with the `wep` ***eplabel*** `status` command.	Run the workflow **TCA_PingAgent** against port 9510 to check the status of a TCA.
Report on recent managed system login dates and times.	The **wepstatus** command displays the last reported status of some or all endpoints. The **wep ls** command can list information, such as the last login time.	In Tivoli Provisioning Manager for Software, this information is stored in the Agent Manager database and is accessible using reports. The standard report "Inventory010 - What is the status of all endpoints?" shows the last time all of the agents reported in along with the last error reported and other information. The discovery configuration "IBM Tivoli Common Agent Discovery" should be regularly scheduled to update the DCM from the Agent Manager database. You can create custom reports to present the information in different formats.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Run an ad-hoc command on a managed system.	The `wadminep` command can perform a variety of actions on individual endpoints.	You can run commands on individual managed systems using workflows. This mechanism does not use the Scalable Distribution Infrastructure (SDI).
Run a script on a managed system.	You can run scripts on multiple managed nodes or endpoints. The result of the script plus any output can be returned. You can create a Job that runs a task with predefined parameters and options.	Tivoli Provisioning Manager for Software 5.1 FixPack 02 and Tivoli Provisioning Manager 5.1 FixPack 02 give you the capability to create and run endpoint tasks on common agent systems through SDI. The facility to run endpoint tasks was available with Tivoli Provisioning Manager for Software GA code and Tivoli Provisioning Manager FixPack 01 but only with tasks imported from Tivoli Management Framework and run from an activity plan. Without using SDI (or Tivoli Management Framework), workflow and scriptlet tasks can be used. These run on the server and contact target systems one at a time.
Automatically power on targets using Wake on LAN (WoL).	Tivoli Configuration Manager has an option that you can set to send a WoL packet to endpoints prior to the start of a distribution. The packet is sent from the endpoint's gateway. The wadminep CLI is also available to send the WoL packet as required.	Tivoli Provisioning Manager for Software supports Wake on LAN but the WoL broadcast packet is sent from the Tivoli Provisioning Manager for Software server and can be initiated by the workflow Device.PowerOn. The workflow WOL_TaskTargetsSubmitted can be run against a group of targets that were previously the target of a software distribution to wake those devices that have not completed the distribution.
Use MultiCast to minimize network usage.	Tivoli Configuration Manager can use the MultiCast functionality of MDIST2 to broadcast a distribution to a number of endpoints at once.	Tivoli Provisioning Manager for Software does not currently support MultiCast.

6.1.3 Software package preparation

Table 6-3 covers the operations typically performed when building and preparing software packages.

Table 6-3 Software package preparation

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Edit a software package in a graphical editor.	You can run the Software Package Editor (SPE) on managed nodes or on other systems.	Tivoli Provisioning Manager for Software 5.1 FixPack 02 includes a Java Web Start-based version of the SPE that you can install on administrators workstations. There is also an Eclipse-based version of the SPE that is installed by default on the Tivoli Provisioning Manager for Software server and can also be installed on administrators workstations. Both versions allow you to save the software packages locally and to the Tivoli Provisioning Manager for Software repository.
Convert software package from unbuilt to built.	You can convert software packages from unbuilt to built in the SPE GUI or using the managed node command **wconvspo** or the disconnected environment command **wdbldspb**.	The SPE GUI can save software packages in built format. The CLI commands are available in the disconnected-command environment on the Tivoli Provisioning Manager for Software server.
Import a software package from administrators' workstation into the management environment.	Software packages can be built onto the Source Host from the SPE when it is started from the Tivoli Management Framework Desktop. The CLI **wimpspo** is also available.	You can import a Software Product using the SPE GUI, the Tivoli Provisioning Manager for Software GUI, or using SOAP commands (FixPack 02 onwards). SOAP commands are covered in detail in Chapter 9, "Automating Tivoli Provisioning Manager for Software tasks using the SOAP CLI" on page 225.
Test software packages before importing and distributing.	The SWD disconnected CLI provides a number of commands for installing and querying software packages on the local system.	The disconnected CLI is available on the Tivoli Provisioning Manager for Software server. It is also included with a subagent on the common agent but some work is involved in configuring the environment to use it effectively in a disconnected way.

6.1.4 Software distribution operation

Table 6-4 shows the common functions performed when deploying software packages.

Table 6-4 Software distribution operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Load software package onto the gateway or depot prior to distribution.	You can pre-load the software package into the repeater's depot cache using the GUI, CLI `wldsp,` or using Activity Planner.	You can load software products onto depots with the Publish operation in the GUI. Note that the behavior of a Tivoli Provisioning Manager for Software depot is different from a Tivoli Management Framework gateway. In Tivoli Management Framework, if a depot is marked as having permanent storage, then built packages are permanently stored in the depot cache when they are distributed through the gateway. This is not the case in Tivoli Provisioning Manager for Software, where packages are automatically removed from the depot when the distribution is complete unless they were previously published to that depot. Publish and Load operations are not permitted in Tivoli Provisioning Manager Activity Plans.
Unload software package from staging servers.	You can unload the software package from a gateway's depot cache using the GUI, CLI `wuldsp,` or using Activity Planner.	You can unpublish software products from all depots from the GUI. Unloading from individual depots is not supported. Publish and Load operations are not permitted in Tivoli Provisioning Manager Activity Plans.
Distribute and install a software package in one operation.	A Non-Transactional Software Install performs a distribution of a software package followed immediately by an installation, which you can perform from the GUI, in an activity plan, or by the command `winstsp -tn`. If the package is not built, it is temporarily built at distribution time.	Initiate non transactional installs using the SOAP interface (fix pack 02 onwards) or using the GUI. Most operations can also be performed by workflows. Tivoli Provisioning Manager for Software does not currently support the concept of unbuilt software packages.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Software package—Distribute without install.	You can initiate a transactional install to distribute a software package to targets without performing an installation. As above, this can be done from Activity Plan Model (APM), GUI, or CLI. The software package has a status of `IP---` in both the Tivoli Configuration Manager database and on the target.	This operation is called a distribution in Tivoli Provisioning Manager.
Software package—Install only.	You can use the APM, GUI, or the CLI to perform a Commit operation to install a software package that was previously distributed. The Commit fails if the package was not already distributed.	There is no separate commit operation in Tivoli Provisioning Manager. Instead, an install is performed. If the package was already distributed, then the install effectively performs a commit, but if it was not distributed, then it is distributed before installation.
Install a software package at reboot.	You can perform a commit so that it does not occur immediately but will start the next time the device starts (Windows only). Install a transactional with separate commit or auto commit, either option can either commit at the next reboot or initiate a reboot.	Commit at reboot is not supported in Tivoli Provisioning Manager.
Remove Software.	You can remove a software package by APM, CLI, or via the GUI.	You can initiate an uninstall action via the SOAP interface or using the GUI.
Backup overwritten files when performing a software distribution. Also accept or reject the installation.	You can instruct Tivoli Configuration Manager to take a backup copy of files that the SPB directly modifies or overwrites. This backup can then be deleted (accept the install) or restored (reject the install).	This is not supported in Tivoli Provisioning Manager.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Perform a forced installation.	Specify the Force option to ensure that the installation proceeds even if the package is already installed or is in an error state (`IC--E`).	There is no force option within the install operation in Tivoli Provisioning Manager but you can replicate this behavior for SPB installs. Each SPB type software product can have a force parameter in its configuration template that you can set to true to force the install. You can define and specify different templates at install time, as well as change the values within a template at install time.
Verify a software installation.	Tivoli Configuration Manager can perform a verification of a previously installed software package.	Tivoli Provisioning Manager for Software does not support the verify operation.
Byte level differencing.	Tivoli Provisioning Manager for Software can perform a distribution based upon a binary byte-level difference between the source and target.	Tivoli Provisioning Manager for Software does not support byte-level differencing.
Mobile console.	You can install the Tivoli Mobile Console (TMC) on Windows targets. The TMC is allows users working remotely to pause and resume distributions.	Tivoli Provisioning Manager for Software does not currently support the Tivoli Mobile Console.
Install a sequence of software products	Typically an Activity plan is created to install a series of software packages in sequence. The activities can be conditioned so that the plan stops if any of the packages fail.	Tivoli Provisioning Manager for Software supports the use of Activity Plans; however, the delay between activities could be extensive due to the client initiated polling mechanism. An alternative to Activity Plans is to create a Software Stack, which is a list of software packages that you install in a particular order. Software Stacks can also be used in Compliance Checking.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Use variables to control software package execution.	Variables are used for a number of purposes in software packages, for example, to specify an installation path at distribution time rather than package built time. The variables can be stored on the target in swdis.ini but are more often specified in the distribution command.	Variables are still available in Tivoli Provisioning Manager for Software but their values are specified in configuration templates that are associated with the software product. You can change the values at distribution time.
Pause distributions.	You can pause software distributions during the delivery phase but not while the commit is running.	It is not currently possible to pause SDI distributions.
Undo failed installations.	In Tivoli Configuration Manager you can perform an undo operation on a software package so that if a fatal error occurs during the installation process, the undo operation is triggered automatically and returns the system to its state prior to the execution of the last install or remove operation.	Undo is not currently supported in Tivoli Provisioning Manager.
Set distribution lifetime.	You can set the deadline parameter at distribution time to control when a distribution will expire.	By default, file distribution to endpoints expires in two hours (7200 seconds). The global variables retryWindowSec and maxRetryIntervalSec allow you to change the default.
Provide messages to user.	Tivoli Configuration Manager can display messages to the user on Windows platforms when a package is being installed. The options user_notification and enable_notification are used to control this.	Tivoli Provisioning Manager does not support user notification except for reboot notification when using automated patch management.

6.1.5 Inventory

Table 6-5 shows the common inventory related tasks and operations.

Table 6-5 Inventory operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Create an Inventory profile definition.	Inventory profile objects are most commonly created and customized using the GUI, but you can also use the command line tools.	The equivalent object in Tivoli Provisioning Manager for software is the Discovery Configuration, which you create and modify using the GUI.
Scan for software or hardware configuration.	Inventory Config Profiles are configured to perform a hardware scan, software scan, or both. The hardware scan can be limited to scan for certain device types only (mice, disk, memory, and so on). Software information can be gathered by scanning file information, patch information (Windows only) or from an OS or Registry scan.	Discovery Configurations are not just used to scan managed systems for hardware and software. They can also scan networks for devices, extract information from Tivoli Application Discovery Dependency Manager, Active Directory, and so on. The equivalent to an Inventory Config profile is a Discovery Configuration that uses the discovery method “Tivoli Provisioning Manager Inventory Discovery”. You can configure these device discoveries to scan for hardware (no subdivision), software, or both. The software scan can scan the registry or file system using signatures. Unlike Tivoli Configuration Manager scans, you cannot configure discoveries in Tivoli Provisioning Manager for software to return only the differences since the last scan.
Scan, retrieve, and store custom information.	Inventory Config Profiles can run scripts to create and gather custom Management Information Format (MIF) files. The information gathered is stored in custom database tables.	It is not currently possible to perform custom scans in Tivoli Provisioning Manager.
Initiate a scan from an endpoint.	Run the `wepscan` command on endpoints to perform scans locally rather than as the result of a profile distribution. The scan results can be gathered at a later date.	Endpoint initiated scans are not supported in Tivoli Provisioning Manager.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Query inventory data.	A number of queries are provided to extract information from the inventory database. These queries are often used for target selection purposes.	A powerful reporting engine is available in Tivoli Provisioning Manager for Software that can produce detailed reports in a number of formats.
Use inventory data for target selection.	You can use the set of Subscription Queries to select targets for profile distributions.	You can use reports to define the membership of Dynamic Groups, which you can use for target selection, compliance, and so on.

6.1.6 Activity Planner

Table 6-6 covers the Activity Planner functionality. Much of the functionality of APM was maintained between Tivoli Configuration Manager and Tivoli Provisioning Manager for Software, but there are some exceptions.

Table 6-6 APM functions

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Edit, submit, and manage Activity Plans.	Two activity plan GUIs are available. Use the Editor to import, create, modify, and save plans. The Monitor GUI enables you to submit, monitor, and control plans. CLI commands are available for almost all activity plan functions.	The Editor GUI runs as a Java applet under the Tivoli Provisioning Manager for Software GUI and has a very similar look and feel to the Tivoli Configuration Manager version. You can submit plans from the Task Management section of the Tivoli Provisioning Manager for Software GUI. There is no dedicated activity plan monitor GUI, but you can track plans from the Track Tasks section.
Import plan definition stored in an XML format file.	Use the GUI or CLI to import XML definitions of activity plans.	Use the GUI to import XML definitions of activity plans.
Define start and end times for activity plans.	You can specify a start time for Activity Plans; likewise, you can also set a relative or absolute end time.	You can specify a start time for Activity Plans; likewise, you can also set a relative or absolute end time.

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Define execution windows for activities.	You can define execution windows for activities. The distributions in the activities are paused outside of the window times.	Execution windows are not fully supported in Activity Plans within Tivoli Provisioning Manager for Software. If a window is specified, then the task is not submitted into device management service until the window opens. The operation is never paused (unlike MDIST2, dynamic content delivery service does not support pausing distributions). Due to the polling interval of the targets, the distribution may not start exactly when the window opens.
Load a software package onto a depot.	You can use Activity Planner activities to load software packages onto depots.	The load operation is not supported within APM in Tivoli Provisioning Manager for Software.
Condition an activity upon the results of previous activities.	You can start activities based upon the results of previous activities, down to target level if required. You can base these conditions on completion, success, or failure. You can also start activities based on the result of a load operation on the targets' gateway, which is called conditioning by depot.	The same conditioning is available in Tivoli Provisioning Manager for Software with the exception that the depot conditioning as Load operations is not supported. Note that because of the way that dynamic content delivery service and device management service work, there could be noticeable delays between conditioned activities.
Define targets.	You can specify targets at either the plan level (all activities to all targets) or at the activity level (each activity has its own target list). You can resolve target lists from a number of sources, such as queries, profile manager subscribers, files, and so on.	In Tivoli Provisioning Manager for Software, you can also specify targets at the plan or activity level. The targets can be provided as a list of target names, a file, from reports, or from groups' membership.
Pause activity plans.	You can pause Activity Plans from the GUI or the CLI. If an activity is paused then the MDIST2 distribution is paused.	As it is not possible to pause distributions that have already started, pausing an activity plan pauses the next activity, and the running activity will continue to run.

6.1.7 Change Manager

Table 6-7 covers the Change Manager (CCM) component of Tivoli Configuration Manager and how this relates to Tivoli Provisioning Manager. This component does not exist in Tivoli Provisioning Manager for Software, but much of its functionality is available and indeed extended by using compliance checks and remediation actions.

Table 6-7 Change Manager operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Define the software components that must be installed on groups of targets.	In Tivoli Configuration Manager, this is accomplished by creating Reference Models that define what software should be or should not be installed on groups of systems.	In Tivoli Provisioning Manager for Software, this is achieved by using compliance. To use compliance: 1. Groups of computers are defined and one or more software compliance checks are added, which can be individual software products, software stacks, groups of software products, or even patches. 2. Optionally, an inventory scan is run. 3. A compliance check runs that compares the desired state with the state that is recorded in the DCM. This check produces remediation recommendations that detail what software needs installing. 4. After approval, the recommendations are applied, which generates software distributions.

6.1.8 The Tivoli Configuration Manager Automated Patch Management Solution

We cover the comparison between Tivoli Configuration Manager and Tivoli Provisioning Manager patch management solutions in detail in Chapter 7, “Patch management” on page 185.

6.1.9 Tivoli Configuration Manager Web Interface

Table 6-8 covers the use of the Web interface for making Software packages and inventory profiles available to users.

Table 6-8 Tivoli Configuration Manager Web Interface

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
User initiated install of software package.	Using the Tivoli Configuration Manager Web interface authorized users can download and install software packages and perform inventory scans of their workstation.	There is no support for end-user initiated installs or scans via a Web interface in Tivoli Provisioning Manager.
User initiated inventory scan.		

6.1.10 Data Moving

Table 6-9 covers the Data Moving function of Tivoli Configuration Manager in this section.

Table 6-9 Data Moving operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Move or copy files to or from endpoints.	Data Moving can move single or multiple files to and from endpoints using MDIST2 methods. Many options are available including the use of wild cards, searching recursive directories, and the ability to select multiple targets for each request.	Tivoli Provisioning Manager for Software does not fully support data moving. Tivoli Provisioning Manager for Software can use the SDI to send to but not to retrieve single files from TCAs using: Software Management → Distribute → Files. First, add the file to be sent to the repository from: Inventory → Infrastructure Management → File Repositories → LocalFileRepository → Edit → Add File. You can also use either DE or SDI-based software distribution techniques to distribute files. You can use workflows to copy and retrieve files, but this does not use the SDI infrastructure.

6.1.11 Pristine Manager

Table 6-10 provides a brief comparison between Tivoli Configuration Manager Pristine Manager and the functions provided with Tivoli Provisioning Manager for OS Deployment.

Table 6-10 Pristine Manager

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Install an operating system.	Tivoli Configuration Manager includes the Pristine Manager, which is essentially an integration of Tivoli Configuration Manager with Microsoft Automated Deployment Services (ADS) or Microsoft Remote Installation Services (RIS).	Tivoli Provisioning Manager & Tivoli Intelligent Orchestrator include IBM Tivoli Provisioning Manager for OS Deployment Embedded Edition, which provides a powerful and versatile way to install OS images. In Tivoli Provisioning Manager for Software, this is an optional component that you can purchase separately. The separate product, Tivoli Provisioning Manager for OS Deployment, currently has more features than the Embedded Edition.

6.1.12 Managing pervasive devices

Table 6-11 covers the management of mobile devices such as PalmOS and Windows CE.

Table 6-11 Pervasive device operations

Operation or function	Tivoli Configuration Manager	Tivoli Provisioning Manager
Install software on pervasive devices.	Tivoli Configuration Manager includes Resource Manager, which enables basic operations on pervasive devices, such as creating or deleting devices, distributing software, and scanning and customizing devices.	Tivoli Provisioning Manager for Software does not currently support any pervasive devices.
Perform inventory scan on pervasive devices.		

6.2 Tivoli Provisioning Manager for Software and Tivoli Configuration Manager operations in detail

In this section, we describe, in detail, the operational flow of some of the more common operations in Tivoli Provisioning Manager for Software and Tivoli Configuration Manager.

6.2.1 Publish

The Publish operation stores software installables on depot servers to reduce network traffic for frequently distributed packages and to manage slow, unreliable network links more efficiently.

The Publish operation exposed by the Web User Interface is a static publish because it allows you to select the depots where the file will be loaded but does not leverage the dynamic content delivery service capabilities of dynamically loading depots.

Publish is implemented as a workflow:

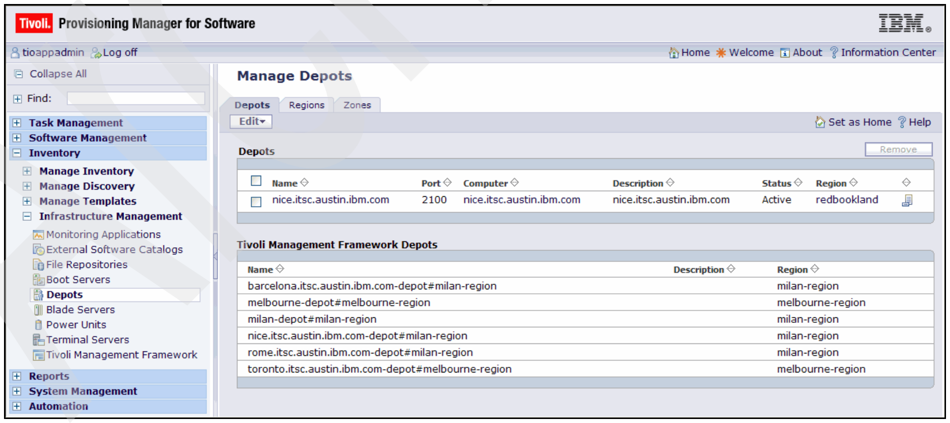
- TMF_FileRepository_Publish for TMF depots.
- CDS_PublishFile for Tivoli Provisioning Manager depots.

The Tivoli Management Framework implementation invokes the Load operation on a software package.

Depots Web User Interface page

Both dynamic content delivery service depots and Tivoli Management Framework Repeaters are mapped to Tivoli Provisioning Manager for Software depot instances.

The Web User Interface page for Tivoli Provisioning Manager for Software depot servers provides two separate sections for dynamic content delivery service and Tivoli Management Framework depots, as shown in Figure 6-1.



Depots

Name	Port	Computer	Description	Status	Region
nice.itsc.austin.ibm.com	2100	nice.itsc.austin.ibm.com	nice.itsc.austin.ibm.com	Active	redbookland

Tivoli Management Framework Depots

Name	Description	Region
barcelona.itsc.austin.ibm.com-depot#milan-region		milan-region
melbourne-depot#melbourne-region		melbourne-region
milan-depot#milan-region		milan-region
nice.itsc.austin.ibm.com-depot#milan-region		milan-region
rome.itsc.austin.ibm.com-depot#milan-region		milan-region
toronto.itsc.austin.ibm.com-depot#melbourne-region		melbourne-region

Figure 6-1 Dynamic content delivery service and Tivoli Management Framework depots

You cannot configure Tivoli Management Framework Repeaters from the Tivoli Provisioning Manager for Software Web User Interface; instead, you must use Tivoli Management Framework tools.

Publish using the Scalable Distribution Interface

The publish operation is performed in several steps:

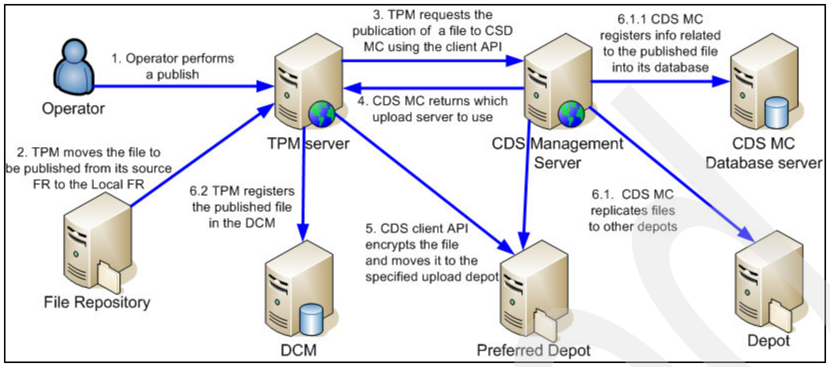
- The Tivoli Provisioning Manager for Software server:
  - Moves the files into the Local File Repository
  - Invokes the dynamic content delivery service client API to publish the file to a depot
- The dynamic content delivery service client API communicates with the dynamic content delivery service management center to:
  - Find the correct Upload Server for the target list
  - Add the file information into the dynamic content delivery service management center
  - Schedule any needed replication to other depots
- The dynamic content delivery service management center instructs the dynamic content delivery service client API to move the file from the Tivoli Provisioning Manager server to a specific depot.
- The Tivoli Provisioning Manager for Software server then updates a DCM table (FileTransfer) to manage the file:
  - Associating a dynamic content delivery service file ID with the installable being published
  - Associating a target list ID that represents the list of depots

Selecting a list of depots for a publish operation forces the dynamic content delivery service management center to replicate the file to all the depots in the list. If one of them is not available:

- Dynamic content delivery service management center publishes the file to a temporary depot in the list of available depots:
  - Temporary depot selected by nearness
- The publish operation completes successfully:
  - Tivoli Provisioning Manager for Software Web User Interface does not provide the capability of tracking the publish operation by depot
- Dynamic content delivery service management center can detect when the depot becomes available again. In this case it moves the file from the temporary depot to the depot that the operator selected.

Flow

Figure 6-2 depicts the publish operation flow.



Operator
1. Operator performs a publish
TPM server
3. TPM requests the publication of a file to CSD MC using the client API
4. CDS MC returns which upload server to use
CDS Management Server
6.1.1 CDS MC registers info related to the published file into its database
CDS MC Database server
2. TPM moves the file to be published from its source FR to the Local FR
File Repository
6.2 TPM registers the published file in the DCM
DCM
5. CDS client API encrypts the file and moves it to the specified upload depot
Preferred Depot
6.1. CDS MC replicates files to other depots
Depot

Figure 6-2 Publish operation flow

If the selected file is already associated with the Local File Repository, then the move described in step 2 is not performed.

6.2.2 Unpublish

The unpublish operation reverts the action performed by the Publish, removing the software installable from the depot location.

- It allows the selection of one or more installable files.
- It does not support the selection of specific depots, since it addresses all the depots where the software was previously published.
- It is implemented as a workflow:
  - TMF_FileRepository_UnPublish for Tivoli Management Framework depots
  - CDS_UnPublishFile for Tivoli Provisioning Manager depots

  Tivoli Management Framework implementation invokes the unload operation on a software package.

6.2.3 Distribute

The distribute operation delivers a software installable to each target endpoint without installing it to facilitate a staged installation. It allows you to select one or more software definitions to distribute and one or more targets (computers or groups of computers).

For Tivoli Management Framework terminology, you can perform a staged installation by these operations:

- Transactional install
- Commit

The corresponding mapping in Tivoli Provisioning Manager for Software is represented by the following operations:

- Distribute
- Install

The distribute operation for SPB installables allows the following options, which are passed to the software management subagent and used when installing the package:

- The dependency check allows you to check package dependencies on the local catalog
- The force allows you to force installation regardless of the local catalog contents

For TMA targets, the distribute operation is implemented as a transactional installation, where the software installable is delivered to the target but the application is not installed.

Scalable Distribution Infrastructure

The distribute operation consist of several phases:

- The dynamic content delivery service management center selects the depot servers that are nearest to the targeted agents to pre-cache the software. The management center uses dynamic depot selection, where it is given a list of targets (IP address and domains) and generates a list of depot servers that can be used to pre-cache the distribution based on their nearness to the targets and the number of targets.
- The Tivoli Provisioning Manager for Software server waits only until the publish to the first depot server has completed. After this is complete, the Tivoli Provisioning Manager for Software server submits a device management service job request to instruct the targets to start downloading the file.
- When a dynamic content delivery service client requests a download plan for a file, the dynamic content delivery service management center checks:
  - If the depot servers in the client's zone (if it is a branch office for example) are still waiting to replicate the file, and no peer has the file, then the client will wait for a time that the management center specifies and then performs another request for a download plan.

  – If the zone has only peers, but none of them have the file, then the management center checks to see if there are any distributions in progress to the target's peers. If so, the client waits for a time that the management center specifies and then performs another request for a download plan.

- When all of the targets of the distribution are completed, the Tivoli Provisioning Manager for Software server checks to see if there are any other outstanding distributions using the same file. If there are no outstanding distributions, then Tivoli Provisioning Manager for Software instructs the dynamic content delivery service to delete the file on all depot servers.

Flow

The distribute operation flow can be divided into three stages:

Figure 6-3 illustrates that in stage 1 the file is published to dynamic content delivery service infrastructure.

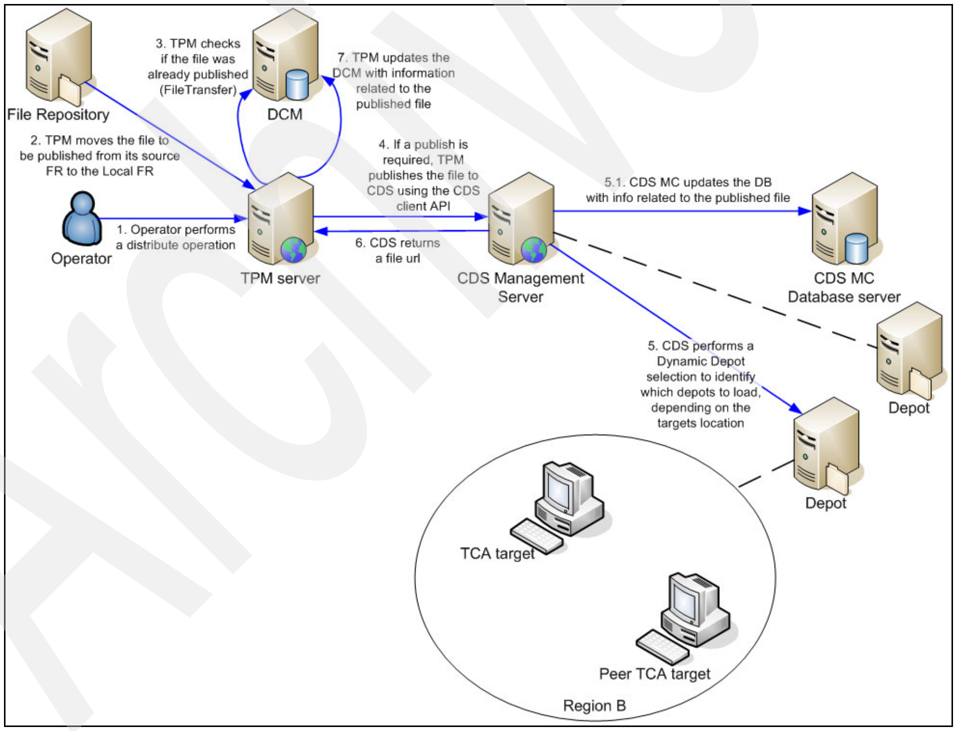

File Repository
3. TPM checks if the file was already published (FileTransfer)
DCM
7. TPM updates the DCM with information related to the published file
2. TPM moves the file to be published from its source FR to the Local FR
4. If a publish is required, TPM publishes the file to CDS using the CDS client API
5.1. CDS MC updates the DB with info related to the published file
Operator
1. Operator performs a distribute operation
TPM server
6. CDS returns a file url
CDS Management Server
CDS MC Database server
5. CDS performs a Dynamic Depot selection to identify which depots to load, depending on the targets location
Depot
Depot
TCA target
Peer TCA target
Region B

Figure 6-3 Distribute operation flow - stage 1

If the file was already published, this first stage stops at step 3 (see Figure 6-3 on page 168) because the dynamic content delivery service URL is already available in the DCM.

Figure 6-4 illustrates that in stage 2, a distribution job is created on the device management service infrastructure.

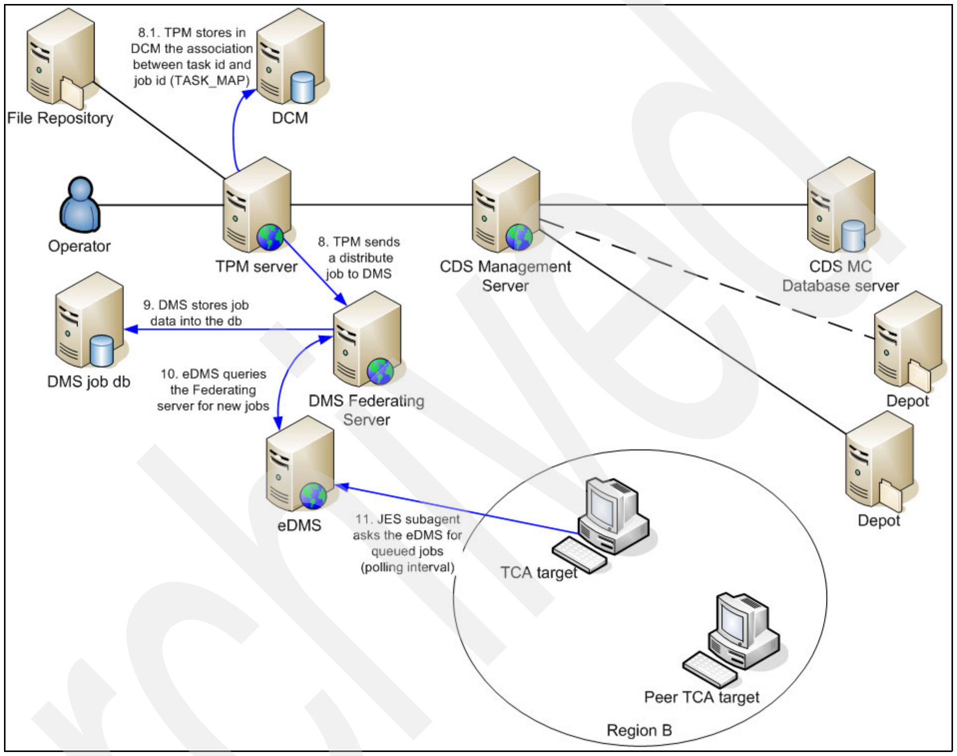

8.1. TPM stores in DCM the association between task id and job id (TASK_MAP)
File Repository
DCM
Operator
TPM server
8. TPM sends a distribute job to DMS
CDS Management Server
CDS MC Database server
9. DMS stores job data into the db
DMS job db
10. eDMS queries the Federating server for new jobs
DMS Federating Server
Depot
Depot
eDMS
11. JES subagent asks the eDMS for queued jobs (polling interval)
TCA target
Peer TCA target
Region B

Figure 6-4 Distribute operation flow - stage 2

In stage 3, illustrated in Figure 6-5, the agent processes the distribution job.

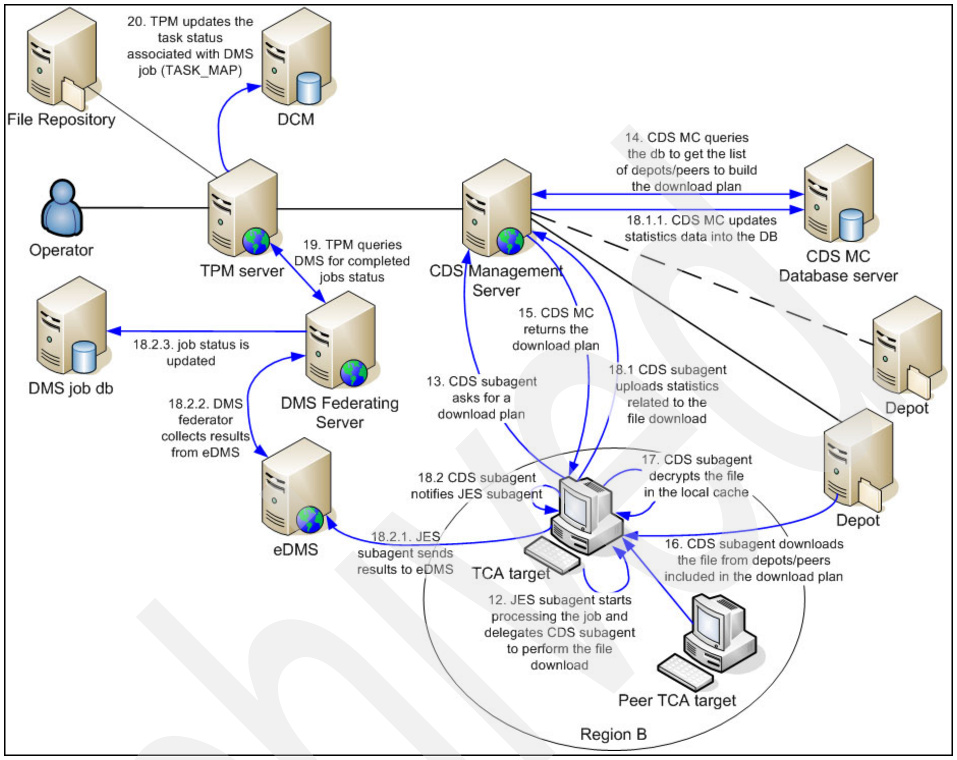

File Repository
20. TPM updates the task status associated with DMS job (TASK_MAP)
DCM
Operator
TPM server
19. TPM queries DMS for completed jobs status
CDS Management Server
14. CDS MC queries the db to get the list of depots/peers to build the download plan
18.1.1. CDS MC updates statistics data into the DB
CDS MC Database server
Depot
DMS job db
18.2.3. job status is updated
DMS Federating Server
18.2.2. DMS federator collects results from eDMS
15. CDS MC returns the download plan
13. CDS subagent asks for a download plan
18.1 CDS subagent uploads statistics related to the file download
Depot
17. CDS subagent decrypts the file in the local cache
18.2 CDS subagent notifies JES subagent
eDMS
18.2.1. JES subagent sends results to eDMS
TCA target
16. CDS subagent downloads the file from depots/peers included in the download plan
12. JES subagent starts processing the job and delegates CDS subagent to perform the file download
Peer TCA target
Region B

Figure 6-5 Distribute operation flow - stage 3

6.2.4 Install

The install operation allows you to install one or more software products to one or more target devices.

- Selecting to install multiple software instances produces the same number of installation requests.
- Selecting computers or groups of computers to specify targets.
- The Install operation for SPB installables allows the following options, which are passed to the SIE subagent and used when installing the package:
  - Dependency check allows you to check package dependencies on the local software catalog.
  - Force allows you to force the installation regardless of the local catalog contents.

Installing on a common agent target

The install on Tivoli common agent targets always implements the following three steps as illustrated in Figure 6-6.

1. Publishing of the software installable file to the dynamic content delivery service infrastructure if not already published, which we describe in detail in the previous section.
2. Distributing of the Software Installable to the target devices, unless the file was already distributed.
   - `LocalFileExist` - checks for the installable existence on the Tivoli common agent local cache.
   - `GetInstallable` - downloads the installable (if missing) from dynamic content delivery service to the local cache.
3. Performing the software installation. The SIE sub-agent installs the software package from the local cache.
   - `InstallInstallable` - installs the installable.

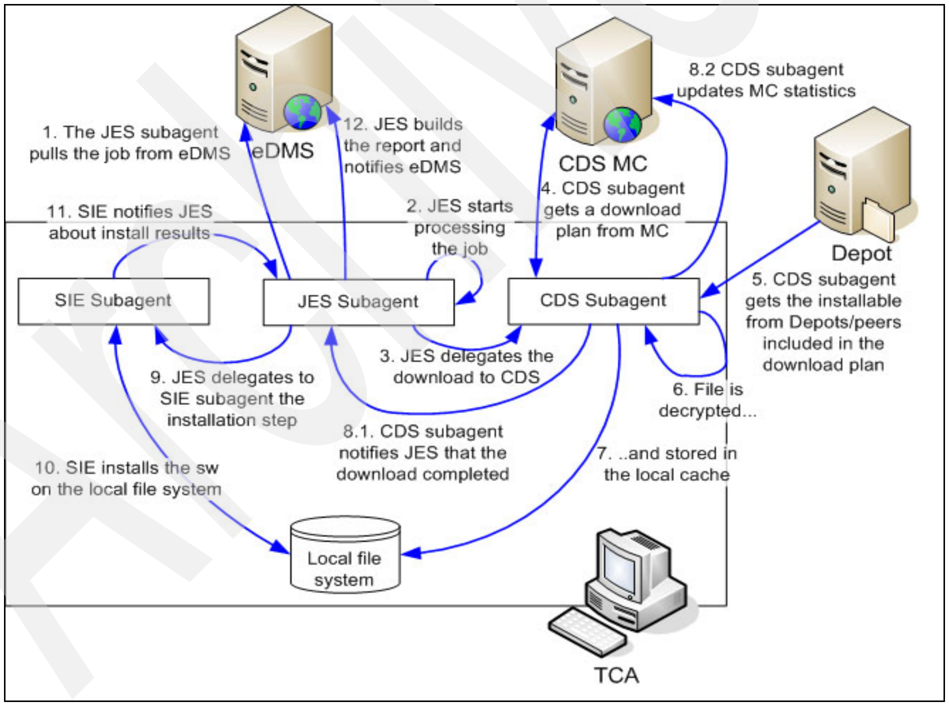

eDMS
CDS MC
Depot
TCA
1. The JES subagent pulls the job from eDMS
12. JES builds the report and notifies eDMS
8.2 CDS subagent updates MC statistics
4. CDS subagent gets a download plan from MC
2. JES starts processing the job
11. SIE notifies JES about install results
SIE Subagent
JES Subagent
CDS Subagent
5. CDS subagent gets the installable from Depots/peers included in the download plan
3. JES delegates the download to CDS
9. JES delegates to SIE subagent the installation step
6. File is decrypted...
8.1. CDS subagent notifies JES that the download completed
7. ..and stored in the local cache
10. SIE installs the sw on the local file system
Local file system

Figure 6-6 Install operation flow

Tivoli Management Agent targets

An install operation on Tivoli Management Framework can be converted to different Tivoli Configuration Manager operations depending on the status of the Software Installable for each target.

Table 6-12 Install on Tivoli Management Framework targets

Software installable status	Resulting Tivoli Configuration Manager operation
No action was performed on the installable	Install
Publish was performed on the installable	Install from depot (not available out-of-the-box)
Distribute (transactional install) was performed on the installable	Commit

Flow chart of the install operation

Figure 6-7 shows a flow chart of the install operation.

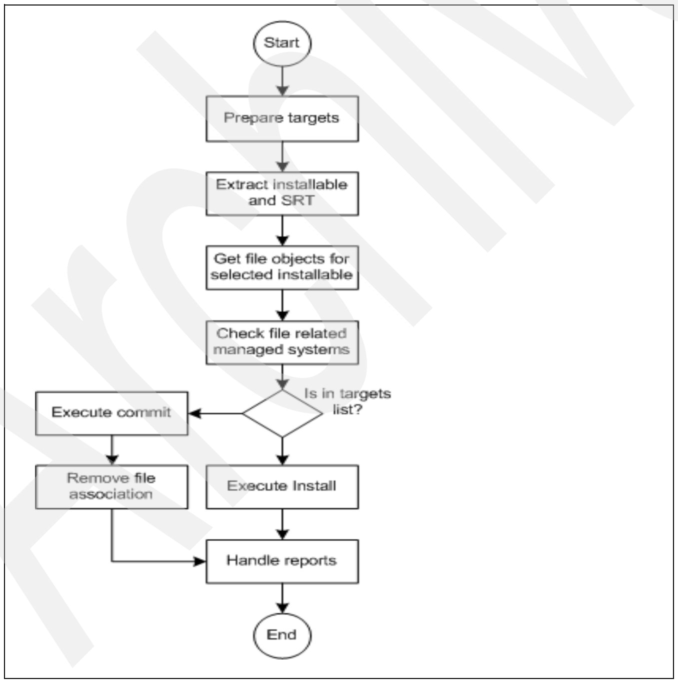


Figure 6-7 Install operation flow chart

6.2.5 Uninstall

The uninstall operation performs an uninstall of one or more software products from one or more target devices.

The uninstall operation for SPB installables allows the following options, which are passed to the SIE subagent and used when uninstalling the package:

- Dependency check allows you to check package dependencies on the local catalog.
- Force allows you to force the uninstallation regardless of the local catalog contents.

Tivoli Management Agent targets

Uninstall on Tivoli Management Framework can be converted in different Tivoli Configuration Manager operations depending on the status of the software installable for each target.

Table 6-13 Uninstall on Tivoli Management Framework targets

Software installable status	Resulting Tivoli Configuration Manager operation
No action was performed on the installable	Remove
Distribute (transactional install) was performed on the installable	Undo

Flow chart of the uninstall operation

Figure 6-8 shows a flow chart of the uninstall operation.

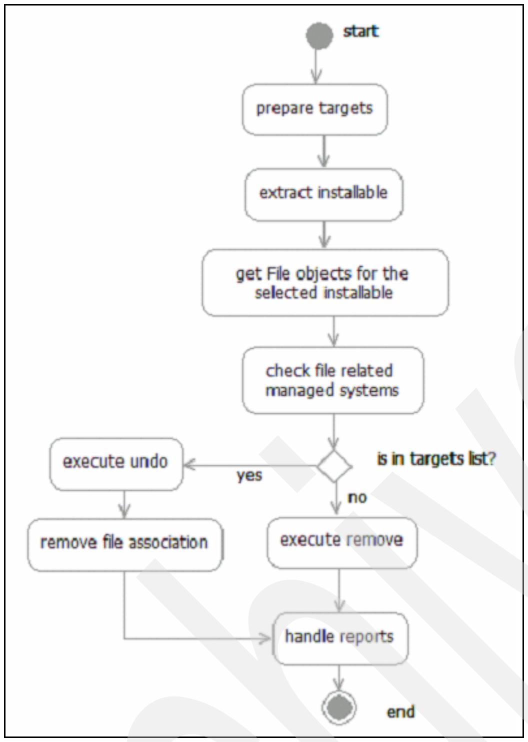


Figure 6-8 Uninstall operation flow chart

6.2.6 Discovery

The discovery operation executes a scan of the hardware and software contents of a computer using Common Inventory Technology (CIT) v 2.2.

Table 6-14 on page 175 shows the resources discovered by a Tivoli Provisioning Manager for Software discovery configuration (grouped by Scan Type) and compares them to the corresponding Tivoli Configuration Manager scan.

Table 6-14   Hardware and software discovery resource comparison

Scan type	Resources	Comparison with Tivoli Configuration Manager
Hardware	► Computer ► OS platform ► Storage ► CD-ROM & floppy ► Memory ► CPU ► Video ► Keyboard ► Mouse ► Printer ► Network ► USB devices ► Modem ► BIOS	When a hardware scan is performed, all of the related resources are discovered. It is not possible to perform a scan for certain hardware resources.
Software	► Registry ► Signatures ► Selected Signatures	Tivoli Configuration Manager file basic and header scan apply only to discovery configurations that are migrated from Tivoli Configuration Manager. Selected signature scan is not supported on Tivoli Management Agent targets.

After integrating with Tivoli Configuration Manager, you can perform a discovery using either new discovery configurations or old discoveries that were created from replicated inventory profiles.

Changing the settings of a replicated discovery configuration modifies the related Tivoli Configuration Manager Inventory profile for the next time it is deployed on the Tivoli Management Framework.

Table 6-15 shows a comparison between the options you can specify in a Tivoli Provisioning Manager for Software discovery configuration and in a Tivoli Configuration Manager Inventory profile.

Table 6-15   Discovery Configuration and Inventory profile comparison

Scan type	Tivoli Provisioning Manager Discovery Configuration	Tivoli Configuration Manager Inventory profile
Hardware	Hardware enabled	Hardware scan (regardless of the resource type selection)

Scan type	Tivoli Provisioning Manager Discovery Configuration	Tivoli Configuration Manager Inventory profile
Software	Registry scan	► Registry scan ► File header info scan ► File basic info scan
	Signature scan	Signature scan
	Selected signature scan	Signature scan with all signatures

Even if the discovery operation is defined as an endpoint operation, when targeting common agent computers it is managed using a workflow:

- `Cit_OnDevice` is the workflow that runs the scan on a common agent computer without leveraging the SOA infrastructure. This workflow remotely accesses the common agent computer and leverages the common agent Web services that are exposed by the common inventory technology (CIT) subagent to run the scan.
  - Results are formatted by the agent in XML format and copied back on the Tivoli Provisioning Manager server, where they are imported into DCM.
  - This is not highly scalable because targets are addressed one by one.
- `Cit_SOA_OnDevice` is the workflow that runs the scan using the SDI. This workflow prepares a device management service job to run the scan on the common agent's agent.
  - Results are sent back to the Tivoli Provisioning Manager for Software server using an HTTP post on a specific servlet. This servlet usually runs on the Tivoli Provisioning Manager for Software server.
  - As the SDI is used, this method allows for much greater scalability than the deployment engine (DE) method.

Cit_OnDevice workflow description

The Cit_OnDevice workflow invokes a few additional workflows to move files to and from the agent, perform the CIT scan, and handle the results.

The workflow responsible for triggering the CIT discovery invokes some Java code, which interacts with the Web services exposed by the CIT subagent on the common agent side, which is illustrated in step 5 of Figure 6-9.

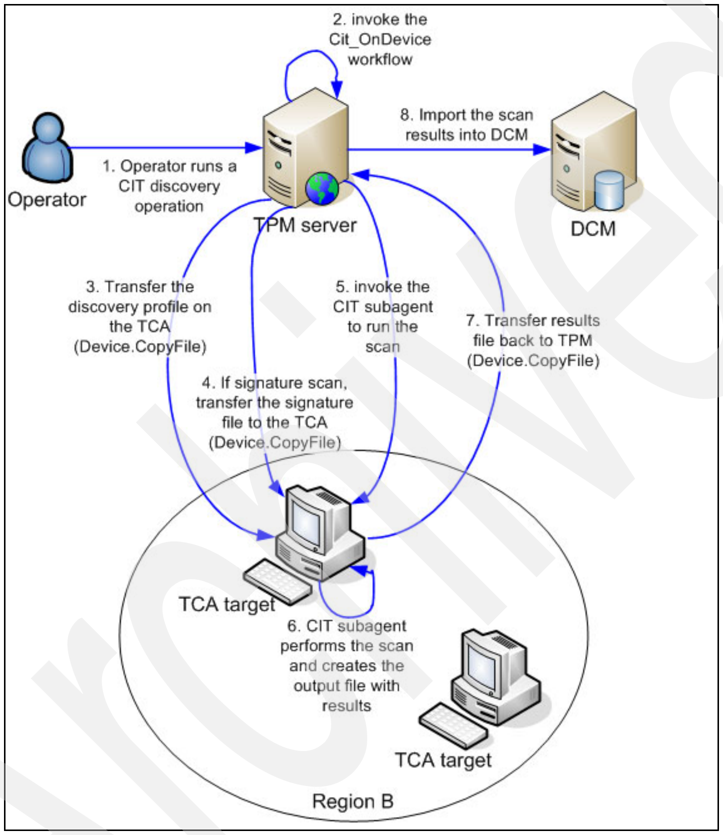

2. invoke the Cit_OnDevice workflow
8. Import the scan results into DCM
Operator
1. Operator runs a CIT discovery operation
TPM server
DCM
3. Transfer the discovery profile on the TCA (Device.CopyFile)
5. invoke the CIT subagent to run the scan
7. Transfer results file back to TPM (Device.CopyFile)
4. If signature scan, transfer the signature file to the TCA (Device.CopyFile)
TCA target
6. CIT subagent performs the scan and creates the output file with results
TCA target
Region B

Figure 6-9 Cit_OnDevice workflow diagram

Cit_SOA_OnDevice workflow description

On the Tivoli Provisioning Manager for Software server side, the Cit_SOA_OnDevice workflow performs the following steps:

- Publishes the signature catalog to dynamic content delivery service if a signature scan is selected, which is illustrated in step 2 in Figure 6-10 on page 178.

- Creates a device management service job with the following work items:
  - Download the signature catalog in case of a signature scan
  - Perform the CIT scan
  - Compress the results
  - Upload the results to Tivoli Provisioning Manager for Software collector servlet.

Figure 6-10 shows the Cit_SOA_OnDevice workflow diagram from Tivoli Provisioning Manager for Software side.

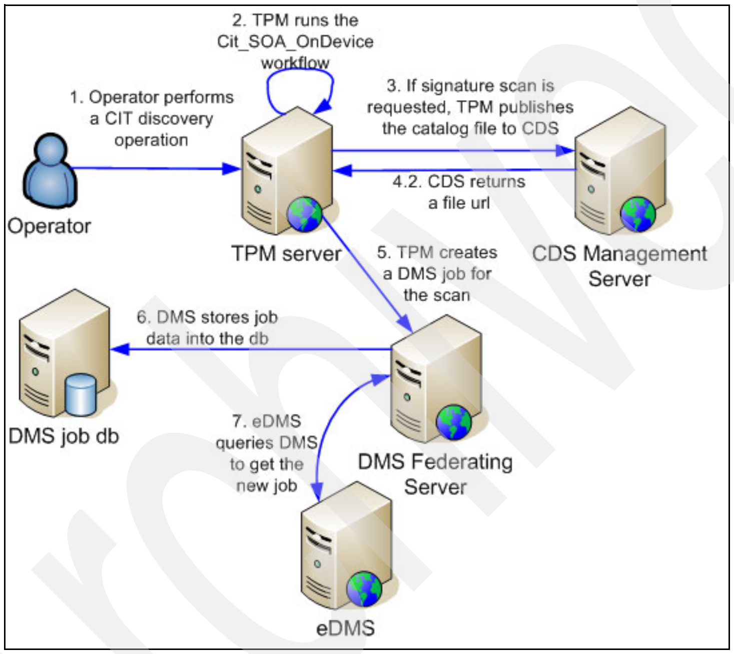

2. TPM runs the Cit_SOA_OnDevice workflow
1. Operator performs a CIT discovery operation
3. If signature scan is requested, TPM publishes the catalog file to CDS
4.2. CDS returns a file url
Operator
TPM server
5. TPM creates a DMS job for the scan
CDS Management Server
6. DMS stores job data into the db
DMS job db
7. eDMS queries DMS to get the new job
DMS Federating Server
eDMS

Figure 6-10 Cit_SOA_OnDevice workflow diagram Tivoli Provisioning Manager for Software side

From the target side, the workflow is a little different, as you can see in Figure 6-11 on page 179.

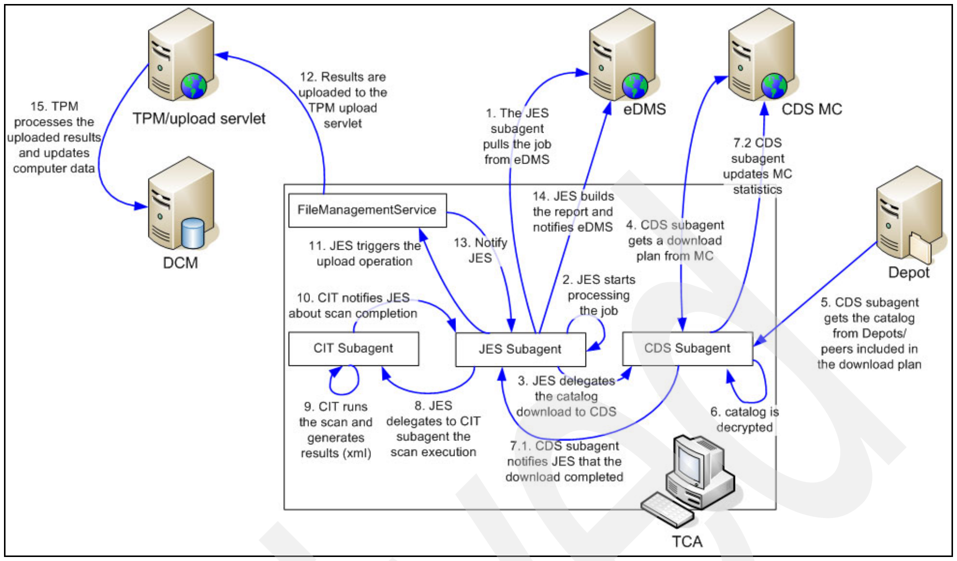

TPM/upload servlet
DCM
eDMS
CDS MC
Depot
TCA
FileManagementService
CIT Subagent
JES Subagent
CDS Subagent
1. The JES subagent pulls the job from eDMS
2. JES starts processing the job
3. JES delegates the catalog download to CDS
4. CDS subagent gets a download plan from MC
5. CDS subagent gets the catalog from Depots/ peers included in the download plan
6. catalog is decrypted
7.1. CDS subagent notifies JES that the download completed
7.2 CDS subagent updates MC statistics
8. JES delegates to CIT subagent the scan execution
9. CIT runs the scan and generates results (xml)
10. CIT notifies JES about scan completion
11. JES triggers the upload operation
12. Results are uploaded to the TPM upload servlet
13. Notify JES
14. JES builds the report and notifies eDMS
15. TPM processes the uploaded results and updates computer data

Figure 6-11 Cit_SOA_OnDevice workflow diagram - target side

On the common agent side, when the agent gets the discovery job from the device management service, it performs the following actions:

- Downloads the signature catalog from the dynamic content delivery service, if a signature scan was requested (steps 3 to 7 in Figure 6-11).
- Performs the CIT scan.
- Compresses the results and uploads to the following URL:

  `https://`*`tpm_server`*`:9045/pobox/`

On the Tivoli Provisioning Manager for Software server, the results are saved into a DCM table.

A set of separate threads is responsible for processing the results and updating the related computer info (step 15 in Figure 6-11).

Tivoli Management Agent targets

A discovery request that originates from the Tivoli Provisioning Manager for Software server is converted into the scan execution of a Tivoli Configuration Manager Inventory profile.

If the Discovery Configuration was created on the Tivoli Provisioning Manager for Software server, then a corresponding Inventory profile is created on the Tivoli Management Framework side in the `TPM_Region` policy region. See 6.3.2, “The TPM_Region policy region” on page 182 for more information.

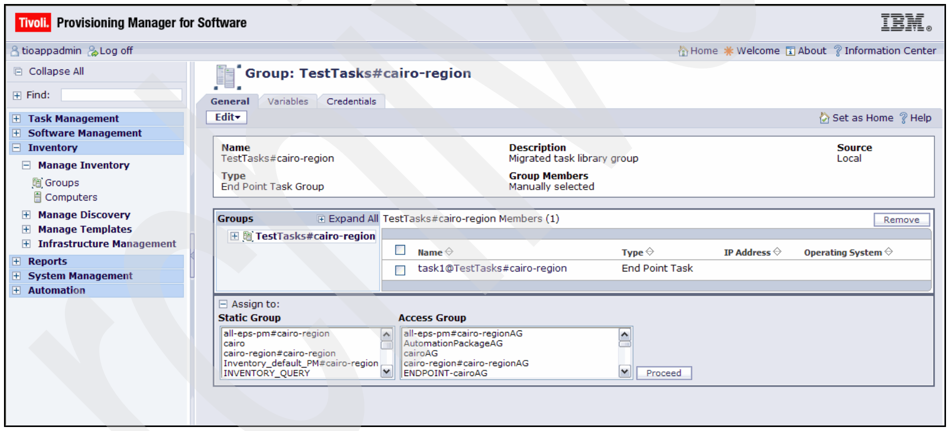
When the scan completes, results are stored into the Tivoli Configuration Manager inventory database, and a selective replication is triggered to update the DCM. The replication addresses all of the discovered resources for the computers that are defined as targets of the discovery operation.

6.2.7 Endpoint tasks

Tivoli Management Framework tasks are mapped into EndPointTask objects when an existing Tivoli Configuration Manager environment is replicated into the Tivoli Provisioning Manager for Software DCM database.

Each EndPointTask object contains the name of the Tivoli Management Framework Task along with the information about files to run for each platform.

EndPointTasks are listed on the Groups page. They are replicated under the TaskLibrary group to which they belong the in Tivoli Management Framework. See Figure 6-12.



Group: TestTasks#cairo-region

Name	Description	Source
TestTasks#cairo-region	Migrated task library group	Local

Type	Group Members
End Point Task Group	Manually selected

TestTasks#cairo-region Members (1)

Name	Type	IP Address	Operating System
task1@TestTasks#cairo-region	End Point Task		

Static Group: all-eps-pm#cairo-region, cairo, cairo-region#cairo-region, Inventory_default_PM#cairo-region, INVENTORY_QUERY

Access Group: all-eps-pm#cairo-regionAG, AutomationPackageAG, cairoAG, cairo-region#cairo-regionAG, ENDPOINT-cairoAG

Figure 6-12 Migrated task library group example from fix pack 2

From the Web User Interface, you can perform the following actions:

- Include an endpoint task into an activity plan instance
- Create new endpoint tasks (fix pack 2 onwards)
- Modify existing endpoint tasks (fix pack 2 onwards)
- Run endpoint tasks without the need of using an activity plan for this action (fix pack 2 onwards)

Common agent targets

All of the files associated with a task are copied to the Tivoli Provisioning Manager for Software infrastructure. The script files are moved from the Tivoli Management Region server to the Local File Repository using the `TMF_FileRepository_GetFile` workflow.

A device management service job is prepared and submitted to:

- Download the files needed by each target to run the task
- Execute the command with the arguments passed by the user

Tivoli Management Agent targets

The original Tivoli Management Framework Task is used when addressing targets that belong to the same or interconnected Tivoli Management Region.

When managing targets belonging to a Tivoli Management Region that is not connected to the source Tivoli Management Region, files are moved from one environment to the other using the Local File Repository as temporary location.

6.3 Driving Tivoli Management Framework from Tivoli Provisioning Manager for Software

This section describes the interaction between Tivoli Provisioning Manager for Software and Tivoli Management Framework when driving an interconnected environment from Tivoli Provisioning Manager for Software.

6.3.1 Object moving and mapping

Several steps are required to drive the Tivoli Management Framework from the Tivoli Provisioning Manager for Software infrastructure. These steps are performed by the replication tools.

- The Tivoli Management Framework resources that need to be associated to a file repository to be managed from Tivoli Provisioning Manager for Software are:
  - Source host for SPBs
  - Tivoli Management Region managed node for binaries associated to a Tivoli Management Framework task
- When performing actions on common agent targets, these resources need to be moved to the Tivoli Provisioning Manager for Software server. The operation executed is `FileRepository.GetFile`.

- When performing actions on disconnected Tivoli Management Regions, these resources need to be moved to the destination Tivoli Management Region.
  - `FileRepository.GetFile` operation moves the resource to a temporary location on the Local File Repository
  - `FileRepository.PutFile` operation moves the resource from the Local File Repository to the destination Tivoli Management Region

An example of the moves for a software package is shown in Figure 6-13.

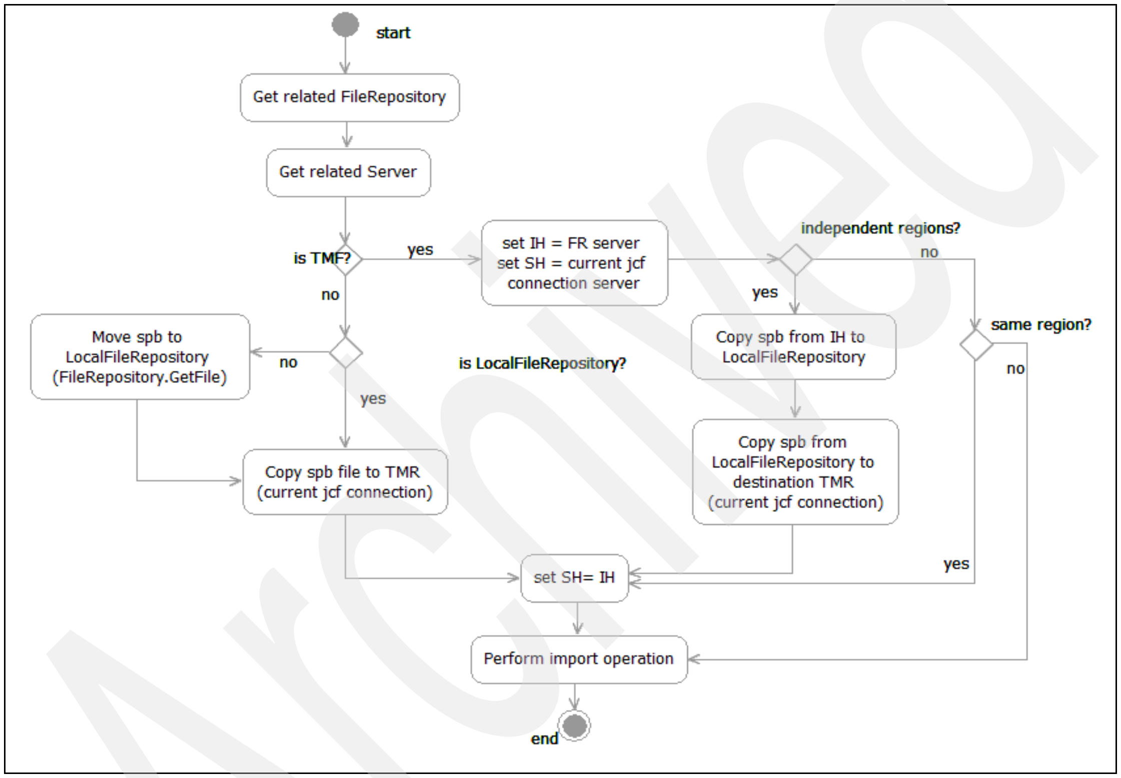

start
Get related FileRepository
Get related Server
is TMF?
yes
no
set IH = FR server
set SH = current jcf connection server
independent regions?
no
yes
Copy spb from IH to LocalFileRepository
same region?
no
Move spb to LocalFileRepository (FileRepository.GetFile)
no
is LocalFileRepository?
yes
Copy spb file to TMR (current jcf connection)
Copy spb from LocalFileRepository to destination TMR (current jcf connection)
yes
set SH= IH
Perform import operation
end

Figure 6-13 Software package example

6.3.2 The TPM_Region policy region

Tivoli Provisioning Manager for Software allows you to create certain new resources (software definitions, discovery configurations, endpoint tasks) that you can use on both TMA and common agent targets.

When targeting TMA endpoints, you must first create those new resources in the Tivoli Management Framework environment before you can use them.

To accomplish this, a Top Level policy region, called `TPM_Region` is created. This policy region can contain these managed resources:

- A profile manager for software packages, called `TPM_Swd_ProfileManager`.
- A profile manager for Inventory Profiles called `TPM_Inv_ProfileManager`.
- A Task Library for Tivoli Management Framework Tasks called `TPM_TaskLibrary`.

The objects are created and populated on first use.

6.3.3 Performing management operations on Tivoli Management Framework using the Tivoli Provisioning Manager for Software user interfaces

Depending on which user interface you use, there are a few things to keep in mind while you perform management operations on Tivoli Management Framework resources. These points are equally valid when using SOAP or CLI commands.

Using the Tivoli Provisioning Manager for Software user interface

When you use the Tivoli Provisioning Manager for Software user interface, the following applies:

- Replicated resources are used as they are on Tivoli Management Framework.
- Newly created resources in Tivoli Provisioning Manager for Software that are also managed by Tivoli Configuration Manager (for example, software, inventory discovery configurations, and endpoint tasks) are usable in both environments.
- Results of the operation are collected on the Tivoli Management Framework side, and then sent back to Tivoli Provisioning Manager for Software.
- Results of a CIT scan are collected on Tivoli Management Framework (inventory), and a selective replica is triggered by Tivoli Provisioning Manager for Software to synchronize the DCM with the Inventory data.

Using the Tivoli Management Framework and Tivoli Configuration Manager user interfaces

When you use the Tivoli Management Framework and the Tivoli Configuration Manager user interfaces, the following applies:

- Results of the operations are collected on the Tivoli Management Framework side, but no interaction with Tivoli Provisioning Manager for Software occurs.
- Changes to the inventory database are reported into the DCM the next time a Tivoli Management Framework discovery is triggered by Tivoli Provisioning Manager for Software.
- Tivoli Configuration Manager does not have any knowledge of Tivoli Provisioning Manager for Software: the bridge between the two infrastructures is unidirectional.

7

Patch management

In this chapter, we discuss the patch management functionality of Tivoli Provisioning Manager for Software. We discuss the following topics:

7.1 Introduction to patch management

Patch management is an integral part of protecting customer services, corporate and customer data, and day-to-day operations. Tivoli Provisioning Manager for Software provides tools to manage patch acquisition, evaluation, and deployment. However, keeping systems up-to-date and compliant with corporate security requirements is complex due to the following main factors:

- Vendors release a large number of updates on a frequent basis.
- Tracking and evaluating security risks and the appropriate software updates is time consuming and requires in-depth knowledge of the installed software, hardware and software dependencies, and corporate requirements.
- Deployment of patches must be managed to reduce downtime or service interruptions.
- Effective auditing and reporting is required to monitor deployment, verify compliance, and troubleshoot errors.

7.1.1 Patch management scenarios

In a dynamic IT environment, several situations require the services of patch management:

- Vendor releases new operating system patches.
- New computers are added to the data center or extended enterprise.
- Existing computers are re-purposed.
- Computers are reconnected to the network after an absence of several days or weeks.

7.1.2 Typical steps for patch management

In this section, we define the steps for patch management:

1. Review the patch bulletin from vendor. This might be for operating system or application vendors.
2. Acquire the patch and test it for site-wide applicability (system admin role). Note that the level to which patches are scrutinized depends on the environment on which the computers are used. Less critical roles may only require computers to have all vendor recommended patches (automatic update). Mission critical systems need to have patches evaluated in a test environment to ensure that they do not break the production system. Different steps are typically performed by different people.

3. Approve patches for groups of computers (software approver role). Patches must be tested and approved before being installed in the production environment.
4. Schedule, distribute, and apply approved patches, and monitor status (software operator role). Distribution can involve considerable planning for a large enterprises. It may involve maintenance windows.
5. Verify the patch installation (software operator role), which includes looking at the computers where patch installation failed, troubleshooting the problems, and reinstalling failed patches.

7.2 Tivoli Provisioning Manager for Software patch management

Tivoli Provisioning Manager for Software provides the ability to manage patches using the scalable software distribution infrastructure. This section describes how, in a Tivoli Configuration Manager and a Tivoli Provisioning Manager for Software coexistence environment, you can perform patch management on Tivoli Management agents using the Tivoli Provisioning Manager for Software user interface.

Tivoli Configuration Manager patch management solution uses:

- Activity plans to deploy patches to the agents.
- Microsoft Windows Server Update Services (WSUS) server to discover and approve patches and to download patch executable files and patch catalogs.
- Tivoli Configuration Manager automation server, version 2.1 to manage patches using patch management workflows.

Tivoli Provisioning Manager for Software patch management solution uses:

- Activity Plans are replaced by the compliance and remediation functionality provided by Tivoli Provisioning Manager for Software.
- Microsoft WSUS server is no longer required because patch executable files and patch catalogs are automatically downloaded from the Microsoft Web site (optionally using a proxy server).
- Tivoli Configuration Manager automation server, Version 2.1 is no longer required because the Tivoli Provisioning Manager for Software user interface manages patches.

The procedure for managing patches on Tivoli Management agents and Tivoli Common Agents from Tivoli Provisioning Manager for Software Web User

Interface depends on whether or not you have an existing Tivoli Configuration Manager patch solution already implemented. The following paragraphs detail the steps involved in each case.

We assume that the patch management solution provided with Tivoli Configuration Manager 4.2.3 fix pack 2 as well as the required interim fixes are up and running when you migrate an existing Tivoli Configuration Manager patch management solution. After you create the coexistence environment described in Chapter 5, “Implementing coexistence” on page 93, you perform patch management on Tivoli management agents and common agents using the Tivoli Provisioning Manager for Software user interface.

We assume that the patch management solution that is provided with Tivoli Configuration Manager is not installed. After you create a coexistence environment, as described in Chapter 5, “Implementing coexistence” on page 93, you perform patch management on Tivoli management agents and common agents using the Tivoli Provisioning Manager for Software user interface.

Note: The client compliance and remediation features of Tivoli Provisioning Manager for Software are only supported on the Tivoli Common Agent. Additionally, the Tivoli management agent (LCF endpoint) only supports the Windows platform.

To migrate the Tivoli Configuration Manager patch management solution, perform the following actions:

1. Dismiss the use of Tivoli Configuration Manager automation server, version 2.1 in the Tivoli Configuration Manager 4.2.3 environment.
2. Dismiss the use of Microsoft WSUS server in the Tivoli Configuration Manager 4.2.3 environment.

Use the Tivoli Provisioning Manager for Software user interface to manage patches. You can find detailed information in the Tivoli Provisioning Manager for Software Information Center, which is at the following Web page:

http://tioid2.torolab.ibm.com:8884/help/index.jsp

To set up the Tivoli Provisioning Manager for Software patch management solution, perform the following actions:

1. Install the Tivoli Configuration Manager Patch Management on the Tivoli Management Region server in your environment. You do not need to install the Tivoli Configuration Manager automation server or the Microsoft WSUS server, as was required in the Tivoli Configuration Manager version 4.2.3 solution. Refer to the *IBM Tivoli Configuration Manager Patch Management Guide*, SC23-5263 for specific details on how to install this component.
2. Create a coexistence environment, as described in Chapter 5, “Implementing coexistence” on page 93.
3. Deploy the Windows Update Agent on the Tivoli management agents (TMA) using the Windows Update Agent software module from the Tivoli Provisioning Manager for Software user interface.

Use the Tivoli Provisioning Manager for Software user interface to manage patches. Refer to the patch management topics in the Tivoli Provisioning Manager for Software Information Center at the following Web page:

`http://tioid2.torolab.ibm.com:8884/help/index.jsp`

Patch management in Tivoli Provisioning Manager for Software is tightly integrated with compliance management. At a high level, patch management is viewed in the same light as other compliance subjects, such as password length and strength or whether an instance of an installed software product (such as, a firewall or anti-virus product), exists on a target. In addition to required compliance in those areas, compliance checks are run to determine whether or not a target has had a particular operating system patch installed. If the target has, that particular compliance check is passed. If the patch is not installed, the compliance check fails and Tivoli Provisioning Manager for Software makes recommendations that the target is added to a list of systems that require remediation.

7.2.1 Recommended architecture

Disk space is the only additional architectural item to consider when you plan a Tivoli Provisioning Manager for Software environment for patch management (as well as for high numbers of large software products). Follow the architecture recommendations for CPU and memory that is listed in the product release notes. Depending on the number of languages required for patches or other software products, the disk-space requirements could vary greatly. We recommend that the administrator review the number and estimated size of patches and products that could be deployed, as well as retention requirements when attempting to determine the amount of disk storage that is required in the Tivoli Provisioning Manager for Software environment.

7.3 Tivoli Provisioning Manager for Software target interpreter support

Tivoli Provisioning Manager for Software with Fix Pack 2 supports patching on the following platforms:

- Microsoft (Windows operating systems and Office products)
- IBM AIX
- Sun Solaris
- Red Hat Linux
- Hewlett Packard HP-UX

7.3.1 Patching Microsoft products

Note: The first part of this section describes the Windows patching method using Tivoli Configuration Manager 4.2.3. Do not confuse this with the methods used with Tivoli Provisioning Manager for Software. We provide the description of the Tivoli Configuration Manager method for comparison and contrast purposes only.

In a migration from Tivoli Configuration Manager 4.2.3 with Patch Management to a Tivoli Provisioning Manager for Software environment, there are changes, many of which can be considered enhancements. The architecture in a Tivoli Configuration Manager 4.2.3 Patch Management environment consists of:

- A Tivoli Management Region Server with Tivoli Configuration Manager 4.2.3 Patch Management installed
- An automation server
- A Microsoft Windows Server with Windows Software Update Service installed

The following paragraphs define the purpose of each system.

The Tivoli Management Region Server acts as the delivery mechanism for the patches, as well as the "decision maker" regarding which target(s) gets which patch(es). Targets for the patch distribution are identified through inventory scans that initiate a Microsoft-provided executable called Windows Update Agent (WUA) on the endpoint and are processed using the activity plan feature of Tivoli

Configuration Manager. The WUA must be installed on the endpoint prior to a patch inventory scan.

The automation server is one of two user interfaces to the patching process. The automation server is installed with an earlier version of Tivoli Provisioning Manager and acts as the workflow engine, which obtains files from go.microsoft.com and moves those files to the Tivoli Management Region Server.

The WSUS Server is the other user interface to patch management. The WSUS Server connects to Microsoft, gets the most recent list of patches, and presents them to the patch administrator in a list. The administrator can drill down to learn more about each patch, the systems for which the patch is designed, and the threat level that Microsoft assigned to the patch. The administrator can accept or reject the patch.

Upon exiting from the WSUS console, an “approved” file is created that is routed to the other machines so that when the inventory and distribution occur, Tivoli Configuration Manager can reconcile the patch list with the list of targets.

When the *Group Status Updater* workflow is run, the approved patches are downloaded from Microsoft, sent to the Tivoli Management Region Server, and turned into software package blocks. Based on queries of the inventory database, Activity Plans are created. The Activity Plans include the targets and take the patch prerequisites into consideration. An additional Microsoft executable allows multiple patches to be sent to the targets with a single reboot at the end, even if an individual patch normally requires a reboot when its installation is complete.

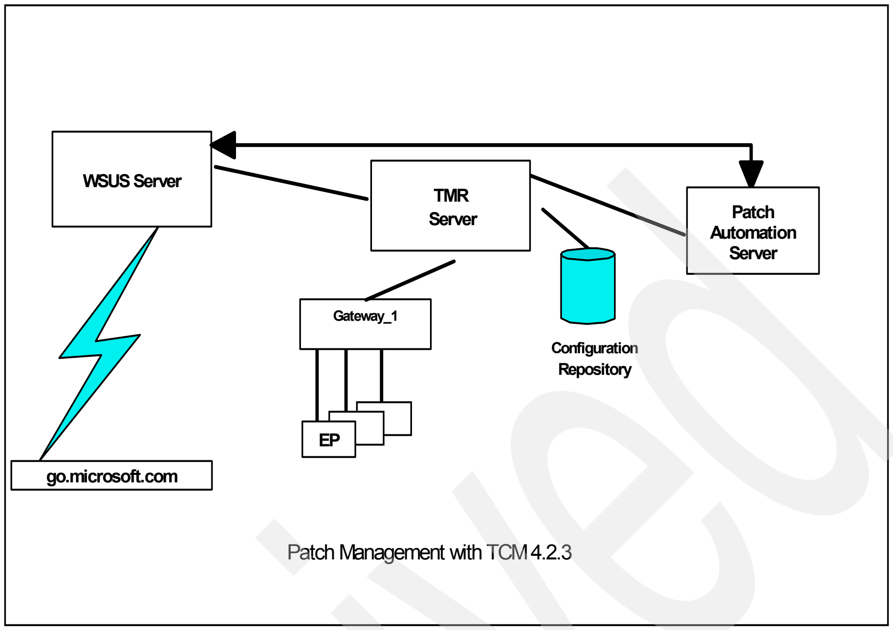


Figure 7-1 Tivoli Configuration Manager 4.2.3 Patch Management architecture

7.4 The differences caused by the migration

The major difference between Tivoli Configuration Manager 4.2.3 Patch Management and Tivoli Provisioning Manager for Software with Fix Pack 2 is that the WSUS server is no longer required (providing the Tivoli Provisioning Manager server is connected to the internet). Additionally, the automation server is deleted because the Tivoli Provisioning Manager for Software Server performs the workflow management.

The architecture in which Tivoli Configuration Manager and Tivoli Provisioning Manager are coexistent appears similar to Figure 7-2 on page 193. Tivoli Provisioning Manager "drives" Tivoli Configuration Manager, and the patches are deployed through the device management service (DMS) infrastructure or by the workflow engine, depending on the type of patch, as discussed earlier in this chapter.

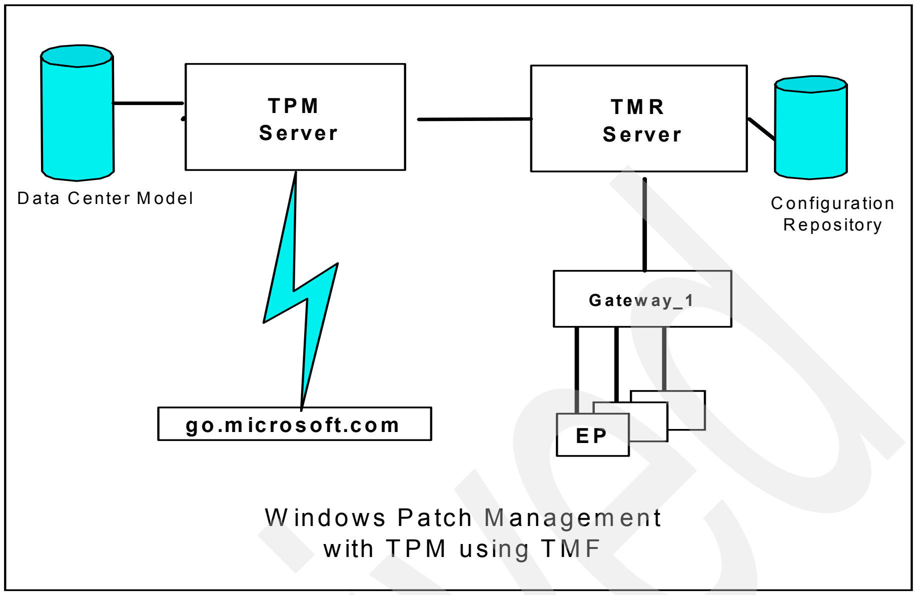


Figure 7-2 Patch management in a Tivoli Configuration Manager-Tivoli Provisioning Manager coexistent environment

In an environment where Tivoli Provisioning Manager is the sole distribution mechanism, the patch management architecture is a lot simpler. It looks similar to Figure 7-3 on page 194.

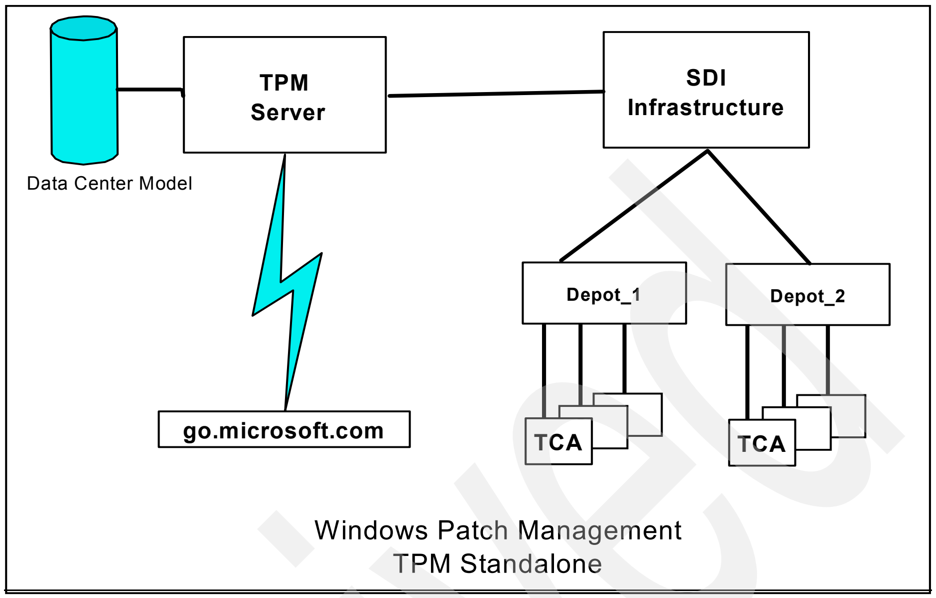


Figure 7-3 Patch management in a Tivoli Provisioning Manager standalone environment

7.4.1 Windows Patch Management in Tivoli Provisioning Manager for Software Fix Pack 2

Patch Management in Fix Pack 2 is integrated with higher-level compliance and remediation functions. It uses groups, discovery, compliance, software distribution, notifications, and reporting.

We assume that the Tivoli Provisioning Manager for Software server is connected to the internet, either directly or through a proxy server. This is a requirement.

Patch identification and download

- The Microsoft Windows' available patch definition is retrieved:
  - A discovery configuration (Microsoft Update Discovery) is provided to automate the download of the Microsoft Windows Update offline scan CAB file and to process its contents to create patch definitions into the DCM.
  - You can schedule discovery to run on a regular basis.
  - No patch executables are downloaded during the first pass, only the metadata (name, severity, description, and so on); however, an additional

workflow is provided to pre-download the patches onto the Windows download server (based on what is in the CAB file). Otherwise, patches are downloaded at install time.

Patch approval process

- Prerequisite steps
  - Microsoft Windows available patches are registered in the DCM
- Tivoli Provisioning Manager Web user interface allows you to select available patches and set their status to:
  - Approved
  - Not Approved
  - Discontinued
- Approved patches define the base set used for detecting missing patches on the target systems
- The approved process is not related to groups of computers:
  - The approval applies to the entire enterprise
  - Single Patch Approver role available

Vulnerability awareness and assessment

- Prerequisite steps:
  - Approval process was performed and at least one patch is in the Approved state
- Tivoli Provisioning Manager Web User Interface allows the patch scanning process to run:
  - A discovery configuration (Microsoft WUA scan) is provided to detect missing patches on a selected set of target systems
  - The discovery configuration downloads the Microsoft Windows Update offline scan CAB file to every target system every time it runs (to manage updated catalog information)
  - The discovery configuration can be scheduled to run at regular intervals on specific groups of computers
  - Results of the scan are collected on the Tivoli Provisioning Manager server, which creates a software stack of patches for every target system

Patch deployment process using Tivoli Provisioning Manager for Software compliance

This section summarizes the patch deployment process using Tivoli Provisioning Manager for Software compliance.

Prerequisite steps

The following tasks are required:

- All available patches are registered in the DCM
- The patch approval process is complete
- The Microsoft WUA scan is performed and missing patches are identified

Tivoli Provisioning Manager Web User Interface gives you the capability of managing compliance status of the enterprise.

You can see which computers require additional patches to comply with security standards.

1. Create a compliance group.

   This is an optional step where you define a group of computers that you plan to use for compliance checks. You can apply compliance checks to single computers as well.

2. Define a compliance check for patch management.

   You need to define a security compliance check for operating system patches and updates. You can define the compliance check for individual computers or on a compliance group.

3. Run the security compliance check.

   The compliance check compares the patches that were discovered on the target computers with the desired level of approved patches; thus, identifying the missing patches and generating the related recommendations. You can schedule this check to run on a specific interval. You can define notification settings to alert one or more administrators.

4. Manage the remediation activities.

   Recommendations and compliance checks can be related to single machines or compliance groups. You must approve a recommendation in order to apply it.

   > **Note:** Each administrator can act at either a group or computer level. Instance level security allows approval recommendations only for the computers owned by a specific administrator.

   You can implement approved recommendations immediately, as well as scheduling them for a later time. Ignore recommendations related to unwanted patches by changing the recommendation status to “closed”. An administrator of a given group might receive unwanted recommendations because the approval process is enterprise wide. A task is created when one or more approved recommendations are implemented.

Figure 7-4 illustrates and describes the compliance flow.

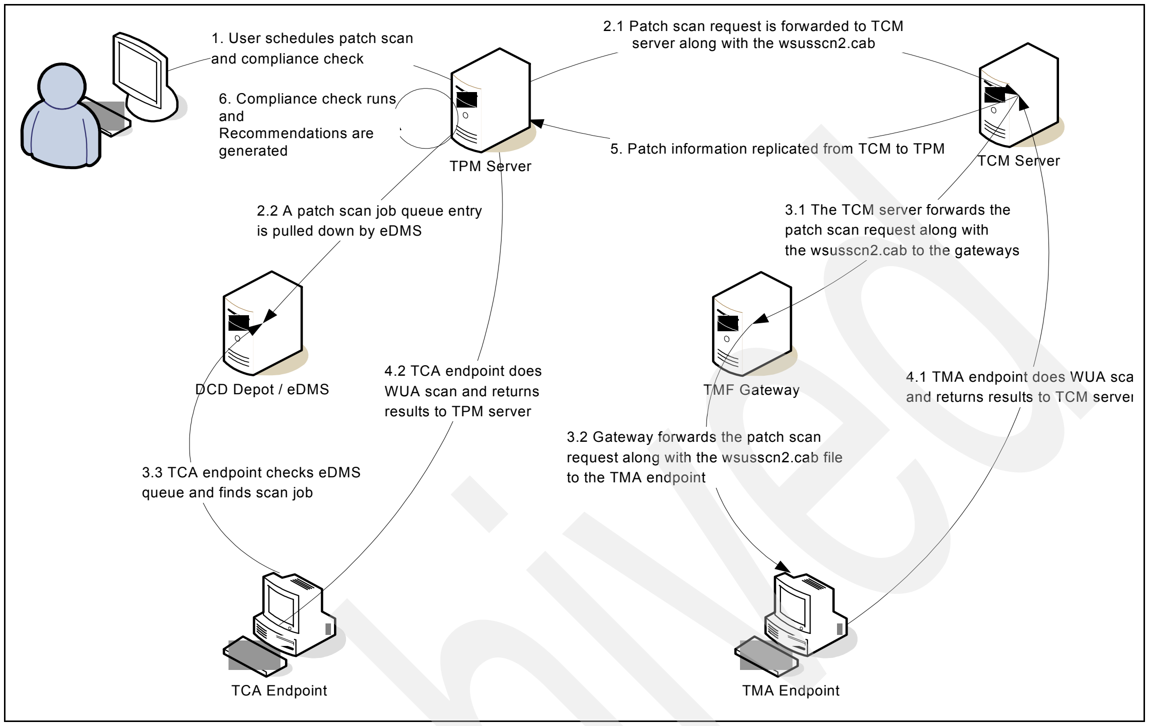
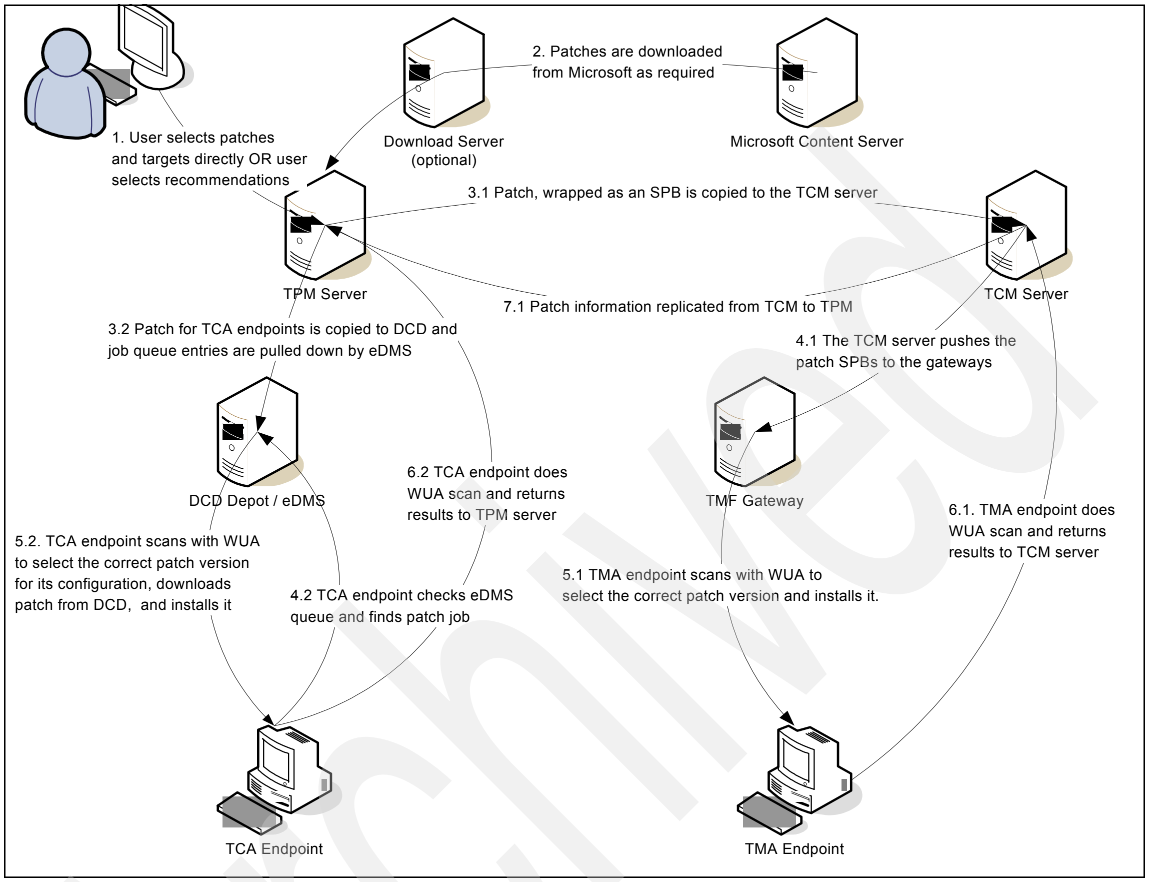

1. User schedules patch scan and compliance check
2.1 Patch scan request is forwarded to TCM server along with the wsusscn2.cab
6. Compliance check runs and Recommendations are generated
TPM Server
5. Patch information replicated from TCM to TPM
TCM Server
2.2 A patch scan job queue entry is pulled down by eDMS
3.1 The TCM server forwards the patch scan request along with the wsusscn2.cab to the gateways
DCD Depot / eDMS
4.2 TCA endpoint does WUA scan and returns results to TPM server
TMF Gateway
4.1 TMA endpoint does WUA sca and returns results to TCM server
3.2 Gateway forwards the patch scan request along with the wsusscn2.cab file to the TMA endpoint
3.3 TCA endpoint checks eDMS queue and finds scan job
TCA Endpoint
TMA Endpoint

Figure 7-4 Recommended compliance flow

The following figure shows the process flow for installing the patches.



1. User selects patches and targets directly OR user selects recommendations
2. Patches are downloaded from Microsoft as required
Download Server (optional)
Microsoft Content Server
3.1 Patch, wrapped as an SPB is copied to the TCM server
TPM Server
TCM Server
7.1 Patch information replicated from TCM to TPM
3.2 Patch for TCA endpoints is copied to DCD and job queue entries are pulled down by eDMS
4.1 The TCM server pushes the patch SPBs to the gateways
DCD Depot / eDMS
6.2 TCA endpoint does WUA scan and returns results to TPM server
TMF Gateway
6.1. TMA endpoint does WUA scan and returns results to TCM server
5.2. TCA endpoint scans with WUA to select the correct patch version for its configuration, downloads patch from DCD, and installs it
4.2 TCA endpoint checks eDMS queue and finds patch job
5.1 TMA endpoint scans with WUA to select the correct patch version and installs it.
TCA Endpoint
TMA Endpoint

Figure 7-5 Recommended installation flow

8

Image management

Deployment and re-deployment of operating systems and applications to bare-metal computers is a key responsibility of IT operations. The reliability, speed, and ease of deployment and re-deployment of computers can have a major impact on minimizing disruptions to the business, reducing IT operations costs, and improving customer satisfaction—particularly in large, distributed computing environments and mission-critical situations.

In this chapter, we provide the guidelines to understand the Image Management Solution evolution from Tivoli Configuration Manager Pristine Manager tool, which is shipped with Tivoli Configuration Manager V4.2.1, to the Tivoli Provisioning Manager V5.1 product family. The Pristine Manager component of Tivoli Configuration Manager V4.2.1 leverages the Microsoft Remote Installation Service (RIS) or Automated Deployment Services Components (ADS) available in your environment to install images on the target machines. Tivoli Provisioning Manager V5.1 includes the Tivoli Provisioning Manager for Operating Systems Deployment Embedded Edition that natively manages images to capture and deploy them from and to Windows and Linux systems. For Tivoli Provisioning Manager for Software V5.1, to get the image management functionality, you can license Tivoli Provisioning Manager for Operating Systems Deployment Embedded Edition separately.

In the last section of this chapter we briefly review and compare the Image Management Solutions available in Tivoli Configuration Manager V4.2.3 and Tivoli Provisioning Manager for Software.

This chapter has the following sections:

8.1 The Tivoli Configuration Manager and Tivoli Provisioning Manager product family

As part of your Tivoli Configuration Manager or Tivoli Provisioning Manager deployment, it is important to understand how each product has evolved to validate what patches, Fix Packs, or enhancements are available on the installation or upgrade you are about to perform. We compare the supported operating systems as servers and clients, installation procedures, architectural description, and differences between Tivoli Configuration Manager V4.2.1 Pristine Manager Component and Tivoli Provisioning Manager V5.1 Image Management feature.

8.2 Supported operating systems

We review the supported operating systems as the server and the target client platform to install the images stored in the Pristine Manager component in Tivoli Configuration Manager V4.2.1 (Images are stored in Microsoft Remote Installation Service or Automated Deployment Services Windows Servers and tools.) and Tivoli Provisioning Manager V5.1 Server.

8.2.1 Tivoli Configuration Manager Version 4.2.1: Pristine Manager Component

With the introduction of IBM Tivoli Configuration Manager V4.2.1, a new component, named Pristine Manager, became available for image management. The Pristine Manager tool 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; instead, Pristine Manager leverages the Automated Deployment Services (ADS) and Remote Installation Service (RIS), which are the two deployment solutions in Windows Server 2003 that Microsoft offers.

Microsoft ADS 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.

Microsoft RIS was first delivered with Windows 2000 and was enhanced in Windows Server 2003 to enable fully automated deployments. Microsoft RIS now supports deployments to servers as well as to desktops.

Important: Of the two solutions, only Microsoft RIS supports deployment of desktops, which is computers targeted to run a Windows client operating system, such as Windows XP. Microsoft ADS is designed and optimized for deployment of servers, computers targeted to run a Windows server operating system, such as Windows Server 2003.

Microsoft RIS Deploys:

- Windows XP: All editions except 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)

While Microsoft ADS deploys:

- Windows Server 2003: All 32-bit versions
- Windows 2000: All Server editions (does not deploy Windows 2000 Professional)

RIS: RIS requires several Windows Operating System components to identify the computers in your network: Dynamic Host Configuration Protocol (DHCP) Server, DNS, and Active Directory. RIS also relies on a unique identifier that each manufacturer assigns to its equipment, a Universal/Global unique Identifier (UUID or GUID). The machines are presented to their RIS Machine using the UUID/GUID.

Pristine Manager Operating Systems support

The Pristine Manager component is installed on a classic framework architecture of an IBM Tivoli Configuration Manager deployment. The following operating systems and databases are supported:

- Server: Same as Tivoli Configuration Manager V4.2.1 server components

- Gateway: Same as Tivoli Configuration Manager V4.2.1 gateway components
- Endpoint:
  - Windows 2000 Server (Microsoft ADS plug-in)
  - Windows 2000 Advanced Server (Microsoft ADS plug-in)
  - Windows 2003 Standard Edition (Microsoft ADS plug-in)
  - Windows 2003 Enterprise Edition (Microsoft ADS plug-in)
  - Windows 2000 Professional (Microsoft RIS plug-in)
  - Windows XP Professional (Microsoft RIS plug-in)
  - GUI (embedded in Activity Planner Monitor & Change Manager GUIs): Same as APM/CM V4.2.1 GUIs
  - Database: Same as Tivoli Configuration Manager V4.2.1 supported RDBMS

Pristine Manager server prerequisites

- Your RIS or ADS server is already installed, configured, and working.
- There is a Tivoli endpoint on the machines where the RIS or ADS server is installed.
- You already created the images on your RIS or ADS server. Refer to the appropriate Microsoft documentation for instructions.
- On the ADS or RIS server, you created a directory with the binary files to install the Tivoli endpoint. The directory must be shared and have full-control permissions.

8.2.2 Tivoli Provisioning Manager Version 5.1: Tivoli Provisioning Manager for Operating System Embedded Edition

Tivoli Provisioning Manager for OS Deployment Embedded Edition is a solution that deploys the operating system on selected target systems or managed systems using the boot strap program known as the Pre-boot Execution Environment (PXE). PXE is one of the Intel® components that allows a managed system to boot from the server in the network rather than booting the operating system from its hard drive. Using the Tivoli Provisioning Manager for Operating System Deployment Embedded Edition technology also enables remote configuration or re-configuration of the remote or target computer.

The Tivoli Provisioning Manager for OS Deployment Embedded Edition software is already copied within when you install a Tivoli Provisioning Manager Server. *You do not need to copy the files manually from the product CD-ROMs or DVDs*.

Tivoli Provisioning Manager Version 5.1 Fix Pack 1

With the introduction of Fix Pack 1, you can install the Tivoli Provisioning Manager for OS Deployment Embedded Edition server on the following platforms:

- Windows 2003, Window XP, or Windows 2000 computer
- RedHat Enterprise Linux (RHEL), version 3 and version 4
- SUSE Linux Enterprise Server (SLES), version 9

Tip: You can install Tivoli Provisioning Manager for OS Deployment Embedded Edition on a Windows 2000 server that has Windows Script Host (WSH), version 5.6. Windows Script Host is available from the Microsoft Web site. After you install Windows Script Host, run the Windows command `cscrip`t to verify the version information. The version must be displayed as Microsoft Windows Script Host Version 5.6.

The Fix Pack 1 includes updates to the Rembo Toolkit, which is now available as Tivoli Provisioning Manager for OS Deployment Embedded Edition. Updates include:

- Tivoli Provisioning Manager for OS Deployment Embedded Edition is globalized.
- Static IP support for imaging.
- Support for 32-bit and 64-bit images.
- Upgrading to Tivoli Provisioning Manager for OS Deployment Embedded Edition.

Tivoli Provisioning Manager Version 5.1 Fix Pack 2

With Fixpack 2, Tivoli Provisioning Manager for Operating System Embedded Edition included in Tivoli Provisioning Manager V5.1 provides the same level of image management operating systems server support as Tivoli Provisioning Manager for OS Deployment Fix Pack 2:

- Windows 2003, Windows XP, or Windows 2000 computer
- RedHat Enterprise Linux (RHEL): version 3 and version 4 (i386™)
- SUSE Linux Enterprise Server (SLES), version 9 and version 10 (i386)
- SUN Solaris version 8, 9, 10 (SPARC)

Note: At the time that we are writing this IBM Redbooks publication, Tivoli Provisioning Manager for OS Deployment Fix Pack 3 was released where AIX 5.3 is supported as Tivoli Provisioning Manager for OS Deployment Server. Currently there is no Fix Pack 3 available for Tivoli Provisioning Manager or Tivoli Provisioning Manager for OS Embedded Edition.

Attention: A fix pack 2 is not available for a Tivoli Provisioning Manager V5.1 Fast Start Installation at the time that we are writing this book.

Tivoli Provisioning Manager Version 5.1 image management client computer prerequisites

In this section, we describe the Tivoli Provisioning Manager V5.1 image management client prerequisites for hardware and software.

Hardware requirements

Remote-boot clients must be equipped with a PXE-compliant bootrom, version 2.00 and above. Most recent computers with on-board network adapters have built-in PXE support. For computers without built-in PXE support, you may consider using a PXE emulation floppy, which is available from various third-party sources. Solaris client computers need to at least have Open Boot PROM version 3.25 or above. SUN provides a patch for upgrading all older SUN Ultra™ machines to Open Boot PROM 3.25.

Other client-computer requirements are:

- Minimal CPU: Pentium® type level.
- Minimal RAM memory: 128 MB.
- VESA compliant (release 2.0 or later) Video BIOS to get high resolution (VGA fallback is always possible in case of incompatibility). However Tivoli Provisioning Manager for Operating System Deployment Embedded Edition Fix Pack2 can also work on headless machines. Either a historical Integrated Drive Electronics (IDE) (with Ultra Direct memory access (DMA) support, if speed is required) or a BIOS-supported hard drive.
- Desktop Management Interface (DMI) support for collecting hardware information, such as model and serial number.
- Tivoli Provisioning Manager for Operating System Deployment Embedded Edition Fix Pack2 also works with VMWare Workstations.

Software requirements

Tivoli Provisioning Manager V5.1 for OS Embedded Edition supports the following operating systems:

- Windows 2000, Windows 2000 Server, Windows 2003 Server
- Windows XP, Windows XP/64
- Windows 98, Millennium
- RedHat Enterprise Linux (RHEL): versions 3, 4
- SUSE Linux Professional: versions 8, 9
- SUSE Linux Enterprise Server (SLES): version 9

Tivoli Provisioning Manager for Operating System Deployment Embedded Edition can natively write files on the Second Extended File system (Ext2) and Third Extended File system (Ext3). This means that the product can create and format Ext2 and Ext3 partitions, and can add, delete, and modify files on these partitions. Cloning is only supported on Ext2 and Ext3 partitions.

8.3 Installation

In this section, we compare the installation methods for both products.

8.3.1 Tivoli Configuration Manager Version 4.2.1: Pristine Manager

You can use the classic Tivoli Framework installation procedure to install the Tivoli Configuration Manager Pristine Manager component.

Note: The InstallShield Multiplatform (ISMP) integrated install (guided installation) does most of the following steps automatically.

1. Use the classic framework installation procedure to install the component.
2. Register Activity Plan Monitor/Change Manager (APM/CCM) plug-ins (**`reg_pristine_apm_plug-in.sh`**, **`reg_pristine_ccm_plug-in.sh`**).
3. Run SQL scripts (pristine_xxx_admin.sql, pristine_xxx_schema.sql).
4. Configure RIM ("pristine" is the name of the RIM).
5. Install gateway components on the gateways to which RIS & ADS endpoints are connected. This enables the product to leverage RIS/ADS services (Pristine Manager Gateway).
6. Install gateway components on those gateways to which the new installed endpoints (on bare-metal machines) will do the first logon (mini-scan).

7. Install an endpoint on every RIS/ADS server that Pristine Manager has to leverage.
8. Configure installation time-outs (wpristine set timeout xx, where xx is the hours time to install the operating system on your machine).

8.3.2 Tivoli Provisioning Manager Version 5.1

The Tivoli Provisioning Manager for OS Deployment Embedded Edition software is part of the Tivoli Provisioning Manager for OS Deployment Embedded Edition automation package. Therefore you can install the Tivoli Provisioning Manager for OS Deployment Embedded Edition software with the Add Boot Server wizard. You do not have to copy the package manually.

We discussed the installation and the specific scenarios to capture and deploy images in "Chapter 9, *Image management*" of the following document: *Deployment Guide Series: IBM Tivoli Provisioning Manager Version V5.1*, SG24-7261.

Tivoli Provisioning Manager for Operating Systems Extended Edition Fix Pack 1 installation

You need to complete the steps in this section if:

- You are using an existing Tivoli Provisioning Manager V5.1 installation before applying the fix pack.
- You have one or more Rembo Toolkit V4.0 boot servers installed using Tivoli Provisioning Manager, and you have images that were captured with the Rembo server.

To upgrade your Rembo Toolkit to Tivoli Provisioning Manager for OS Deployment Embedded Edition:

1. Navigate to **Inventory** → **Manage Inventory** → **Computers**, and find the computer that has the Rembo Toolkit installed.
2. Click the computer name.
3. Click the **Software** tab.
4. Next to Rembo Toolkit 4.0, select **Actions** → **Upgrade**.

The Rembo Toolkit is updated. The software installation now appears as Tivoli Provisioning Manager for Operating System Deployment Embedded Edition.

Tivoli Provisioning Manager for Operating System Embedded Edition Fix Pack 2 installation

In Tivoli Provisioning Manager Version V5.1 GA, Tivoli Provisioning Manager for OS Deployment Embedded Edition was called Rembo Toolkit 4.0.

To upgrade Tivoli Provisioning Manager for OS Deployment Embedded Edition:

1. Click **Inventory** → **Manage Inventory** → **Computers**, and then find the computer that has the Tivoli Provisioning Manager for OS Deployment Embedded Edition server installed.
2. Click the computer name, and select the **Software** tab.
3. Next to Tivoli Provisioning Manager for Operating System Deployment Embedded Edition, click **Actions** → **Upgrade**. The upgrade takes a few moments to complete. Please note that in Tivoli Provisioning Manager Version V5.1 GA, the software name to upgrade is Rembo Toolkit 4.0.

Important: In Tivoli Provisioning Manager V5.1 and Fix Pack 1, the **`sysprep.exe`** and **`setupcl.exe`** files were located in the following directory: %TIO_HOME%\repository\rembo\<win_subdir>, where <win_subdir> is one of the following:

- win2k\32-bit, for Windows 2000 Server 32-bit
- win2k\64-bit, for Windows 2000 Server 64-bit
- win2k3\32-bit, for Windows Server 2003 32-bit
- win2k3\64-bit, for Windows Server 2003 64-bit
- winxp\32-bit, for Windows XP 32-bit
- winxp\64-bit, for Windows XP 64-bit

After you upgrade to Tivoli Provisioning Manager Fix Pack 2, these files are moved to the following directory:

%TIO_HOME%\repository\windows-operating-system\<win_subdir>

Note that If you installed Tivoli Provisioning Manager Fix Pack 1, you must start Tivoli Provisioning Manager at least once before you apply fix pack 2. The first time that you start the product after applying a fix pack, the automation package migration process runs.

This fix pack 2 upgrades to the latest version of Tivoli Provisioning Manager for OS Deployment Embedded Edition. All of your existing operating system images are preserved during the upgrade.

8.4 Architectural description

In this section, we review the main features provided by Tivoli Configuration Manager V4.2.1: Pristine Manager and Tivoli Provisioning Manager for Operating Systems Embedded Edition. We assume that you are familiar with the Tivoli Configuration Manager architecture and terminology.

8.4.1 Tivoli Configuration Manager V4.2.1: Pristine Manager Component

The main features that the Pristine Manager tool provides are:

- Leverage Microsoft RIS/ADS for performing pristine installations
  - PXE based.
  - Common interface defined on the endpoint. Extensible approach.
  - There is one implementation for RIS and one for ADS as dynamic libraries.
  - Images are created using Microsoft RIS and ADS native tools and GUIs.
  - Existing images are manipulated for OS configuration and Endpoint/Tivoli management agents (TMA) installation.
  - No callback mechanism available is on RIS/ADS for OS installation start, progress, and completion events.
  - A pristine operation is considered “completed with success” when the associated TMA/Endpoint has logged on to the Tivoli Management Region.
- Activity Planner Monitor plug-in (APM Plug-in)
  - Performs the actual submission of the Pristine action.
  - It is possible to create a Pristine action directly from APM.
  - APM GUI is extended with menu tools to host the Pristine Manager panels (same panels as the Change Manager GUI).
- Change and Configuration Manager plug-in (CCM Plug-in)
  - New Configuration Element: OSElement.
  - New Subscriber Type: Pristine Target Subscriber.
  - New Subscriber Type: Group Subscriber.
  - Current state is checked on the inventory database.
  - Desired state is expressed with the operating system name and version.
  - Pristine Mode determines how the difference is calculated.
  - Pristine Mode values are: Ignore, Force, If Different, If Newer.

The Change Manager determines if the new operating system must be installed to validate the “Pristine mode” attribute of each machine.

- Force: installs the new operating system with no regard of the current status.
- Ignore: *Does not* install the operating system.
- If different: installs the operating system only if the new operating system is different from the current one.
- If newer: installs the operating system only if the new operating system is a new version of the current one.
- If the pristine mode is “if different” or “if newer”, the operating system is installed only if the CCM plug-in can access the current status for that machine.

– CCM generates a plan for each Pristine Target, where the Pristine action is the first, and all the other actions are conditioned.

– CCM GUI is extended with menu tools to host Pristine Manager panels.

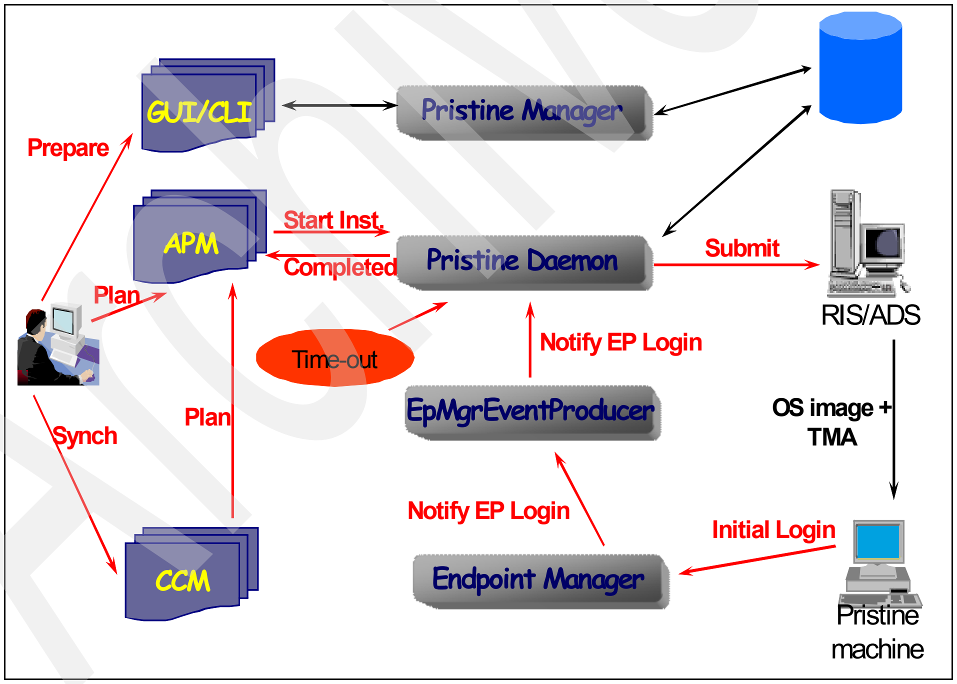


Figure 8-1 Pristine Manager tool Tivoli Configuration Manager V4.2.1 architecture

As shown in Figure 8-1, we prepare the images leveraging the tools (Microsoft RIS / Microsoft ADS) that are integrated through the APM or CCM plug-ins.

Therefore we need to define the Microsoft RIS and ADS Servers, the images that are loaded on them using the OS Element Definitions, and finally the machines where we are going to install the operating system and Tivoli Endpoint.

The Tivoli Management Server (hub, spoke, or just one isolated server) detects the new machine endpoint login in the Tivoli Management Region Server, which in turn, notifies the Activity Planner when the installation is completed. The Pristine daemon considers the operating system to be deployed on the machines after the endpoints login to the gateway that we specify in the machine definition. A Pristine operation is considered “completed with success” when the associated endpoint or Tivoli management agent has logged onto the Tivoli Management Region Server, which is the reason why you must define a time-out at the Pristine daemon level because the Microsoft RIS or ADS servers do not provide any callback method that allows the Pristine daemon to detect the status of the new operating system installation.

You need to define a set of values for each machine that needs to be deployed using the Pristine Manager tool:

- Server: Select the server that will perform the installation. Required.
- Endpoint label: Specify a name for the machine. This becomes the label when the machine becomes a Tivoli endpoint. Required.
- Description: Enter a description.
- MAC address: Enter a 12-character hexadecimal MAC address. Required for installing from an ADS server. For RIS servers, we recommend that you specify this setting because it can be used to verify the success of installation on the pristine machine.
- System Management BIOS (SMBIOS) Globally Unique identifier (GUID): Enter the 32-character hexadecimal System Management BIOS Globally Unique identifier that was assigned to the machine. It can also include dashes. Required for installing from an RIS server.

With the Pristine Manager tool, you need to query the servers (ADS or RIS), so that you can identify and select the images to be deployed. Because the list is being retrieved from the Installation Server, it may take a while to retrieve them.

Note: You can just select one image per ADS or RIS Server.

After you select the image (using the OS Element), you can submit a plan using the Change Manager or the Activity Planner Monitor tool (APM). Be aware that Change Manager generates an activity plan.

8.4.2 Tivoli Provisioning Manager V5.1: Tivoli Provisioning Manager for Operating Systems Extended Edition

Tivoli Provisioning Manager for OS Embedded Edition with Tivoli Provisioning Manager V5.1 provides the following features:

- Tivoli Provisioning Manager Integration:

  The Tivoli Provisioning Manager for OS Embedded Edition integration allows you to manage your images using the Tivoli Provisioning Manager Web interface. However there are certain parameters that you can customize using the Tivoli Provisioning Manager for OS Embedded Edition interface, which you can access at the following Web address:

  `http://tpm:8088`

  In the Web address, tpm is the host name where your Tivoli Provisioning Manager for OS Embedded Edition is installed. To access the Tivoli Provisioning Manager for OS EE Web Interface, use the user name and password that you assigned when you installed Tivoli Provisioning Manager for OS Embedded Edition.

  An example of the available customizations you can make is that you may want to set up the cache partition size. The cache partition is a reserved area of the hard-disk that is used for saving a local copy of disk images that you deploy on computers. If you want to avoid downloading disk images every time you use them, you should have a cache partition big enough to store the compressed disk images.

  You can perform the following operations using the Tivoli Provisioning Manager Web Interface:

  – Image Capture

    Tivoli Provisioning Manager incorporates the ability to natively capture a file-based clone image of a target workstation. Using the Tivoli Provisioning Manager for Operating System Deployment Embedded Edition's built-in Pre-boot eXecution Environment (PXE) server to boot the target system, you can take a cloned image of that system from the Tivoli Provisioning Manager Web console. This image is stored on the Tivoli Provisioning Manager for Operating System Deployment Embedded Edition server, which is integrated within the Tivoli Provisioning Manager Server. You may take a Golden Master image to be used as the reference or a point-in-time snapshot (differences to reference image).

  – Image restoration

    Image restoration is the process of restoring or deploying the clonable image to one or multiple computers. Using Tivoli Provisioning Manager for Operating System Deployment Embedded Edition's built-in Pre-boot eXecution Environment (PXE) server to boot the target system, you can deploy a captured image from a reference machine (gold_master) image using the Tivoli Provisioning Manager Web console.

  - Network sensitive image replication

    The replication of workstation and server images around a WAN is a controversial subject. Many organizations like to have full control over all data on their network, which is why Tivoli Provisioning Manager for OS Deployment Embedded Edition comes with a scheduled, bandwidth-controlled replication. This option allows you to set up a replication schedule between servers and to dictate the maximum bandwidth that can be used by that replication. Therefore, if you want to use the same image on multiple boot servers, you can copy an image from one boot server to another boot server. The image can only be replicated to another boot server of the same type. For example, you can replicate an IBM Tivoli Provisioning Manager for OS Deployment Embedded Edition server to another Tivoli Provisioning Manager for OS Deployment Embedded Edition server. This option is available from your Tivoli Provisioning Manager Web Interface and is useful if you have a large enterprise and want to use multiple boot servers to deploy images more efficiently.

    Note: Set up the credentials for all target systems (Tivoli Provisioning Manager for OS Embedded Edition Servers) in order for this task to complete.

  - Software deployment

    Tivoli Provisioning Manager includes the ability to create software packages that can be deployed along with the operating system image captured with Tivoli Provisioning Manager for Operating Systems Embedded Edition. After you install a captured image on a target computer, you may want to install additional software by creating a software stack. A software stack is a special type of software definition that you can use to identify groups of software that you want to install at the same time and in a specific sequence on target systems. Please note that you can also perform image management if the common agent is included on the image that you capture and install at a later time.

- Tivoli Provisioning Manager Server Operating System Support

  The introduction of fixpack 2 included support to install Tivoli Provisioning Manager for OS Embedded Edition on additional operating systems to the ones supported on Fix Pack 1: SUSE ELS version 10 (i386) and SUN Solaris version 8, 9, 10 (SPARC).

- DHCP and Tivoli Provisioning Manager for OS EE PXE integration

  The PXE server that is available with Tivoli Provisioning Manager for OS Deployment Embedded Edition can run in the two modes specified in the PXE specification: DHCP Proxy mode and Preboot Execution Environment Boot Information Negotiation Layer (BINL) Proxy mode. The proper mode depends on whether the Rembo service is running on the same server as the DHCP service or on a different server.

  The PXE specification defines two ways for a PXE service to cooperate with a standard DHCP server.

  – The first mode is intended to be used when the PXE server (Rembo) is running on the same computer as the DHCP server. The DHCP server must add to its reply the Vendor option 060 set to "PXEClient", and the remote-boot computer will automatically make a BINL request to the same server (using another port) to retrieve boot information. In this case, the Tivoli Provisioning Manager for OS EE server is said to work in BINL proxy mode.

  – The second mode is intended to be used when the PXE server (Tivoli Provisioning Manager for OS EE) is running on a different computer than the DHCP server. In this case, the DHCP server should not add a Vendor option 060 to its reply, and the remote-boot client expects to receive simultaneously a plan DHCP reply from the DHCP server, and a complementary DHCP reply from the Rembo server, acting this time as a DHCP Proxy.

  If you experience problems starting a remote-boot client, ensure that the Tivoli Provisioning Manager for OS Embedded Edition server is running in the proper mode. If it is not the case, it is probably due to the fact that the DHCP port was accidentally in use or free when the Rembo service was started. You may simply try to stop and start the Rembo service to solve the problem.

Figure 8-2 is an illustration of the DHCP and Tivoli Provisioning Manager for OS Extended Edition PXE integration.

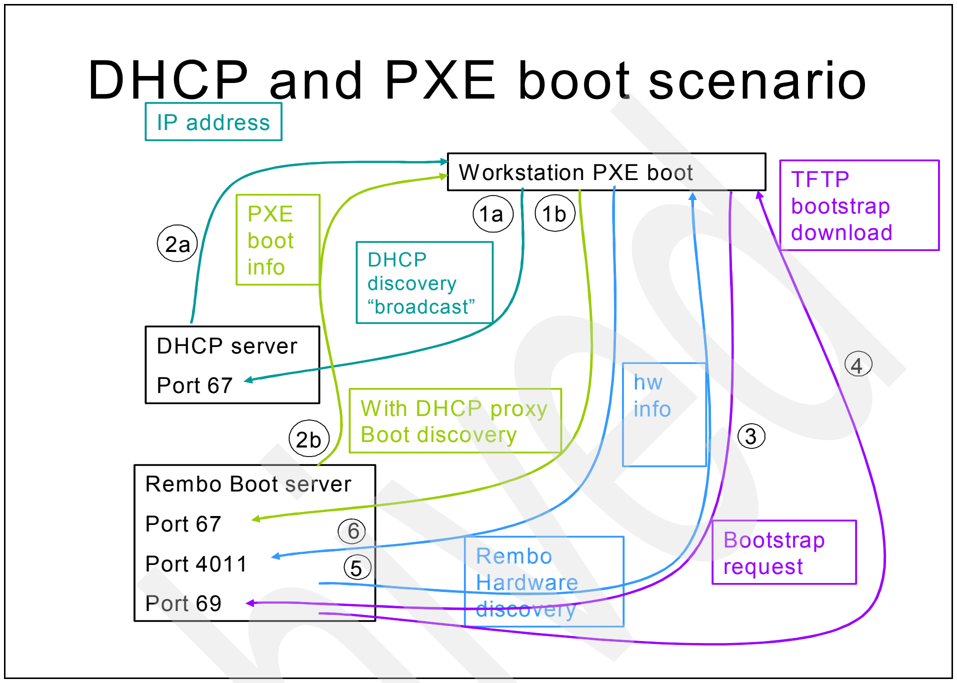


Figure 8-2 Tivoli Provisioning Manager for OS Embedded Edition DHCP/PXE process

The Tivoli Provisioning Manager for OS Embedded Edition uses the PXE programming interface natively, which provides native control of the deployment process. PXE stands for Pre-boot eXecution Environment. Following is the boot sequence, which coincides with Figure 8-2, when you boot using the Rembo Boot Server (Tivoli Provisioning Manager for OS Embedded Edition after applying Tivoli Provisioning Manager Fix Pack 1).

The process when the workstations are booted to deploy or capture an image is:

a. a and b: A user datagram protocol (UDP) discovery broadcast is performed when booting.

b. a and b: Information is sent back by the DHCP server and the Rembo.

c. Bootstrap request from the client.

d. Rembo Boot Server uploads the Rembo Kernel.

e. The boot server request some basic boot information through a "Rembo hardware discover".

f. The client provides the hardware discovered.

By default, the IBM Tivoli Provisioning Manager for OS Deployment Embedded Edition server uses the following ports for communication:

– DHCP (Dynamic Host Configuration Protocol): port 67 UDP
– PXE BINL (Preboot Execution Environment Boot Information Negotiation Layer): port 4011 UDP
– TFTP (Trivial File Transfer Protocol): port 69 UDP
– MTFTP (Multicast Trivial File Transfer Protocol): port 4015 UDP
– NBP (Name Binding Protocol): port 4012 UDP
– FILE: port 4013 UDP and TCP
– MCAST (Multicast Protocol): port 10000-10500 UDP Address: 239.2.0.1 - 239.2.1.1

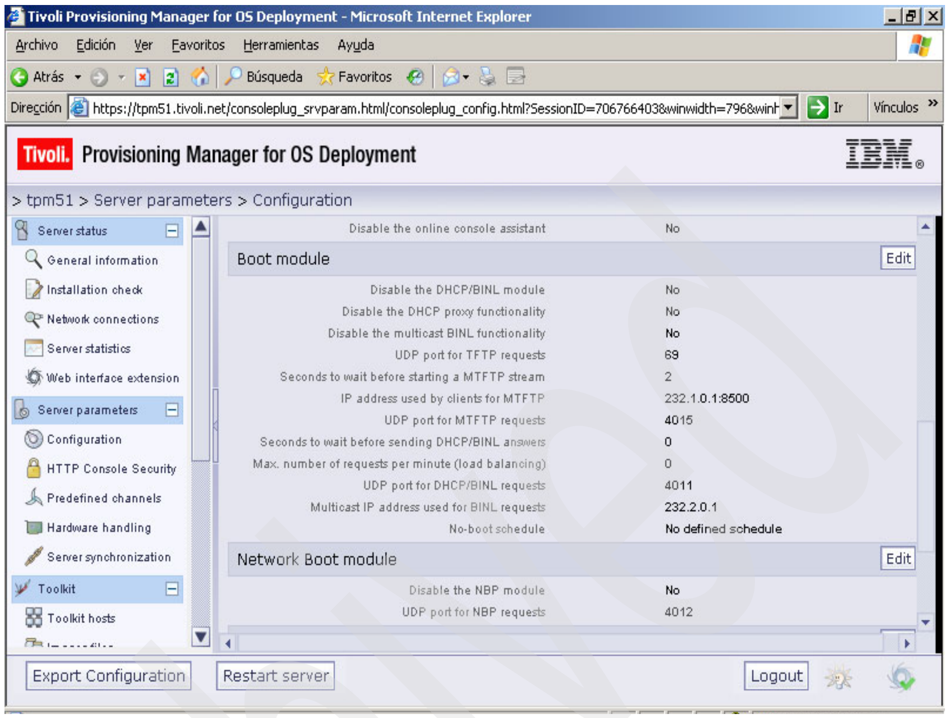
On the clients, the default ports are the following:

– DHCP: port 68 UDP
– MTFTP: port 8500-8510 UDP Address: 232.1.0.1
– MCAST: port 9999 UDP
– Remote control (Web Interface Extension): port 4014 UDP

You can modify all of these ports using the Tivoli Provisioning Manager for OS Embedded Edition Web Interface. There is one notable exception, which is port 69 for TFTP. Port 69 is part of the PXE specifications, which is independent of the Tivoli Provisioning Manager for OS Deployment Embedded Edition product; therefore, it cannot be modified. Any product using the PXE boot needs to have this port open to permit the PXE boot. This port needs to be open only on the Tivoli Provisioning Manager for OS Deployment Embedded Edition server, and not on the client machines.

To modify the ports using the Tivoli Provisioning Manager for OS Embedded Edition Web Interface:

a. Login to the Tivoli Provisioning Manager for OS Deployment Embedded Edition Web interface.
b. Click **Server parameters** → **Configuration**.
c. Select **Edit** to change the ports as needed.


Tivoli Provisioning Manager for OS Deployment - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección https://tpm51.tivoli.net/consoleplug_srvparam.html/consoleplug_config.html?SessionID=7067664038winwidth=7968winh
Ir Vínculos
Tivoli. Provisioning Manager for OS Deployment
IBM
> tpm51 > Server parameters > Configuration
Server status
General information
Installation check
Network connections
Server statistics
Web interface extension
Server parameters
Configuration
HTTP Console Security
Predefined channels
Hardware handling
Server synchronization
Toolkit
Toolkit hosts
Export Configuration
Restart server
Logout

Disable the online console assistant	No

Boot module Edit

Disable the DHCP/BINL module	No
Disable the DHCP proxy functionality	No
Disable the multicast BINL functionality	No
UDP port for TFTP requests	69
Seconds to wait before starting a MTFTP stream	2
IP address used by clients for MTFTP	232.1.0.1:8500
UDP port for MTFTP requests	4015
Seconds to wait before sending DHCP/BINL answers	0
Max. number of requests per minute (load balancing)	0
UDP port for DHCP/BINL requests	4011
Multicast IP address used for BINL requests	232.2.0.1
No-boot schedule	No defined schedule

Network Boot module Edit

Disable the NBP module	No
UDP port for NBP requests	4012

Figure 8-3 Changing configuration ports in Tivoli Provisioning Manager for OS Embedded Edition

Next, we briefly review the evolution from Tivoli Configuration Manager V4.2.1 Pristine Manager tool to IBM Tivoli Provisioning Manager V5.1. We also address the Image Management Solutions available with ITMC V4.2.3 and Tivoli Provisioning Manager for Software V5.1 in the last sections.

8.4.3 Tivoli Provisioning Manager V5.1, Tivoli Provisioning Manager for Software V5.1 and Tivoli Configuration Manager V4.2.3 Fix Pack 2

Support for coexistence was included in Tivoli Provisioning Manager for Software Version V5.1 and was added to Tivoli Provisioning Manager V5.1 Fix Pack 1. When you apply the fix pack, Tivoli Provisioning Manager is updated to version V5.1.0.1.

After applying fix pack 1 on Tivoli Provisioning Manager V5.1, if you are an existing Tivoli Configuration Manager customer, you can continue using the Tivoli Management Framework infrastructure while taking advantage of the capabilities of Tivoli Provisioning Manager. This capability brings your existing Tivoli Management Framework infrastructure within the management of the

Tivoli Provisioning Manager provisioning server. A stepped process is supported so that you can gradually start to work with the Tivoli Provisioning Manager interfaces, while maintaining your Tivoli Management Framework environment. When you are ready, you can make a plan to gradually migrate to a full implementation of Tivoli Provisioning Manager. This is also valid for Tivoli Provisioning Manager V5.1 Fix Pack 2.

Tivoli Configuration Manager V4.2.3 introduces support for the Rembo Auto-Deploy toolkit, which IBM acquired in 2006. The Rembo Auto-Deploy was re-branded as an IBM product called Tivoli Provisioning Manager for Operating System Deployment. Tivoli Configuration Manager V4.2.3 Fix Pack 3 introduced support for Tivoli Provisioning Manager for Operating System Deployment.

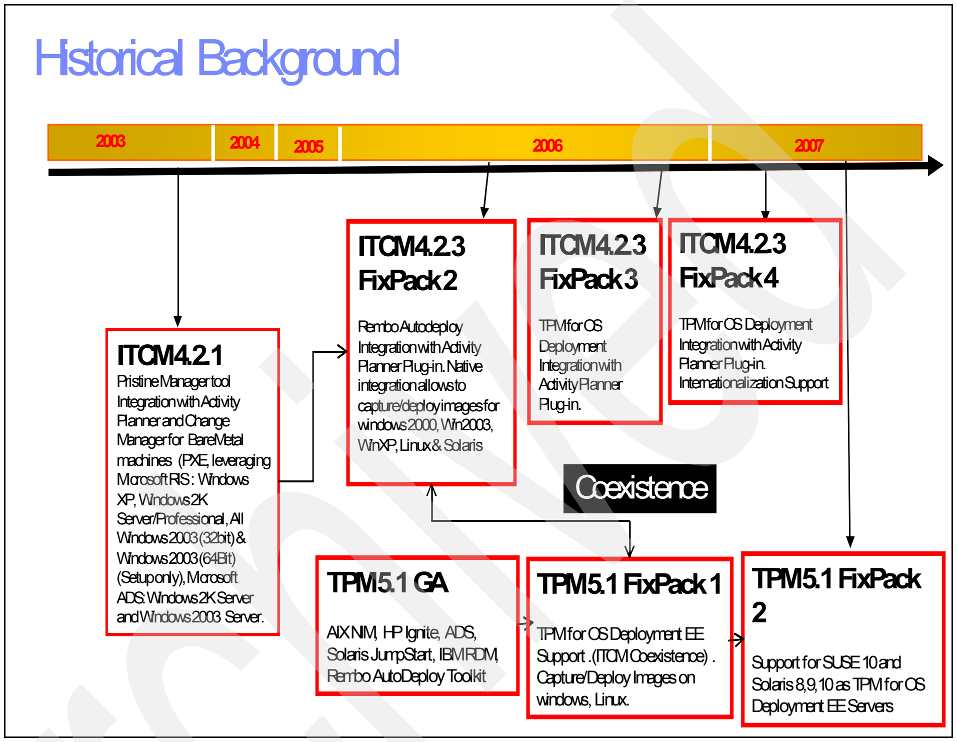
Rembo, when acquired, had another product called Rembo toolkit that was re-branded as Tivoli Provisioning Manager for OS Deployment Embedded Edition.

8.5 Tivoli family evolution

To summarize all of the products and enhancements that are related with image management, which were introduced by the different fix packs that were released up to the time we wrote this book, we can outline the following sequence of events:

- Tivoli Configuration Manager V4.2.1: Pristine Manager tool (leveraging Microsoft RIS & ADS). Control of the Image Deployment (Capture and Deploy) is not performed natively. You capture images using Microsoft RIS or Microsoft ADS tools. As there are no callback methods provided by the Microsoft products, the image deployment is not natively controlled by the Pristine Manager. The tools assumes the image was successfully deployed after the endpoint has login to the Tivoli Management Region Server environment.
- Tivoli Configuration Manager V4.2.3: Fix Pack 2 introduced the Rembo Auto-Deploy Integration. Fix pack 3 introduced the Tivoli Provisioning Manager for OS Deployment integration. Fix pack 4 introduced Internationalization.
- Tivoli Provisioning Manager V5.1: Although Tivoli Provisioning Manager already supports several Image Boot Servers, such as AIX Network Installation Management (NIM), HP Ignite, Microsoft Automated Deployment Server (ADS), Solaris JumpStart™, Linux KickStart, and IBM Remote Deployment Manager (RDM), the introduction of fix pack 1 supports Tivoli Provisioning Manager for OS Deployment Embedded Edition. Fix pack 2 updates Tivoli Provisioning Manager for OS Deployment Embedded Edition within Tivoli Provisioning Manager V5.1 to the latest version.

Tivoli Provisioning Manager for Software Fix Pack 1 introduced the integration with Tivoli Provisioning Manager for OS Deployment Embedded Edition. Fix pack 2 introduced additional support for Tivoli Provisioning Manager for OS Deployment Embedded Edition servers, such as Suse Linux Enterprise server 10 and Sun Solaris 8, 9, and 10 (Sparc).



Historical Background

ITCM4.2.1
Pristine Manager tool Integration with Activity Planner and Change Manager for BareMetal machines (PXE, leveraging Microsoft RIS : Windows XP, Windows 2K Server/Professional, All Windows 2003 (32bit) & Windows 2003 (64Bit) (Setup only), Microsoft ADS: Windows 2K Server and Windows 2003 Server.

ITCM 4.2.3 FixPack 2
Rembo Autodeploy Integration with Activity Planner Plug-in. Native integration allows to capture/deploy images for windows 2000, Win2003, WinXP, Linux & Solaris

ITCM 4.2.3 FixPack 3
TPM for OS Deployment Integration with Activity Planner Plug-in.

ITCM 4.2.3 FixPack 4
TPM for OS Deployment Integration with Activity Planner Plug-in. Internationalization Support

TPM 5.1 GA
AIX NIM, HP Ignite, ADS, Solaris JumpStart, IBM RDM, Rembo AutoDeploy Toolkit

TPM 5.1 FixPack 1
TPM for OS Deployment EE Support . (ITCM Coexistence) . Capture/Deploy Images on windows, Linux.

TPM 5.1 FixPack 2
Support for SUSE 10 and Solaris 8,9,10 as TPM for OS Deployment EE Servers

Figure 8-4 Tivoli Configuration Manager to Tivoli Provisioning Manager Family image management view

8.6 Image management features comparison

In this section we compare the image management features of Tivoli Configuration Manager V4.2.1 and Tivoli Provisioning Manager V5.1.

Table 8-1 Image management functionality comparison: Pristine Manager (Tivoli Configuration Manager V4.2.1) versus Tivoli Provisioning Manager V5.1

Feature	**Tivoli Configuration Manager V4.2.1 Pristine Manager**	**Tivoli Provisioning Manager V5.1**
Image management	Query Images are available on Microsoft RIS or ADS Servers. Just one image is available to select per RIS or ADS server to deploy.	Native Integration via Tivoli Provisioning Manager Web Interface. Additional parameters via Tivoli Provisioning Manager for OS EE Web Interface.
Operating system server	Microsoft Windows Servers servers supported by RIS /ADS to capture and deploy images. You can install the tool on all Tivoli Configuration Manager V4.2.1 supported Servers (Tier 1).	You can install Tivoli Provisioning Manager for OS EE on Windows 2003, Windows XP, or Windows 2000, RedHat Enterprise Linux (RHEL): version 3 and version 4 (i386), SUSE Linux Enterprise Server (SLES), version 9 and version 10 (i386), and SUN Solaris version 8, 9, 10 (SPARC).
Image capture	No. Need to use Microsoft RIS or ADS tools.	Yes. Fully integrated and managed from Tivoli Provisioning Manager Web Interface via PXE.
Image restoration	No native integration. Need to leverage Microsoft RIS /ADS via Endpoint and Activity Planner Plug-in.	Yes. Fully integrated and managed from Tivoli Provisioning Manager Web Interface.
Image replication	Not available from the Pristine Manager tool.	Yes. Fully integrated and managed from Tivoli Provisioning Manager Web Interface.
Native Pristine Operating Systems supported to deploy	Supported Windows Operating Systems via Microsoft RIS or ADS	Windows 2000, Windows 2000 Server, Windows 2003 Server Windows XP, Windows XP/64 Windows 98, Millennium RedHat Enterprise Linux (RHEL): versions 3, 4 SUSE Linux Professional: versions 8, 9 SUSE Linux Enterprise Server (SLES): version 9

8.7 Tivoli Configuration Manager V4.2.3

In 2006, IBM purchased a Swiss company, which was headquartered in Geneva, called Rembo Technology SaRL. They specialized in image management. They had two main products: the Rembo Toolkit Version and Rembo Auto-Deploy.

We used the Rembo Toolkit to allow Tivoli Provisioning Manager V5.1 to perform image management natively. We re-branded the Rembo Auto-Deploy as an IBM product called Tivoli Provisioning Manager for Operating System Deployment. The Rembo Toolkit is also named Tivoli Provisioning Manager for OS Deployment Embedded Edition.

Image management using the Rembo Boot Server supports two main operations: Image capture and image restoration. Image capture is the process of creating a clonable image from a master computer. Image restoration is the process of restoring or deploying the clonable image to one or multiple computers.

8.7.1 Tivoli Configuration Manager V4.2.3 Fix Pack 2

In June 2006, IBM released fix pack 2, which supports the Image management services integration—integration of Rembo Auto-deploy, build level 4.0.034.16, into Configuration Manager to provide services for deploying operation system images in Windows environments.

Rembo Auto-deploy is a database-driven PXE-based deployment solution for Windows, Linux, and Solaris. Rembo Auto-deploy is a comprehensive management solution that significantly increases an IT manager's ability to remotely manage desktop and notebook computers. Using industry standards, such as Wake On LAN, PXE, ODBC/JDBC™, DMI and PCI detection, Microsoft system preparation tool, kickstart, and autoyast, Rembo Auto-Deploy provides out-of-the-box installation of operating systems and selected software on tens, or even hundreds, of computers simultaneously.

Read Chapter 8 "*Integration and Collaboration with other change management products*" in the IBM Redbooks publication *Deployment guide Series: Tivoli Provisioning Manager for OS Deployment V5.1*, SG247397, for scenarios and a description on installing and using Rembo Auto-deploy on IBM Tivoli Configuration Manager V4.2.3 Fix Pack 2. You will notice that Rembo Auto-Deploy is integrated through an Activity Planner plug-in. This plug-in is installed using the classic framework installation, which in this case is called *ITCM-REMBO Integration, version 4.2.3*.

8.7.2 Tivoli Configuration Manager V4.2.3 Fix Pack 3

In December 2006, IBM released fix pack 3, which introduced the feature coded 58233—the Tivoli Provisioning Manager for Operating System Deployment integration. This fix pack provides the integration with the product Tivoli Provisioning Manager for Operating System Deployment Version V5.1 through the new component Tivoli Provisioning Manager for Operating System Deployment integration Version V4.2.3.

The Tivoli Provisioning Manager for OS Deployment features were available starting from Tivoli Configuration Manager V4.2.3 Fix Pack 2, which was shipped with the integration plug-in (called the Rembo plug-in). We strongly suggest that you use Tivoli Provisioning Manager for OS Deployment to avoid some known issues.

Only the English version was available with fix pack 3 because the localized versions were going to be available in the next fix pack. This integration replaced the old Image Management Services Integration, V4.2.3, which was released with fix pack 2, that you must uninstall before installing the Tivoli Provisioning Manager for Operating System Deployment integration. For more details, refer to the IBM Tivoli Configuration Manager: User's Guide for Operating System Deployment Solution. You need to uninstall the previous plug-in before installing Fix Pack 3 plug-in. This plug-in is installed using the classic framework installation, which, in this case, is called *Tivoli Provisioning Manager for Operating System Deployment Integration, version 4.2.3*.

Tip: You might want to store Tivoli Provisioning Manager for Operating System Deployment files on a large hard disk if you plan to create many hard-disk images, and you might want to use a fast CPU if you want to minimize the time you spend creating these images. The Tivoli Provisioning Manager for Operating System Deployment server is multi threaded and therefore greatly benefits from servers with multiple CPUs.

8.7.3 Tivoli Configuration Manager V4.2.3 Fix Pack 4

By the end of March 2007, IBM released Tivoli Configuration Manager V4.2.3 Fix Pack 4 that featured Tivoli Provisioning Manager for Operating System Deployment internationalized. With this feature, the Tivoli Provisioning Manager for Operating System Deployment product was internationalized.

To enable this internationalized feature, install SWD_L10N under the /images path. Also, install the Tivoli_APM_GUI_L10N_Fix.v4.2.3.FP04.spb software package block under the /packages path.

8.7.4 Tivoli Provisioning Manager for Software V5.1

The image management feature is an optional component of Tivoli Provisioning Manager for Software that requires a separate product, IBM Tivoli Provisioning Manager for OS Deployment Embedded Edition (formerly known as Rembo Toolkit 4.0.). You can order Tivoli Provisioning Manager for OS Deployment Embedded Edition as Tivoli Provisioning Manager OS Deployment Component.

The Tivoli Provisioning Manager for OS Deployment Embedded Edition software is part of the Tivoli Provisioning Manager for OS Deployment Embedded Edition automation package file, rembo.tcdriver. Use this to install the Tivoli Provisioning Manager for OS Deployment Embedded Edition software with the Add Boot Server wizard.

You can obtain the automation package by ordering the separate product, Tivoli Provisioning Manager for OS Deployment Embedded Edition. You can order Tivoli Provisioning Manager for OS Deployment Embedded Edition as a Tivoli Provisioning Manager OS Deployment component through your IBM sales representative. The instructions for installing the automation package are provided with the CD-ROM, or you can view them in the prerequisites section of the Managing operating systems tutorial.

8.7.5 Tivoli Provisioning Manager for Software Fix Pack 1

You can install the Tivoli Provisioning Manager for OS Deployment Embedded Edition server on the following platforms:

- Windows 2003, Windows XP, or Windows 2000 computer
- RedHat Enterprise Linux (RHEL), version 3 and version 4
- SUSE Linux Enterprise Server (SLES), version 9

Tip: You can install Tivoli Provisioning Manager for OS Deployment Embedded Edition on a Windows 2000 server that has Windows Script Host (WSH), version 5.6. Windows Script Host is available from the Microsoft Web site. After you install Windows Script Host, run the Windows command `cscript` to verify the version information. The version must be displayed as Microsoft Windows Script Host Version 5.6.

Note: For Linux systems, imaging is only supported for second-extended file system (Ext2) and third-extended file system (Ext3). Currently, there is no support for ResiserFS and Logical Volume Manager (LVM) file systems.

You can install Tivoli Provisioning Manager for OS Deployment Embedded Edition on a Windows Tivoli Provisioning Manager for Software server.

Important: Please note that when you add a boot server in Tivoli Provisioning Manager for Software, only the Rembo boot server type installs and configures the IBM Tivoli Provisioning Manager for OS Deployment Embedded Edition server.

8.7.6 Tivoli Provisioning Manager for Software V5.1 Fix Pack 2

In June 2007, IBM released Tivoli Provisioning Manager for Software Fix Pack 2 that provides an upgrade to Tivoli Provisioning Manager Extended Edition.

All of your existing operating system images are preserved during the upgrade.

You can install the Tivoli Provisioning Manager for OS Deployment Embedded Edition server on additional platforms:

- SUSE Linux Enterprise Server (SLES), version 10
- Solaris 8, Solaris 9, Solaris 10 (Sparc)

9

Automating Tivoli Provisioning Manager for Software tasks using the SOAP CLI

In this chapter, we provide an introduction into automating common software deployment tasks using the SOAP services and command line interface (CLI) available within Tivoli Provisioning Manager for Software.

You can find the full reference guide for all the commands referred to in this chapter in the Coexistence and Migration Considerationsand IBM Tivoli Provisioning Manager online Information center, which you can access at the following Web site:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v13r1/index.jsp`

We provide an overview of the SOAP CLI by discussing the following topics:

- "SOAP overview" on page 226
- "Location of SOAP client" on page 227
- "Examples of using SOAP" on page 230
- "Tivoli Provisioning Manager for Software logical device operations" on page 239
- "Tivoli Provisioning Manager for Software scripts" on page 246

We also provide an end-to-end provisioning scenario. You perform each of the following tasks on the command line, in Tivoli Configuration Manager and Tivoli Provisioning Manager for Software:

- Creating a deployment package
- Importing a deployment package
- Distributing a deployment package
- Installing a deployment package
- Uninstalling a deployment package

9.1 SOAP overview

SOAP is a platform-neutral protocol, which consists of a network, transport, and programming services that allows a client to call a remote service. All messages sent using a SOAP client are in XML format. SOAP is the chosen XML messaging protocol for many reasons:

- It is the standardized enveloping mechanism for communicating document-centric messages and remote procedure calls using XML. The term *envelope*, in this context, means a group of XML components that are wrapped together and sent as a single message. The envelope has a defined structure: a message header, and a location for an XML document (also known as the *payload*).
- A SOAP message is similar to an HTTP POST message with an XML envelope as payload.

SOAP is a specification for the exchange of structured information in a decentralized, distributed environment, and represents the main way of communication between the three main services in SOA:

- The service provider
- The service requestor
- The service broker

Many Tivoli Provisioning Manager for Software internal properties and methods are available through SOAP commands.

Some services enable you to perform data center administration tasks, such as changing the operating mode. Other commands provide access to logical device operations for making configuration changes to data center assets.

To run a particular SOAP command, you must have the required security role and access permissions. You can use SOAP commands to manage your data center from the command line. There are two main categories of SOAP commands:

- For data center administration

  These commands enable you to perform tasks, such as changing operating modes, approving and declining recommendations, and managing failed servers.

- For logical device operations

  These commands invoke workflows from the command line or from client applications.

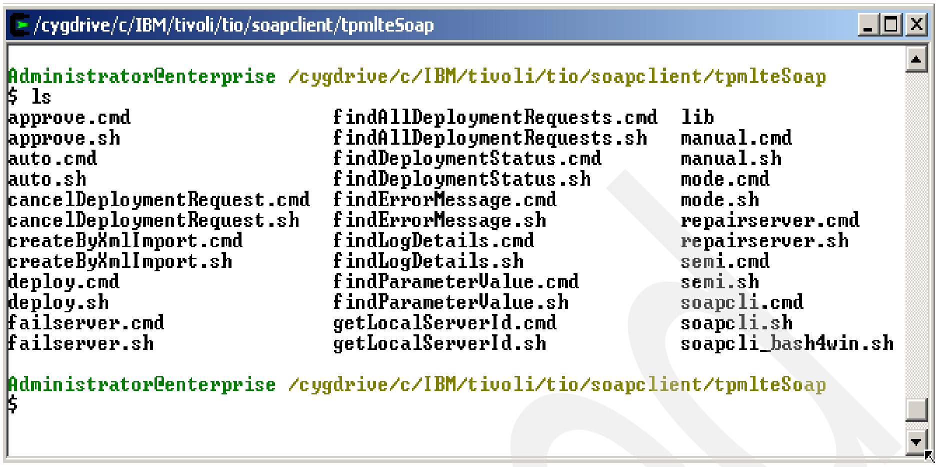
9.1.1 Using SOAP

You invoke SOAP commands from the command line. They are useful for integrating and automating processes that are managed by an external management system. For example, you can create a SOAP script that is invoked when specific events occur.

In another example, consider an IBM Tivoli Monitoring solution that monitors a cluster of Web servers, which are being used as a home shopping Web site. If one Web server in this cluster develops a performance problem and stops responding, then IBM Tivoli Monitoring detects the problem because it actively monitors the servers in the cluster. IBM Tivoli Monitoring can then invoke a SOAP command that sends a message to and IBM Tivoli Provisioning Manager. This message causes and IBM Tivoli Provisioning Manager to start a workflow to provision a new Web server for the Web cluster, hence minimizing the impact of the server that stopped responding.

9.2 Location of SOAP client

As shown in Figure 9-1 on page 228, the SOAP client is located in the <*Installation directory*>/soapclient/tpmLteSoap/ directory, for example, C:/ibm/tivoli/tio/soapclient/tpmLtsSoap.



```
/cygdrive/c/IBM/tivoli/tio/soapclient/tpmlteSoap

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/soapclient/tpmlteSoap
$ ls
approve.cmd                    findAllDeploymentRequests.cmd  lib
approve.sh                     findAllDeploymentRequests.sh   manual.cmd
auto.cmd                       findDeploymentStatus.cmd       manual.sh
auto.sh                        findDeploymentStatus.sh        mode.cmd
cancelDeploymentRequest.cmd    findErrorMessage.cmd           mode.sh
cancelDeploymentRequest.sh     findErrorMessage.sh            repairserver.cmd
createByXmlImport.cmd          findLogDetails.cmd             repairserver.sh
createByXmlImport.sh           findLogDetails.sh              semi.cmd
deploy.cmd                     findParameterValue.cmd         semi.sh
deploy.sh                      findParameterValue.sh          soapcli.cmd
failserver.cmd                 getLocalServerId.cmd           soapcli.sh
failserver.sh                  getLocalServerId.sh            soapcli_bash4win.sh

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/soapclient/tpmlteSoap
$
```

Figure 9-1 and IBM Tivoli Provisioning Manager supplied SOAP Scripts

This directory contains a number of sample SOAP scripts (discussed in 9.3.1, “Running the SOAP client” on page 230) together with the SOAP client. There are three variants of the SOAP client scripts that are platform dependent. Use the script that coincides with the platform you are running:

- `soapcli.cmd`

  Use in a Windows DOS shell. You can also use it in a cygwin shell, which is opened on a Windows and IBM Tivoli Provisioning Manager server.

- `soapcli.sh`

  Use on a UNIX or Linux and IBM Tivoli Provisioning Manager server.

- `soap_bash4win.sh`

  Use inside a bash shell running on a Windows and IBM Tivoli Provisioning Manager server. You must modify this script to point to a suitable Java Development Kit (JDK™) version1.4 or later.

9.2.1 Running a remote SOAP Client

To enable a remote server to communicate with the and IBM Tivoli Provisioning Manager, perform the following set up needs on the remote server:

1. Copy the appropriate SOAP client script and Java archive (JAR) files to the remote server:

   a. On the remote server, create a directory for the SOAP client.

   b. Copy the contents of the SOAP client directory from the and IBM Tivoli Provisioning Manager server to the SOAP client directory on the remote server:

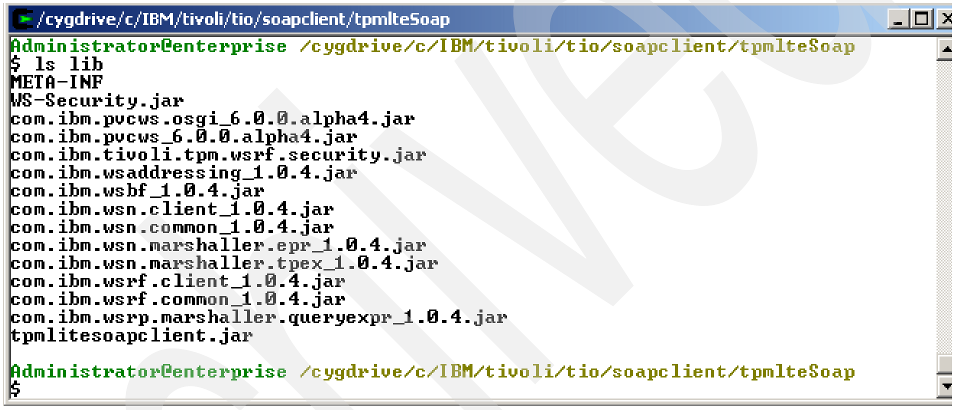
   c. On the remote server, change to the SOAP client directory.

   d. Open the appropriate SOAP script in a text editor.

   e. Set the value of TIO_LIB by changing the value of the TIO_HOME variable to the SOAP client directory that you created on the remote server:

      - On Windows: `set TIO_LIB=%TIO_HOME%\tpmlteSoap\lib`
      - On UNIX or Linux: `TIO_LIB=$TIO_HOME\tpmlteSoap\lib`

2. The JAR files that you need to copy are in the `lib` subdirectory of the SOAP client directory, as shown in Figure 9-2.



```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/soapclient/tpmlteSoap
$ ls lib
META-INF
WS-Security.jar
com.ibm.pvcws.osgi_6.0.0.alpha4.jar
com.ibm.pvcws_6.0.0.alpha4.jar
com.ibm.tivoli.tpm.wsrf.security.jar
com.ibm.wsaddressing_1.0.4.jar
com.ibm.wsbf_1.0.4.jar
com.ibm.wsn.client_1.0.4.jar
com.ibm.wsn.common_1.0.4.jar
com.ibm.wsn.marshaller.epr_1.0.4.jar
com.ibm.wsn.marshaller.tpex_1.0.4.jar
com.ibm.wsrf.client_1.0.4.jar
com.ibm.wsrf.common_1.0.4.jar
com.ibm.wsrp.marshaller.queryexpr_1.0.4.jar
tpmlitesoapclient.jar

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/soapclient/tpmlteSoap
$
```

Figure 9-2 and IBM Tivoli Provisioning Manager SOAP Client JAR files

3. Modify the **`soapcli`** script that you will use to point to the correct location of the JDK. You must install a Java Development Kit (JDK) that supports Java Technology version 1.4 or later on the remote server. Before you begin to configure the remote server, ensure that the JDK is installed and configured:

   a. Install a JDK if one is not currently installed. You can obtain a JDK from IBM, Sun Microsystems, and various open source groups.

      - The IBM developer kits are available from:

        `http://www-106.ibm.com/developerworks/java/jdk/`

      - The Sun Microsystems developer kits are available from:

        `http://java.sun.com`

Note: Do not include spaces in the JDK installation directory.

b. Ensure that the JAVA_HOME environment variable is set to the JDK installation directory.

c. If you are using a remote Windows server, add the \java\bin directory to the system path.

9.3 Examples of using SOAP

In this section, we show a number of examples using the various SOAP scripts and commands.

9.3.1 Running the SOAP client

The general syntax of making a SOAP call is:

```
soapcli.cmd username password wsdl_location operation parameters
```

Following is an explanation of the syntax:

- The username and password are for a valid user that is created in and IBM Tivoli Provisioning Manager.
- The wsdl location refers to the URL for the Web Services Definition Language (WSDL) file, and is of the form:

  ```
  http://hostname:port/ws/pid/wsdlservicename?wsdl
  ```

  The Web address includes the following parameters:

  – Host name

    The fully qualified domain name of the and IBM Tivoli Provisioning Manager server or local host, if the client is running locally on the and IBM Tivoli Provisioning Manager server.

  – Port

    The port number (the default is 8777).

  – wsdlservicename

    The name of the WSDL service.

  – Operation

    The WSDL operation that you want to run.

  – Parameters

    The parameters for the specified method or operation. You can obtain the values for most of the parameters from the Web interface. Move the mouse cursor over the appropriate device to obtain the device ID.

Table 9-1 shows the WSDL services that are supported in and IBM Tivoli Provisioning Manager. Type the URL shown into a browser on the and IBM Tivoli Provisioning Manager, to get a list of all operations and their definitions available.

Table 9-1 WSDL service name and URL

WSDL service name	WSDL service URL
TpmLiteSoapService	`http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl`
CredentialsManagerService	`http://localhost:8777/ws/pid/CredentialsManagerService?wsdl`
SPOfferingService	`http://localhost:8777/ws/pid/SPOfferingService?wsdl`
SPSubscriptionService	`http://localhost:8777/ws/pid/SPSubscriptionService?wsdl`
EffectiveModeService	`http://localhost:8777/ws/pid/EffectiveModeService?wsdl`
OperationsModeService	`http://localhost:8777/ws/pid/OperationsModeService?wsdl`
FaultManagementService	`http://localhost:8777/ws/pid/FaultManagementService?wsdl`
RecommendationsSerivice	`http://localhost:8777/ws/pid/RecommendationsService?wsdl`
ResourceInformationService	`http://localhost:8777/ws/pid/ResourceInformationService?wsdl`

As an example, on the and IBM Tivoli Provisioning Manager server, the following URL shows an XML file:

```
http://localhost:8777/ws/pid/OperationsModeService?wsdl
```

As shown in Example 9-1, the portion of this file from which the operations can be determined begins with - `<xsd:schema targetNamespace="http://service.Webservice.soap.tpm.tivoli.ibm.com">`

The operation keyword, which you use in the SOAP command, is shown in bold.

Example 9-1 Sample output from browsing to a WSDL URL

```
- <xsd:schema targetNamespace="http://service.webservice.soap.tpm.tivoli.ibm.com">
  <xsd:element name="setGlobalMode" type="xsd:string" />
  <xsd:element name="createWSResourceResponse" type="apache-soap:Element" />
+ <xsd:element name="getGlobalMode">
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="getEffectiveClusterMode" type="xsd:int" />
  <xsd:element name="setApplicationMaintenanceMode"
type="pns1:SetApplicationMaintenanceModeRequest" />
  <xsd:element name="getEffectiveClusterModeResponse" type="xsd:string" />
  <xsd:element name="getClusterModeResponse" type="xsd:string" />
+ <xsd:element name="setApplicationModeResponse">
```

```
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="getEffectiveApplicationModeResponse" type="xsd:string" />
+ <xsd:element name="setGlobalModeResponse">
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="createWSResource" type="apache-soap:Element" />
+ <xsd:element name="setMaintenanceModeResponse">
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="setMaintenanceMode" type="pns1:SetMaintenanceModeRequest" />
  <xsd:element name="setClusterMode" type="pns1:SetClusterModeRequest" />
+ <xsd:element name="setApplicationMaintenanceModeResponse">
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="setApplicationMode" type="pns1:SetApplicationModeRequest" />
  <xsd:element name="getApplicationModeResponse" type="xsd:string" />
  <xsd:element name="getClusterMode" type="xsd:int" />
+ <xsd:element name="setClusterModeResponse">
  <xsd:complexType />
  </xsd:element>
  <xsd:element name="getEffectiveApplicationMode" type="xsd:int" />
  <xsd:element name="getApplicationMode" type="xsd:int" />
  <xsd:element name="getGlobalModeResponse" type="xsd:string" />
  </xsd:schema>
```

Example 9-2 is an example of making a SOAP call to find all the deployment requests that were executed.

Example 9-2 Using the soapcli.cmd to retrieve all deployment requests

```
$ ./soapcli.cmd tioadmin object00
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl findAllDeploymentRequests
```

The output from the command in Example 9-2 is shown in Example 9-3.

Example 9-3 Output of the findAllDeplotmentRequests operation

```
Deployment Requests:
Total: 14
requestId=10085, workflowName=Cluster.RemoveServer, status=success, createdDate=Tue
Sep 19 22:04:54 BST 2006
requestId=10083, workflowName=Cluster.AddServer, status=success, createdDate=Tue Sep
19 22:01:51 BST 2006
```

```
requestId=10101, workflowName=Cluster.RemoveServer, status=success, createdDate=Tue
Sep 19 21:52:58 BST 2006
requestId=10100, workflowName=Cluster.AddServer, status=success, createdDate=Tue Sep
19 21:50:31 BST 2006
requestId=10080, workflowName=Execute_Local_Command, status=success, createdDate=Tue
Sep 19 21:30:10 BST 2006
requestId=10062, workflowName=loop, status=success, createdDate=Tue Sep 19 16:03:49
BST 2006
requestId=10061, workflowName=loop, status=success, createdDate=Tue Sep 19 16:03:20
BST 2006
requestId=10060,workflowName=Education_Installable_install,status=success,
createdDate=Tue Sep 19 15:27:46 BST 2006
requestId=10042, workflowName=Education_Installation_AddInstance, status=success,
createdDate=Tue Sep 19 13:06:45 BST 2006
requestId=10041,workflowName=Education_Installable_install,status=success,
createdDate=Tue Sep 19 13:06:14 BST 2006
requestId=10040, workflowName=Education_Installable_install, status=success,
createdDate=Tue Sep 19 12:57:04 BST 2006
requestId=10021, workflowName=Education_Installable_install, status=success,
createdDate=Tue Sep 19 08:07:06 BST 2006
requestId=10020, workflowName=Education_Instance_Start, status=success,
createdDate=Tue Sep 19 08:05:53 BST 2006
requestId=10000, workflowName=Cluster.AddServer, status=success, createdDate=Wed Aug
23 12:04:20 BST 2006
```

9.3.2 Running the SOAP scripts

The out of the box SOAP scripts are simply wrapper scripts for commonly used soap commands, as shown in Table 9-2.

Table 9-2 SOAP Wrapper Scripts

Script name and usage	Description
approve.cmd username password recommendation_id	Approves the specified deployment recommendation, while in semiautomatic mode.
deploy.cmd username password cluster_ID number_of_servers	Adds or removes the specified number of servers to or from a tier. The tier must be in manual mode. Use negative numbers for server removal.
failserver.cmd username password server_id	Sets a given server to the Failed state.

Script name and usage	Description
repairserver.cmd username password server_id	Repairs a given server.
manual.cmd username password	Changes the current global operating mode to manual. (Relevant to IBM Tivoli Intelligent Orchestrator.)
semi.cmd username password	Changes the current global operating mode to semiautomatic. (Relevant to IBM Tivoli Intelligent Orchestrator.)
auto.cmd username password	Changes the current global operating mode to automatic. (Relevant to IBM Tivoli Intelligent Orchestrator.)
mode.cmd username password	Returns the current global operating mode. (Relevant to IBM Tivoli Intelligent Orchestrator.)
cancelDeploymentRequest.cmd username password DeploymentRequestID	Cancels the specified deployment request.
createByXmlImport.cmd username password file_name	Imports a resource defined in XML format into the data center model.
findDeploymentStatus.cmd username password DeploymentRequestID	Checks the status of the workflow.
findErrorMessage.cmd username password DeploymentRequestID	Finds the error message associated with a workflow.
findLogDetails.cmd username password DeploymentRequestID	Finds log information associated with a workflow.
findParameterValue.cmd username password DeploymentRequestID parametername	Find the value of a parameter in a deployment request.
getLocalServerId.cmd username password	Returns the value of the local server ID.

9.3.3 Running logical operations using SOAP

A common use of the SOAP interface is to run logical operations as part of driving the automation from an external system, such as a help desk or event management system.

The *TpmLiteSoapService* provides an operation called *executeDeploymentRequest*, which you can use to run any logical device operation, as shown in Example 9-4.

Example 9-4 Using soapcli to run a logical device operation

```
$ ./soapcli.cmd <user id> <password>
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl executeDeploymentRequest
<Logical Device.Logical Operation> "<parameter1=value1> <parameter2=value2>"
```

The logical device operation parameter is of the form Logical Device.Logical Operation. The following list contains a few examples:

- Cluster.AddServer
- Cluster.RemoveServer
- BootServer.CaptureImage
- BootServer.InstallImage
- SoftwareModule.Install
- SoftwareInstallation.AddInstance
- SoftwareInstallation.Uninstall
- SoftwareInstance.RemoveInstance
- SoftwareInstance.Start
- SoftwareInstance.Stop

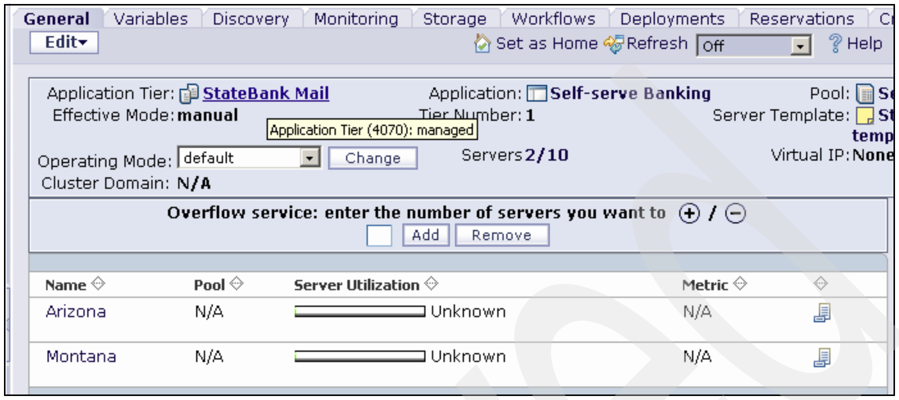
Example 9-10 on page 239, provides a complete list of logical device operations (LDOs).

Use the command in Example 9-5 to add a server to an application tier.

Example 9-5 Using soapcli to run the Cluster.AddServer Logical Device Operation

```
$ ./soapcli.cmd tioadmin object00
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl executeDeploymentRequest
Cluster.AddServer "ClusterID=4070"
```

The Cluster.AddServer LDO requires the ClusterID as an input parameter. You obtain this parameter by using the User Interface and moving the mouse over the Application Cluster hyperlink, as shown in Figure 9-3.



Application Tier: StateBank Mail
Effective Mode: manual
Application Tier (4070): managed
Operating Mode: default
Cluster Domain: N/A
Application: Self-serve Banking
Tier Number: 1
Servers 2/10
Pool: Se
Server Template: St temp
Virtual IP: None
Refresh: Off

Overflow service: enter the number of servers you want to ⊕ / ⊖

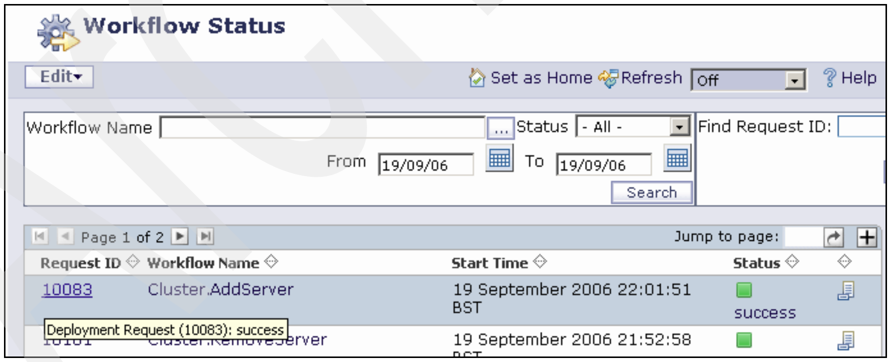
Name	Pool	Server Utilization	Metric	
Arizona	N/A	Unknown	N/A	
Montana	N/A	Unknown	N/A	

Figure 9-3   Obtaining the Application Cluster ID

The output of the SOAP command is the deployment request id, as shown in Example 9-6.

Example 9-6   The output of running the Cluster.AddServer executeDeploymentRequest

```
$ ./soapcli.cmd tioadmin object00
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl executeDeploymentRequest
Cluster.AddServer "ClusterID=4070"Result: 10083
```



Refresh: Off
Workflow Name
Status - All -
Find Request ID:
From 19/09/06 To 19/09/06

Request ID	Workflow Name	Start Time	Status	
10083	Cluster.AddServer	19 September 2006 22:01:51 BST	🟩 success	
10101	Cluster.RemoveServer	19 September 2006 21:52:58 BST	🟩	

Deployment Request (10083): success

Figure 9-4   Workflow Execution Status

You can view this request, together with the completion, in the User Interface “Workflow Status” panel, as shown in Figure 9-4. You can obtain the completion status using the SOAP command shown in Example 9-7 on page 237.

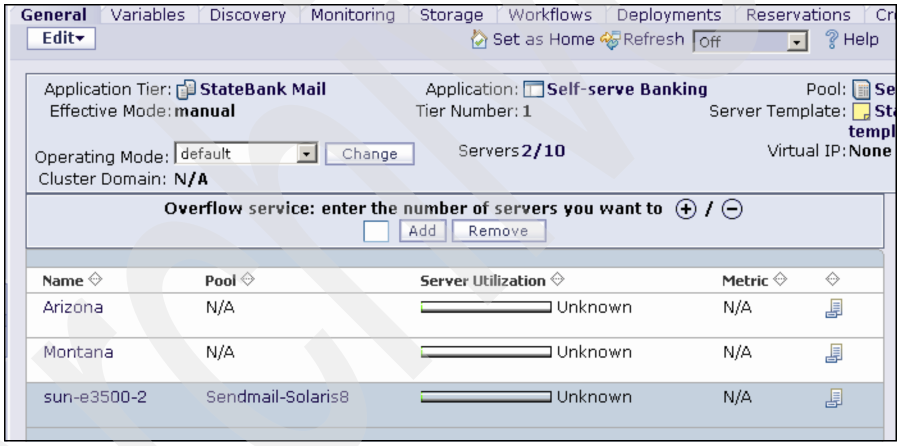
Example 9-7 Using soapcli to view the completion status of a deployment request

```
$ ./soapcli.cmd tioadmin object00
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl findDeploymentStatus 10083
Result: 3
```

The output of the *findDeploymentStatus* operation returns one of the following numerical values, which represents the status of the workflow.

0 unknown
1 in-progress
2 cancelled
3 success
4 failed
5 created

Figure 9-5 shows that the Cluster.AddServer LDO succeeded, and provisioned on a server into the StateBank Mail Application Tier.



General | Variables | Discovery | Monitoring | Storage | Workflows | Deployments | Reservations | Cr

Edit▾ Set as Home Refresh Off Help

Application Tier: StateBank Mail
Effective Mode: manual
Application: Self-serve Banking
Tier Number: 1
Pool: Se
Server Template: St
templ
Operating Mode: default Change
Servers 2/10
Virtual IP: None
Cluster Domain: N/A

Overflow service: enter the number of servers you want to ⊕ / ⊖
Add Remove

Name	Pool	Server Utilization	Metric	
Arizona	N/A	Unknown	N/A	
Montana	N/A	Unknown	N/A	
sun-e3500-2	Sendmail-Solaris8	Unknown	N/A	

Figure 9-5 Result of Cluster.AddServer

Some deployment requests may take longer to execute; therefore, you will see a result code of 1, which indicates that the workflow is in progress, as shown in Example 9-8.

Example 9-8 Using soapcli to show the deployments in progress

```
$ ./soapcli.cmd tioadmin object00
http://localhost:8777/ws/pid/TpmLiteSoapService?wsdl findDeploymentStatus 10085
Result: 1
```

To obtain further execution details of a workflow, you can use the *findLogDetails* operation, as shown in Example 9-9.

Example 9-9 Using soapcli to show execution log details of a deployment request

```
$ ./soapcli.cmd tioadmin object00 http://localhost:8777/ws/pid/TpmLiteSoapServi
ce?wsdl findLogDetails 10085
Log Details:
Total: 37
Start logical operation:  'Cluster.RemoveServer'
Start workflow:  'Simulator_RemoveServer'
Start workflow:  'Get_Current_Deployment_Request_Id'
Start Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.GetCurrentDeploymentRequestId'
End Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.GetCurrentDeploymentRequestId'
End workflow:  'Get_Current_Deployment_Request_Id'
Start workflow:  'Get_Cluster_Attributes'
End workflow:  'Get_Cluster_Attributes'
Start Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMChooseServerForRemoval'
End Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMChooseServerForRemoval'
Processing server template StateBank Mail template 602 for application tier StateBank
Mail.
Cannot find matching NIC for NIC template 41.
Start workflow:  'RM_Remove_Server'
Start Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMRemoveServerFromCluster'
End Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMRemoveServerFromCluster'
End workflow:  'RM_Remove_Server'
Processing server template Sendmail-Solaris8 template 200 for resource pool
Sendmail-Solaris8.
Processing NIC 3858 with NIC template 26
Processing network interface template 26
Start Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMAllocateIPAddress'
End Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.resourcemanager.RMAllocateIPAddress'
Start workflow:  'Create_Network_Interface'
End workflow:  'Create_Network_Interface'
Start workflow:  'Simulator_Move_Net_Interfaces'
Start Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.simulator.SimulatorMoveNetworkInterfaces'
```

```
End Java plugin
'com.thinkdynamics.kanaha.de.javaplugin.simulator.SimulatorMoveNetworkInterfaces'
End workflow:  'Simulator_Move_Net_Interfaces'
Start workflow:  'Set_NIC_s_VLAN'
Start Java plugin 'com.thinkdynamics.kanaha.de.javaplugin.datacentermodel.SetNicVlan'
End Java plugin 'com.thinkdynamics.kanaha.de.javaplugin.datacentermodel.SetNicVlan'
End workflow:  'Set_NIC_s_VLAN'
End workflow:  'Simulator_RemoveServer'
Start workflow:  'PostClusterRemoveServer'
Start Java
plugin'com.thinkdynamics.kanaha.de.javaplugin.GetCurrentDeploymentRequestId'
End Java plugin'com.thinkdynamics.kanaha.de.javaplugin.GetCurrentDeploymentRequestId'
End workflow:  'PostClusterRemoveServer'
End logical operation:  'Cluster.RemoveServer'
```

9.4 Tivoli Provisioning Manager for Software logical device operations

Example 9-10 shows a list of all logical device operations available in and IBM Tivoli Provisioning Manager through the user interface. The operations are in the form of Logical Device.Logical Operation.

Example 9-10   Logical device operations

```
AdminDomain.AddNode

Application.Deploy
Application.DisruptiveUpgrade
Application.NonDisruptivePatch
Application.Undeploy

ApplicationDeployment.Undeploy

ApplicationModule.Deploy

BackupResource.Backup
BackupResource.Restore

BackupSystem.Backup
BackupSystem.Restore

BootServer.CaptureBackupImage
BootServer.CaptureImage
BootServer.DeleteImage
BootServer.InstallGoldenMasterImage
BootServer.InstallImage
BootServer.InstallScriptedImage
BootServer.ReplicateImage
BootServer.RestoreBackupImage

Cluster.AddRepairedServer

Cluster.AddServer

ClusterDomain.AddNode
ClusterDomain.Config
ClusterDomain.Create
ClusterDomain.CreateNode
ClusterDomain.CreateResource
ClusterDomain.CreateResourceGroup
ClusterDomain.Remove
ClusterDomain.RemoveNode
ClusterDomain.Start
ClusterDomain.StartNode
ClusterDomain.Stop
ClusterDomain.StopNode
ClusterDomain.UpdateDomainStatus
ClusterDomain.UpdateNodeStatus

ClusterResource.AddGroupMember
ClusterResource.CreateDependency
ClusterResource.RemoveDependency
ClusterResource.RemoveResource
ClusterResource.RemoveResourceGroup
ClusterResource.Start
ClusterResource.StartResourceGroup
ClusterResource.Stop
ClusterResource.StopResourceGroup
ClusterResource.Update
ClusterResource.UpdateResourceGroup

Compliance.ValidateServer

ComplianceRecommendation.Remediate

ComplianceRecommendationGroup.Remediate

ConfigurationCompliance.OnNonComplianceEvent
```

```
Device.AddMPLSPathToMPLSTunnel
Device.ClearAllTunnels
Device.CopyFile
Device.CreateMPLSPath
Device.CreateMPLSTunnel
Device.CreateServiceAccessPoint
Device.DeleteMPLSPath
Device.DeleteMPLSTunnel
Device.Discover
Device.DiscoverDrift
Device.ExecuteCommand
Device.GetAttribute
Device.HardwareReboot
Device.Initialize
Device.Ping
Device.PowerOff
Device.PowerOn
Device.Reboot
Device.Rebuild
Device.RemoveMPLSPathFromMPLSTunnel
Device.RemoveServiceAccessPoint
Device.SetAttribute
Device.SoftwareRebootAsync
Device.SoftwareRebootSync
Device.UpdateMPLSTunnelActivePath

DeviceDriver.Associate
DeviceDriver.Dissociate

Discovery.CheckStatus
Discovery.Discover
Discovery.Drift
Discovery.OnDevice
Discovery.OnDeviceChangedEvent
Discovery.OnDeviceRemovedEvent
Discovery.OnDevices
Discovery.OnDeviceUpdatedEvent
Discovery.OnNewDevicesUpdateDcm
Discovery.Start
Discovery.Stop
Discovery.Update
Discovery.UpdateDcm

EventConsumer.OnEvent
```

```
FileRepository
FileRepository.GetFile
FileRepository.MountShare
FileRepository.Publish
FileRepository.PutFile
FileRepository.RemoveFile
FileRepository.UnmountShare
FileRepository.UnPublish

Firewall.AddACL
Firewall.DisableACL
Firewall.EnableACL
Firewall.RemoveACL

Group.Refresh

HostingEnvironment.Host
HostingEnvironment.Undeploy

HostPlatform.AllocateVirtualResource
HostPlatform.CreateVirtualServer
HostPlatform.DeallocateVirtualResource
HostPlatform.DestroyVirtualServer
HostPlatform.ExpandResourceAllocation
HostPlatform.ReduceResourceAllocation

IPSystem.AddNetworkInterface
IPSystem.ApplyRoutingTable
IPSystem.RemoveNetworkInterface

LoadBalancer.AddRealIPToVirtualIP
LoadBalancer.CreateVirtualIP
LoadBalancer.RemoveRealIPFromVirtualIP
LoadBalancer.RemoveVirtualIP

MonitoringApplication.ConfigureMonitoring
MonitoringApplication.RemoveMonitoring
MonitoringApplication.StartMonitoring
MonitoringApplication.StopMonitoring

OperatingSystem.AddGroup
OperatingSystem.AddNetworkInterface
OperatingSystem.AddUser
OperatingSystem.AddUserToGroup
OperatingSystem.ApplyRoutingTable
OperatingSystem.CaptureUserProfile
OperatingSystem.CreateDASDPhysicalVolume
OperatingSystem.CreateSANPhysicalVolume
OperatingSystem.RemoveGroup
OperatingSystem.RemoveNetworkInterface
OperatingSystem.RemovePhysicalVolume
OperatingSystem.RemoveUser
OperatingSystem.RemoveUserFromGroup
OperatingSystem.RemoveUserProfile
OperatingSystem.RestoreUserProfile
OperatingSystem.SetUserPassword
OperatingSystem.SoftwareRebootAsync
OperatingSystem.SoftwareRebootSync

PowerUnit.CycleOutlet
PowerUnit.TurnOutletOFF
PowerUnit.TurnOutletON

Repository.ImportPatches
Repository.UpdatePatches

Router.CreateRoute
Router.RemoveRoute

SANFabric.AddZoneMembers
SANFabric.CreateZone
SANFabric.RemoveZone
SANFabric.RemoveZoneMembers

ServerTemplate.ConfigureServerByTemplate

ServiceAccessPoint
ServiceAccessPoint.CopyFile
ServiceAccessPoint.CreatePwdCredential
ServiceAccessPoint.CreateRSACredential
ServiceAccessPoint.CreateSNMPCredential
ServiceAccessPoint.ExecuteCommand
ServiceAccessPoint.GetAttribute
ServiceAccessPoint.PingIP
ServiceAccessPoint.RemovePwdCredential
ServiceAccessPoint.RemoveRSACredential
ServiceAccessPoint.RemoveSNMPCredential
ServiceAccessPoint.ReplacePwdCredential
ServiceAccessPoint.ReplaceRSACredential
ServiceAccessPoint.ReplaceSNMPCredential
ServiceAccessPoint.SetAttribute

Software.CheckStatus
Software.Install
Software.Start
Software.Stop
Software.Uninstall

SoftwareApplicationDeployment
SoftwareApplicationDeployment.Deploy

SoftwareDistributionApplication.RegisterSoftwarePackage
SoftwareDistributionApplication.UnregisterSoftwarePackage

SoftwareInstallable.Distribute
SoftwareInstallable.Install
SoftwareInstallable.MultipleInstall
SoftwareInstallable.Publish
SoftwareInstallable.UnPublish

SoftwareInstallation.AddInstance
SoftwareInstallation.Uninstall

SoftwareInstance.RemoveInstance
SoftwareInstance.Start
SoftwareInstance.Stop

SoftwareModule.DistributeInstallable
SoftwareModule.Install
SoftwareModule.MultipleInstall

SoftwareResource.Configure
SoftwareResource.UpdateDCM
SoftwareResource.Upgrade

SoftwareStack.Install
SoftwareStack.Uninstall

SparePool.CleanupServer
SparePool.InitializeServer

SPService.Deprovision
SPService.Modify
SPService.ProcessOrder
SPService.Provision
SPService.Reserve

StorageManager.AddPhysicalVolumeToLogicalVolume
StorageManager.AddPhysicalVolumeToVolumeContainer
StorageManager.AddServerToVolumeContainer
StorageManager.AddStorageToHost
StorageManager.AddStorageVolumeArrayToHost
StorageManager.AddStorageVolumeToHost
StorageManager.Clone
StorageManager.CreateFileSystem
StorageManager.CreateLogicalVolume
StorageManager.CreateMirrorVolume
StorageManager.CreateStripeVolume
StorageManager.CreateVolumeContainer
StorageManager.ExtendFileSystem
StorageManager.ExtendLogicalVolume
StorageManager.RemoveFileSystem
StorageManager.RemoveLogicalVolume
StorageManager.RemovePhysicalVolumeFromLogicalVolume
StorageManager.RemovePhysicalVolumeFromVolumeContainer
StorageManager.RemoveServerFromVolumeContainer
StorageManager.RemoveStorageFromHost
StorageManager.RemoveStorageVolumeArrayFromHost
StorageManager.RemoveVolumeContainer
StorageManager.RmStorageVolumeFromHost

StoragePool.CreateStorageVolume
StoragePool.CreateStorageVolumeWithName
StoragePool.GetStorageVolumes
StoragePool.RemoveStorageVolume

StorageSubsystem.CreateStorageVolume
StorageSubsystem.CreateStorageVolumeWithName
StorageSubsystem.GetStorageVolumes
StorageSubsystem.LunMapping
StorageSubsystem.LunMasking
StorageSubsystem.LunUnmapping
StorageSubsystem.LunUnmasking
StorageSubsystem.MapVolumeArrayToFA
StorageSubsystem.MapVolumeToFA
StorageSubsystem.RemoveStorageVolume
StorageSubsystem.UnmapVolumeArrayFromFA
StorageSubsystem.UnmapVolumeFromFA
```

```
Switch.CarryVLANThroughSwitchEndpoint
Switch.ChangeSwitchEndpointAttributes
Switch.ClearVLANFromSwitchEndpoint
Switch.CreateVLAN
Switch.MovePort
Switch.MovePortToVLAN
Switch.RemoveVLAN
Switch.TurnPortOFF
Switch.TurnPortON

Task.OnTaskFailEvent
Task.OnTaskStartEvent
Task.OnTaskSuccessEvent

TpmTask.CancelInProgress
TpmTask.Execute

TpmTaskDefinition.Execute
```

9.5 Tivoli Provisioning Manager for Software scripts

A number of scripts are provided in the and IBM Tivoli Provisioning Manager installation subdirectory, called **`tools`**, as shown in Figure 9-6 on page 247. We discuss some of these scripts in this section.

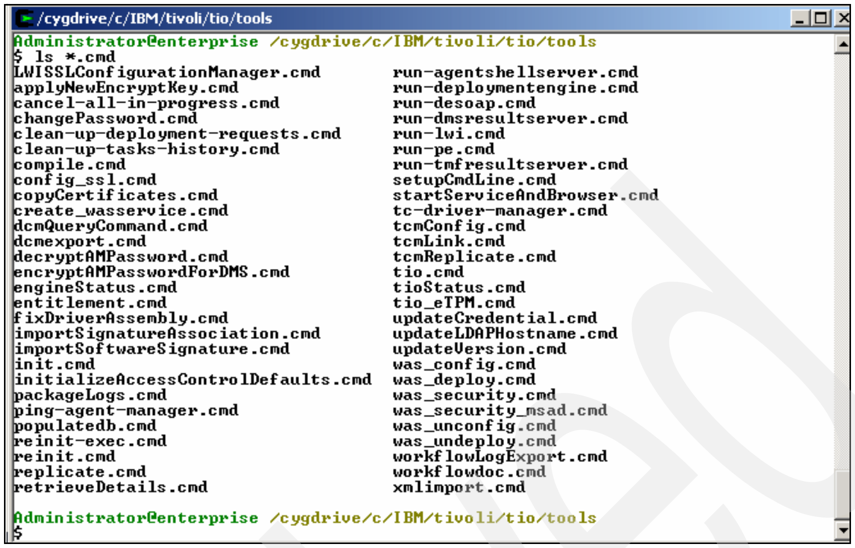
Each script is supplied in two variants:

- **`.cmd`** extension

  Use the **`.cmd`** variant on a Windows and IBM Tivoli Provisioning Manager server.

- **`.sh`** extension

  You can only run the **`.sh`** variant on a Linux or Unix and IBM Tivoli Provisioning Manager server.



```
/cygdrive/c/IBM/tivoli/tio/tools
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ls *.cmd
LWISSLConfigurationManager.cmd        run-agentshellserver.cmd
applyNewEncryptKey.cmd                run-deploymentengine.cmd
cancel-all-in-progress.cmd            run-desoap.cmd
changePassword.cmd                    run-dmsresultserver.cmd
clean-up-deployment-requests.cmd      run-lwi.cmd
clean-up-tasks-history.cmd            run-pe.cmd
compile.cmd                           run-tmfresultserver.cmd
config_ssl.cmd                        setupCmdLine.cmd
copyCertificates.cmd                  startServiceAndBrowser.cmd
create_wasservice.cmd                 tc-driver-manager.cmd
dcmQueryCommand.cmd                   tcmConfig.cmd
dcmexport.cmd                         tcmLink.cmd
decryptAMPassword.cmd                 tcmReplicate.cmd
encryptAMPasswordForDMS.cmd           tio.cmd
engineStatus.cmd                      tioStatus.cmd
entitlement.cmd                       tio_eTPM.cmd
fixDriverAssembly.cmd                 updateCredential.cmd
importSignatureAssociation.cmd        updateLDAPHostname.cmd
importSoftwareSignature.cmd           updateVersion.cmd
init.cmd                              was_config.cmd
initializeAccessControlDefaults.cmd   was_deploy.cmd
packageLogs.cmd                       was_security.cmd
ping-agent-manager.cmd                was_security_msad.cmd
populatedb.cmd                        was_unconfig.cmd
reinit-exec.cmd                       was_undeploy.cmd
reinit.cmd                            workflowLogExport.cmd
replicate.cmd                         workflowdoc.cmd
retrieveDetails.cmd                   xmlimport.cmd

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$
```

Figure 9-6 Command scripts located in $TIO_HOME/tools

9.5.1 Starting Tivoli Provisioning Manager on the command line

You can use the script **`startServiceAndBrowser.cmd`** to start the and IBM Tivoli Provisioning Manager service, and then launch the Internet Browser. If and IBM Tivoli Provisioning Manager is already running, then a message is displayed, and only the Internet browser is launched, as shown in Example 9-11.

Example 9-11 Starting and IBM Tivoli Provisioning Manager on the command line

```
$ ./startServiceAndBrowser.cmd
TPM is already running
```

9.5.2 Working with automation packages

An automation package is an installation unit that consists of the scripts, workflows, documentation, and Java files for a particular device or software package. Automation packages have a .tcdriver extension and are centrally stored in the <installation location>\drivers directory of the and IBM Tivoli Provisioning Manager server.

You can install automation packages using the **`tc-driver-manager.cmd`** command. If this script is run with no or incorrect arguments, then the output shown in Example 9-12 on page 248 is displayed.

You can also use the **tc-driver-manager.cmd** command to list the installation status of each automation package, which exists in the <installation location>\drivers directory, as shown in Example 9-12.

Example 9-12 Listing status of automation packages

```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ./tc-driver-manager.cmd listAllStr
2006-09-20 16:41:51,234 INFO  log4j configureAndWatch is started with configurat
ion file: C:\IBM\tivoli\tio/config/log4j-util.prop
2006-09-20 16:41:51,562 INFO  COPTDM001I TCDrivermanager was started.
2006-09-20 16:41:52,562 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:41:52,656 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:41:52,703 INFO  COPTDM002I Driver directory: "Driver directory: "C
:\IBM\tivoli\tio/drivers/".".
2006-09-20 16:42:04,859 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:42:08,547 INFO  COPTDM005I TCDrivermanager was stopped.
2006-09-20 16:42:08,562 INFO

TC Driver Name                          Version           Status
==============                          ================= ================
AIX-LVM                                                   not-installed
AIX-Operating-System                                      not-installed
AIX_FIX                                                   not-installed
Apache-Web-Server-Windows                                 not-installed
ApacheLinux                             1.0               installed
ApacheWebServerWindows                  5.1.0.0.1767.38   installed
Blade-Center-4p-Gb-Eth                                    not-installed
CDS                                     5.1.0.0.1767.38   installed
CHAMPS                                                    not-installed
CIMClient                                                 not-installed
CSM-Linux-Install                                         not-installed
CiscoSwitchDiscovery                                      not-installed
CitScannerAgent                         5.1.0.0.1767.38   installed
Cygwin                                                    not-installed
DB2                                     5.1.0.0.1767.38   installed
DSMUtils                                5.1.0.0.1767.38   installed
F5-BIG-IP.3.3                                             not-installed
F5-BIG-IP.4.1                                             not-installed
F5-BIG-IP.4.5                                             not-installed
FileRepository                                            not-installed
Globalization_Helper                                      not-installed
```

```
HP-UX                                                        not-installed
HPUX_VPAR_CPU                                                not-installed
IBM-BladeCenter-Management                                   not-installed
IBM-ESS                                                      not-installed
IBM-Http-Server                                              not-installed
IBM-RDM                                                      not-installed
IBM-WebSphere-App-Server                                     not-installed
IBM-WebSphere-Edge                                           not-installed
IBMHttp                                   5.1.0.0.1767.38   installed
IBM_Director_Discovery                                       not-installed
ITM                                                          not-installed
ITM6.1                                                       not-installed
ITM61                                     5.1.0.0.1767.38   installed
LPAR-cpu                                                     not-installed
LocalFileRepository                       5.1.0.0.1767.38   installed
MS-Exchange-Server                                           not-installed
MS-SQL2K                                                     not-installed
MSADS                                                        not-installed
MSAD_Discovery                            5.1.0.0.1767.38   installed
MSClusterServices                                            not-installed
MS_VirtualServer                                             not-installed
MS_WSUS                                   5.1.0.0.1767.38   installed
NIM                                                          not-installed
Nortel_Alteon                                                not-installed
PeopleSoftTools                                              not-installed
Provider1                                                    not-installed
RXA                                       5.1.0.0.1767.38   installed
RedHat_EL_Update                                             not-installed
SCMCollectorAgent                         5.1.0.0.1767.38   installed
SPBHandler                                5.1.0.0.1767.38   installed
Sun_Update_Connection                                        not-installed
SymantecAV                                                   not-installed
TECDriver                                                    not-installed
TMAInstall                                                   not-installed
TPMNetworkDiscovery                       5.1.0.0.1767.38   installed
TivoliCommonAgent                         5.1.0.0.1767.38   installed
TivoliCommonAgent-core                    5.1.0.0.1767.38   installed
TpmTaskInf                                                   not-installed
Tuxedo                                                       not-installed
VxVM-VolumeManager                                           not-installed
apache                                                       not-installed
apc-snmp                                                     not-installed
application-service                                          not-installed
application-topologies                                       not-installed
backup                                                       not-installed
```

```
brocade                                                    not-installed
cisco-csm-6500                         5.1.0.0.1767.38   installed
cisco-css-11000                        5.1.0.0.1767.38   installed
cisco-discovery                                            not-installed
cisco-mds                                                  not-installed
cisco-pix                              5.1.0.0.1767.38   installed
cisco-switches                         5.1.0.0.1767.38   installed
citrix                                                     not-installed
cluster-domain-simulator               5.1.0.0.1767.38   installed
cluster-ldo-templates                  5.1.0.0.1767.38   installed
core                                   5.1.0.0.1767.38   installed
cvsserver                                                  not-installed
db2                                    5.1.0.0.1767.38   installed
default-device-model                   5.1.0.0.1767.38   installed
default_automation_package             5.1.0.0.1767.38   installed
discovery-library                      5.1.0.0.1767.38   installed
discovery-library-adaptor              5.1.0.0.1767.38   installed
dms-client-subagent                    5.1.0.0.1767.38   installed
domino                                                     not-installed
dummy-load-balancer                                        not-installed
dummy-switch                                               not-installed
educationDriver                        1.0               installed
event-subagent                         5.1.0.0.1767.38   installed
extreme-48i                                                not-installed
foundry                                                    not-installed
hacmp                                                      not-installed
image                                  5.1.0.0.1767.38   installed
itds60                                                     not-installed
jes-subagent                           5.1.0.0.1767.38   installed
jumpstart                                                  not-installed
mailer                                                     not-installed
mcdata                                                     not-installed
microsoft-patch                                            not-installed
netview                                                    not-installed
networking-advanced                    5.1.0.0.1767.38   installed
oa-tio-capacity-on-demand                                  not-installed
oracle10g                                                  not-installed
oracle9i                                                   not-installed
pSeries-Server                                             not-installed
proliant-bl-server                                         not-installed
rational-testmgr                                           not-installed
rdp-altiris                                                not-installed
redhat-linux-operating-system          5.1.0.0.1767.38   installed
redline                                                    not-installed
rembo                                  5.1.0.0.1767.38   installed
```

```
sample-compliance-validation                      not-installed
siebel                                            not-installed
simulator                     5.1.0.0.1767.38    installed
sles-operating-system         5.1.0.0.1767.38    installed
software_simulator                                not-installed
solaris-operating-system                          not-installed
storage-simulation            5.1.0.0.1767.38    installed
test                          1.0                installed
tsa-cluster-domain                                not-installed
vmware-4                                          not-installed
weblogic                                          not-installed
Windows-operating-system      5.1.0.0.1767.38    installed
zvm                                               not-installed
```

Example 9-13 shows an example of installing an automation package called *software_simulator.tcdriver*.

Example 9-13 Installing an automation package using tc-driver-manager.cmd

```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ./tc-driver-manager.cmd installDriver software_simulator
2006-09-20 16:51:40,484 INFO  log4j configureAndWatch is started with configurat
ion file: C:\IBM\tivoli\tio/config/log4j-util.prop
2006-09-20 16:51:41,234 INFO  COPTDM001I TCDrivermanager was started.
2006-09-20 16:51:43,734 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:51:43,875 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:51:43,953 INFO  COPTDM002I Driver directory: "Driver directory: "C
:\IBM\tivoli\tio/drivers/".".
2006-09-20 16:51:52,781 INFO  COPTDM003I Installing driver "software_simulator",
 force: "false", installItems: "true".
2006-09-20 16:52:17,000 INFO  COPTDM004I Config directory: "file:C:\IBM\tivoli\t
io/config/".
2006-09-20 16:52:22,156 INFO  COPTDM006I Creating DCM_OBJECT entry for TCDriver
"software_simulator".
2006-09-20 16:52:24,656 INFO  COPTDM011I Installing driver item "xml/dmcategorie
s.xml".
2006-09-20 16:52:26,094 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_CaptureBackupImage.wkf".
2006-09-20 16:52:29,938 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_CaptureImage.wkf".
2006-09-20 16:52:30,375 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_DeleteImage.wkf".
2006-09-20 16:52:30,656 INFO  COPTDM011I Installing driver item "workflow/Softwa
```

```
reSimulator_BootServer/software_simulator_BootServer_InstallGoldenMasterImage.wk
f".
2006-09-20 16:52:30,953 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_InstallImage.wkf".
2006-09-20 16:52:31,484 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_InstallScriptedImage.wkf".
2006-09-20 16:52:31,703 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_ReplicateImage.wkf".
2006-09-20 16:52:32,141 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_BootServer/software_simulator_BootServer_RestoreBackupImage.wkf".
2006-09-20 16:52:32,344 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_HostingSoftwareResource/software_simulator_HostingEnvironment_Host.w
kf".
2006-09-20 16:52:32,578 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_HostingSoftwareResource/software_simulator_HostingEnvironment_Host.w
kf".
2006-09-20 16:52:39,344 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_HostingSoftwareResource/software_simulator_HostingEnvironment_Undepl
oy.wkf".
2006-09-20 16:52:39,625 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_HostingSoftwareResource/software_simulator_HostingEnvironment_Undepl
oy.wkf".
2006-09-20 16:52:39,859 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstallable/software_simulator_SoftwareInstallable_Install.w
kf".
2006-09-20 16:52:40,109 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstallation/software_simulator_SoftwareInstallation_AddInst
ance.wkf".
2006-09-20 16:52:40,297 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstallation/software_simulator_SoftwareInstallation_Uninsta
ll.wkf".
2006-09-20 16:52:40,547 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstance/software_simulator_SoftwareInstance_RemoveInstance.
wkf".
2006-09-20 16:52:40,750 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstance/software_simulator_SoftwareInstance_Start.wkf".
2006-09-20 16:52:40,969 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareInstance/software_simulator_SoftwareInstance_Stop.wkf".
2006-09-20 16:52:42,203 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareModule/software_simulator_SoftwareModule_Install.wkf".
2006-09-20 16:52:42,781 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareResource/software_simulator_SoftwareResource_UpdateDCM.wkf".
2006-09-20 16:52:44,625 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_SoftwareResource/software_simulator_SoftwareResource_Upgrade.wkf".
2006-09-20 16:52:44,812 INFO  COPTDM011I Installing driver item "workflow/softwa
```

```
re_simulator_postinstall.wkf".
2006-09-20 16:52:48,266 INFO  COPTDM011I Installing driver item "workflow/test_S
oftware_LDOs.wkf".
2006-09-20 16:52:48,484 INFO  COPTDM011I Installing driver item "workflow/test_u
secase_one.wkf".
2006-09-20 16:52:50,562 INFO  COPTDM011I Installing driver item "workflow/test_u
secase_three.wkf".
2006-09-20 16:52:52,391 INFO  COPTDM011I Installing driver item "workflow/test_u
secase_two.wkf".
2006-09-20 16:52:53,297 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_ClusterDomain/Software_Simulator_AdminDomain_AddNode.wkf".
2006-09-20 16:52:53,453 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_ClusterDomain/SoftwareSimulator_ClusterDomain_CreateNode.wkf".
2006-09-20 16:52:55,359 INFO  COPTDM011I Installing driver item "workflow/Softwa
reSimulator_ClusterDomain/Software_Simulator_ClusterDomain_Remove.wkf".
2006-09-20 16:52:55,703 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_SoftwareInstallable".
2006-09-20 16:52:58,078 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_SoftwareInstallable".
2006-09-20 16:52:58,125 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstallable_Install" with Device Model "SoftwareSimulator_SoftwareInst
allable".
2006-09-20 16:52:59,109 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_SoftwareInstallation".
2006-09-20 16:52:59,516 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_SoftwareInstallation".
2006-09-20 16:52:59,547 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstallation_AddInstance" with Device Model "SoftwareSimulator_Softwar
eInstallation".
2006-09-20 16:52:59,594 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstallation_Uninstall" with Device Model "SoftwareSimulator_SoftwareI
nstallation".
2006-09-20 16:52:59,703 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_SoftwareInstance".
2006-09-20 16:52:59,891 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_SoftwareInstance".
2006-09-20 16:52:59,953 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstance_RemoveInstance" with Device Model "SoftwareSimulator_Software
Instance".
2006-09-20 16:53:00,016 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstance_Start" with Device Model "SoftwareSimulator_SoftwareInstance"
.
2006-09-20 16:53:00,062 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareInstance_Stop" with Device Model "SoftwareSimulator_SoftwareInstance".
```

```
2006-09-20 16:53:00,125 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_SoftwareModule".
2006-09-20 16:53:00,875 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_SoftwareModule".
2006-09-20 16:53:00,891 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareModule_Install" with Device Model "SoftwareSimulator_SoftwareModule".
2006-09-20 16:53:00,984 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_SoftwareResource".
2006-09-20 16:53:01,391 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_SoftwareResource".
2006-09-20 16:53:01,406 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareResource_UpdateDCM" with Device Model "SoftwareSimulator_SoftwareResou
rce".
2006-09-20 16:53:01,453 INFO  COPTDM017I Associating workflow "software_simulato
r_SoftwareResource_Upgrade" with Device Model "SoftwareSimulator_SoftwareResourc
e".
2006-09-20 16:53:01,500 INFO  COPTDM017I Associating workflow "software_simulato
r_HostingEnvironment_Host" with Device Model "SoftwareSimulator_SoftwareResource
".
2006-09-20 16:53:01,531 INFO  COPTDM017I Associating workflow "software_simulato
r_HostingEnvironment_Undeploy" with Device Model "SoftwareSimulator_SoftwareReso
urce".
2006-09-20 16:53:01,594 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_BootServer".
2006-09-20 16:53:01,734 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_BootServer".
2006-09-20 16:53:01,750 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_CaptureBackupImage" with Device Model "SoftwareSimulator_BootServer
".
2006-09-20 16:53:01,797 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_CaptureImage" with Device Model "SoftwareSimulator_BootServer".
2006-09-20 16:53:01,844 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_DeleteImage" with Device Model "SoftwareSimulator_BootServer".
2006-09-20 16:53:01,875 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_InstallGoldenMasterImage" with Device Model "SoftwareSimulator_Boot
Server".
2006-09-20 16:53:01,922 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_InstallImage" with Device Model "SoftwareSimulator_BootServer".
2006-09-20 16:53:01,953 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_InstallScriptedImage" with Device Model "SoftwareSimulator_BootServ
er".
2006-09-20 16:53:02,000 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_ReplicateImage" with Device Model "SoftwareSimulator_BootServer".
2006-09-20 16:53:02,031 INFO  COPTDM017I Associating workflow "software_simulato
r_BootServer_RestoreBackupImage" with Device Model "SoftwareSimulator_BootServer
```

```
".
2006-09-20 16:53:02,094 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_HostingSoftwareResource".
2006-09-20 16:53:02,234 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_HostingSoftwareResource".

2006-09-20 16:53:02,250 INFO  COPTDM017I Associating workflow "software_simulato
r_HostingEnvironment_Host" with Device Model "SoftwareSimulator_HostingSoftwareR
esource".
2006-09-20 16:53:02,297 INFO  COPTDM017I Associating workflow "software_simulato
r_HostingEnvironment_Undeploy" with Device Model "SoftwareSimulator_HostingSoftw
areResource".
2006-09-20 16:53:02,422 INFO  COPTDM012I Creating Device Model "SoftwareSimulato
r_ClusterDomain".
2006-09-20 16:53:02,562 INFO  COPTDM016I Creating the data center model object p
roperties template for Device Model "SoftwareSimulator_ClusterDomain".
2006-09-20 16:53:02,594 INFO  COPTDM017I Associating workflow "Software_Simulato
r_AdminDomain_AddNode" with Device Model "SoftwareSimulator_ClusterDomain".
2006-09-20 16:53:02,625 INFO  COPTDM017I Associating workflow "SoftwareSimulator
_ClusterDomain_CreateNode" with Device Model "SoftwareSimulator_ClusterDomain".
2006-09-20 16:53:02,656 INFO  COPTDM017I Associating workflow "Software_Simulato
r_ClusterDomain_Remove" with Device Model "SoftwareSimulator_ClusterDomain".
2006-09-20 16:53:03,500 INFO  COPTDM011I Installing driver item "xml/software_si
mulator.xml".
2006-09-20 16:53:29,703 INFO  installing driver plugin to C:\IBM\tivoli\tio/ecli
pse/plugins/software_simulator
2006-09-20 16:53:32,031 INFO  COPTDM005I TCDrivermanager was stopped.
2006-09-20 16:53:32,047 INFO
Successfully installed:
software_simulator
```

Use the following concocted command to confirm the installation:

```
$ ./tc-driver-manager.cmd listAllstr | grep software_simulator
software_simulator                    5.1.0.0.1767.38   installed
$
```

9.5.3 Retrieving information about DCM object definitions

Use the **retreiveDetails.cmd** script to get information about data center objects, by supplying a data center object id, as shown in Example 9-14.

Example 9-14 Retrieving information about data center objects

```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ./retrieveDetails.cmd
2006-09-22 09:14:50,938 INFO  log4j configureAndWatch is started with configuration
file: C:\IBM\tivoli\tio/config/log4j-util.prop
2006-09-22 09:15:05,297 ERROR COPCOM425E Usage: retrieveDetails -i <deviceId> [-
i <deviceId>]*.

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ./retrieveDetails.cmd -i 2222
2006-09-22 09:15:50,703 INFO  log4j configureAndWatch is started with configurat
ion file: C:\IBM\tivoli\tio/config/log4j-util.prop
2006-09-22 09:16:22,906 INFO  ID=[2222]
2006-09-22 09:16:22,906 INFO  Name=[enterprise]
2006-09-22 09:16:22,906 INFO  TypeId=[3]
2006-09-22 09:16:22,922 INFO  TypeName=[server]
2006-09-22 09:16:22,922 INFO  Locale=[en_US]
2006-09-22 09:16:22,938 INFO  ==================================================
=================
```

9.5.4 Importing data center model definitions

The data center model (DCM) represents the physical and logical assets under and IBM Tivoli Provisioning Manager management, such as switches, load balancers, application software, and more. You can define the DCM objects in an XML file, which you can then import into your DCM using the **xmlimport.cmd** command.

A sample XML file containing a sample DCM is located in <*installation location*>/xml/venice.xml.

Example 9-15 is an example of importing the DCM object definition of one computer.

Example 9-15 Importing a DCM object XML file using xmlimport.cmd

```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ cat ../xml/gees_test_server.xml

<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE datacenter PUBLIC "-//Think Dynamics//DTD XML Import//EN"
    "http://www.thinkdynamics.com/dtd/xmlimport.dtd" [
        <!ENTITY resourcepoolconfig SYSTEM
"http://www.thinkdynamics.com/dtd/res
ourcepool.xml">
        <!ENTITY dataacquisitionconfig SYSTEM
"http://www.thinkdynamics.com/dtd/
dataacquisition.xml">
        <!ENTITY simulator-cpu '
    <dae-driver device="server"
classname="com.thinkdynamics.kanaha.dataacquisit
ionengine.NullDriver"/>
    <dae-signal name="cpu-utilization"
        device="server" metric="cpu-utilization" aggregation="average"
        filter="low-pass-filter"/>
  '>
]>

<datacenter>
        <spare-pool name="Local Server" locale="en_US">
                <server name="ghufrans_test_servero" locale="en_US">
                        <network-interface ipaddress="10.2.1.1.106"
management="
true"></network-interface>
                </server>
                <access-domain-membership>

<access-domain-name>sample:all-objects</access-domain-na
me>
                </access-domain-membership>
        </spare-pool>

</datacenter>

Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ./xmlimport.cmd file:../xml/gees_test_server.xml
2006-09-20 16:58:45,172 INFO  log4j configureAndWatch is started with
configuration file: C:\IBM\tivoli\tio/config/log4j-util.prop
user.dir=[c:\IBM\tivoli\tio\tools]
file:../xml/gees_test_server.xml
2006-09-20 16:59:30,312 INFO  Import took 33593 millisecs
```

9.5.5 Exporting the data center model into an XML file

You can use the **dcmexport.cmd** command to export the data center model into an XML file, as shown in Example 9-16.

Example 9-16 Exporting the DCM into an XML file

```
$ ./dcmexport.cmd -d geesExport.xml
2006-09-22 08:58:26,531 INFO  log4j configureAndWatch is started with configuration
file: C:\IBM\tivoli\tio/config/log4j-util.prop

$ ls -al geesExport.xml
-rwx------+ 1 Administrator Domain Users 1953635 Sep 22 09:04 geesExport.xml
```

Note that the created export file is a lot larger than the original venice.xml, which was imported earlier.

```
Administrator@enterprise /cygdrive/c/IBM/tivoli/tio/tools
$ ls -al ../xml/venice.xml
----rwx---+ 1 tioadmin Administrators 147184 Jul 24 19:53 ../xml/venice.xml
```

Important: An XML export is not a substitute for a database backup and restore. Do not use it as a method for disaster recovery purposes.

The **dcmexport.cmd** command enables you to export the entire data center model to an XML format that conforms to the rules defined in the xmlimport.dtd file.

There are several uses for an exported XML file:

- To obtain information about specific data center objects and the necessary XML syntax to import similar data center objects.
- For troubleshooting purposes—you can send your data center configuration to IBM Tivoli Software Support for analysis.

9.5.6 Pinging the agent manager

As shown in Example 9-17 on page 259, the **ping-agent-manager.cmd** script retrieves the status of the agent manager process.

Example 9-17 Pinging the agent manager

```
$ ./ping-agent-manager.cmd
2006-09-20 17:01:33,938 INFO  log4j configureAndWatch is started with configuration
file: C:\IBM\tivoli\tio\config\log4j-util.prop
20-Sep-2006 17:01:47 com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCacheItem
tryConnect
INFO: NOTE  ==>Connected to host=enterprise.gee.local on port=9513
20-Sep-2006 17:01:47 com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCacheItem
directConnect
INFO: Directly connected
20-Sep-2006 17:01:52 com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCacheItem
tryConnect
INFO: NOTE  ==>Connected to host=enterprise.gee.local on port=9511
20-Sep-2006 17:01:52 com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCacheItem
directConnect
INFO: Directly connected
```

9.5.7 Checking Tivoli Provisioning Manager engine status

As shown in Example 9-18, the **`tioStatus.cmd`** script retrieves the status of the and IBM Tivoli Provisioning Manager engines.

Example 9-18 Checking the status of and IBM Tivoli Provisioning Manager engines

```
$ ./tioStatus.cmd
Checking LWI status...
ACTIVE
SUCCESS
2006-09-20 17:03:18,203 INFO  log4j configureAndWatch is started with configurat
ion file: C:\IBM\tivoli\tio\config\log4j-util.prop
2006-09-20 17:03:18,797 INFO  COPCOM421I The deployment engine is started.
2006-09-20 17:03:26,312 INFO  COPCOM423I The policy engine is started.
2006-09-20 17:03:27,891 INFO  COPCOM484I The agent shell server is started.
```

9.6 Automating Tivoli Provisioning Manager tasks

A key objective of Tivoli Provisioning Manager for Software is to enable coexistence with Tivoli Configuration Manager and to provide a first step towards migrating existing Tivoli Configuration Manager assets and reusing Tivoli Configuration Manager skills in a Tivoli Provisioning Manager for Software environment. In this context, the most significant Tivoli Configuration Manager assets, that you can migrate for reuse on the Tivoli Provisioning Manager for Software side, are software packages, Inventory Profiles, and Activity Plans.

Management of these resources in Tivoli Configuration Manager is now straightforward due to the continuous improvement and stabilization of Tivoli Provisioning Manager and because users' skills in this area have reached very high levels. Tivoli Configuration Manager users developed lots of procedures using the Tivoli Management Framework/Tivoli Configuration Manager command line (w* commands) to automate management tasks (for example, create and distribute software packages, monitors distributions, and so on).

On the other hand, such users might be slowed down when attempting to perform the same tasks in Tivoli Provisioning Manager for Software. The most important inhibitors are:

- The lack of the rich set of commands that are available in the Tivoli Management Framework/Tivoli Configuration Manager environment.
- The lack of best practices on how to migrate management tasks from Tivoli Configuration Manager to Tivoli Provisioning Manager for Software—there are tools and documentation for migrating resources but nothing about efficient ways of managing those resources in Tivoli Provisioning Manager for Software.

Our objective in this section is to give Tivoli Configuration Manager users direction on how to implement some key software distribution scenarios using the SOAP-based command line provided by Tivoli Provisioning Manager for Software.

Looking for a direct one-to-one mapping between w* and Tivoli Provisioning Manager for Software commands is not the best thing to do because the products are totally different. They have different technology and different paradigms for managing resources. For example, while Tivoli Configuration Manager is more command line oriented, Tivoli Provisioning Manager for Software has a powerful graphical user interface that facilitates administrator work. So, a better approach is to provide a set of best practices to implement real-life scenarios in the Tivoli Provisioning Manager for Software environment.

9.6.1 Conceptual overview

In this section, we introduce Tivoli Provisioning Manager for Software concepts and terminology with relation to any corresponding concepts in the Tivoli Configuration Manager realm.

Data center model

The data center model (DCM) is a centralized repository that contains resources and their relationships, which Tivoli Provisioning Manager for Software manages. At first glance you might think of the DCM as the Tivoli object database in the Tivoli Management Framework world, but the DCM is more comprehensive and functional than the object database. For example, the DCM contains hardware and software inventory data of managed computers, while such data is stored in the Tivoli Configuration Manager inventory database.

The DCM uses an object-oriented paradigm to model managed resources—devices: A device is characterized by properties, associations with other devices, and management operations that can be performed on it, which are called Logical Device Operations (LDO). Examples of DCM resources are routers, software products, software agents, and computers. Examples of LDOs on those resources are configure router, install software product, and start agent. LDOs are abstract and do not have an implementation.

Workflow

A workflow is a program written in a proprietary language that implements an LDO to perform a management action on a device. Workflows provide a way to accomplish management tasks on DCM assets. Using the workflow framework, you can automatically manage any data center asset. You can manage any device in your data center, from any vendor, by writing workflows that implement the management operations that act on the real device that is defined in the data center and configuring it.

To provide an analogy with the Tivoli Management Framework world, you can see the workflow as a common object request broker architecture (CORBA) method implementation of a Tivoli object, and the LDO as a CORBA method declaration. In the same way as a CORBA method, an LDO could have several implementations depending on the kind of object.

A workflow need not necessarily implement an LDO; instead, you can write a workflow program to perform a task on the Tivoli Provisioning Manager server to query the DCM and so forth. Workflows can implement complex tasks, such as deployment of an application server. From this point-of-view, you can look at a workflow as a Tivoli Management Framework task run on a Tivoli Management Framework server.

Endpoint task

An endpoint task is a program or script that you schedule to run on an endpoint. The analogy with the Tivoli Management Framework world is that the Tivoli Management Framework task runs on endpoints.

SOAP services

Tivoli Provisioning Manager makes tasks (operations) and resources available as Web services. Specifically, resources are available via Web Services Resource Framework (WS-RF). You can invoke operations using the SOAP command line. Available SOAP commands are used for data center administration and for workflow invocation.

You can submit SOAP commands from the Tivoli Provisioning Manager server or from a remote system.

The analogy with the Tivoli Management Framework world, for what concerns Web services, is interface description language (IDL) interfaces. Tivoli Management Framework is based on the CORBA distributed computing model. Access to Tivoli Management Framework services is provided by w* commands, which wrap invocation to methods described by IDL interfaces.

Software module or software definition/software product

The software module is the DCM asset that describes a software product. Besides describing the software with standard properties, such as name and version, the software module is linked to one or more installable images, which are the bundles deployed to target computers for installing the product.

The same concept in Tivoli Configuration Manager is expressed by the software package profile. The installable image in Tivoli Configuration Manager is the software package block (SPB). Note that only one SPB can be linked to a software profile; therefore, if you have more images depending on the target platform, then you are forced to define many software profiles.

Software Installable

The Software Installable is the installable image of a software product. It can be, for example, a software package block, a Red Hat Package Manager (.rpm), a Microsoft Installer (.msi), an archived file as a .zip, or .tar file.

Many Software Installables can be associated to the same Software Module, where each of them might have a requirement that identifies the installation mechanism used to install it or the target platform. For example, an SPB must have the Software Installation Engine (SIE) as a requirement, and it might also have the “os.name = Windows” to indicate that it is the installable image for the Windows platforms.

Software Resource Template

The Software Resource Template (SRT) is the mechanism for defining different configurations of the same software product. You can assign many SRTs to a software product to indicate that the product can be deployed in different configurations.

You can configure a software package in Tivoli Configuration Manager with the use of variables when creating the software package. Variables are similar to SRTs.

Figure 9-7 is a comparison of the software concepts in Tivoli Configuration Manager and Tivoli Provisioning Manager for Software.

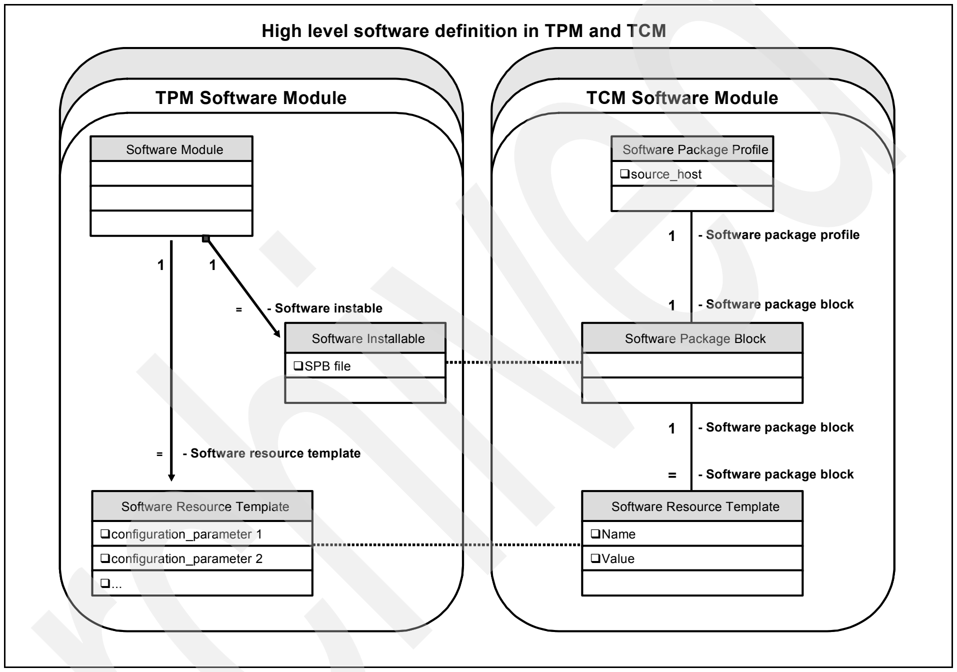


Figure 9-7 Tivoli Configuration Manager - Tivoli Provisioning Manager for Software concepts comparison

Software management operations

Software management operations are operations performed during the life cycle of a software product.

Table 9-3 on page 264 shows a comparison between software management in Tivoli Configuration Manager and in Tivoli Provisioning Manager.

Table 9-3 Comparing Tivoli Configuration Manager and Tivoli Provisioning Manager software deployment operations

Tivoli Configuration Manager Software Operation	**Tivoli Provisioning Manager Software Operation**	**Description**
Load/Unload	Publish/Unpublish	Publish is the operation of making available an installable image to a depot. In Tivoli Configuration Manager the same task is accomplished by loading an SPB onto a Tivoli Management Region Repeater.
Transactional Install	Distribute	Distribute is the operation of sending installable images of a software product to target endpoints. In Tivoli Configuration Manager, you can accomplish a similar task with a transactional install.
Commit	Install	Install is the operation of installing a software product on target endpoints. The same operation is available in Tivoli Configuration Manager. In case of a software package already distributed, the install is equivalent to a Tivoli Configuration Manager commit.
Install	Install	
Remove	Uninstall	Uninstall is the operation of removing the software from target computers.

Publish is the operation of making available an installable image to a depot. In Tivoli Configuration Manager the same task is accomplished loading an SPB onto a Tivoli Management Region Repeater.

Distribute is the operation of sending installable images of a software product to target endpoints. In Tivoli Configuration Manager, you can accomplish a similar task with a transactional install.

Install is the operation of installing a software product on target endpoints. The same operation is available in Tivoli Configuration Manager. In the case of a software package that is already distributed, the install is equivalent to a Tivoli Configuration Manager commit.

Uninstall is the operation of removing the software from target computers.

9.6.2 Configuring the SOAP environment

The Tivoli Provisioning Manager for Software installation provides a pre-configured SOAP environment on the Tivoli Provisioning Manager server. You can run SOAP commands from any computer in the DCM. In this case, be sure that the computer is accessible from Tivoli Provisioning Manager (for example, a service access point is defined), and then configure the SOAP environment as follows:

1. Copy the $TIO_HOME/soapclient directory from the Tivoli Provisioning Manager server to the remote server.
2. Locate and open the SOAP script (that is, **`soapcli.cmd`** for Windows) in the SOAP client directory of the remote server.
3. Set the TIO_LIB variable by replacing the TIO_HOME with the SOAP client directory name.
4. Check if the JAVA_HOME variable is correctly configured (Java Runtime Environment 1.4 or later must be installed). Remove the statement invoking the **`setupCmdLine.cmd`** command.
5. The commands (createByXmlImport.cmd/.sh, findLogDetails.cmd/.sh) listed in the SOAP client directory point to localhost to perform SOAP commands. You must replace localhost with the Tivoli Provisioning Manager server hostname.

The Information Center provides further details. Navigate to the **Tivoli Provisioning Manager v5.1** → **Reference** → **Web Services** → **SOAP Services** → **Running SOAP commands remotely** section.

9.6.3 Configuring the software package's disconnected CLI commands

To build and test software packages on the command line, the software package Editor (SPE) environment has a number of disconnected CLI commands, which begin with **`wd`** and can be found on the Tivoli Provisioning Manager server in the following location:

```
%TIO_HOME%\eclipse\plugins\newSPEditor_4.0.0\swd_cli\bin
```

The following commands are available in this location. These may be familiar to you if you have used the Tivoli Configuration Manager command line interface.

Table 9-4 Software package development and test commands

wchkcond	wdcmmtsp	wdlssp	wdubldsp	wdversp
wdacptsp	wdcrtsp	wdrmvsp	wdundosp	autopack
wdacptsp	wdexptsp	wdsetsps	wdusrpcn	resinit
wdbldspb	wdinstsp	wdswdvar	wdusrprf	--------------

The Tivoli Provisioning Manager for Software's Software Package Editor is very similar to the Tivoli Configuration Manager SPE. It runs as an eclipse-based plug-in that is included in the Automation Package Development Environment (APDE).

To automate the building and testing of software packages, (for example, to create a software package block format) use the **wd*** commands, which we discuss later in this chapter.

Before you use any of these commands, you must define the environment variables shown in Example 9-19 on page 267. You can define the following variables in a command file. A shortcut can be placed on the desktop to run this file in a command window, as shown in Figure 9-8 on page 267.

Example 9-19   Software Package Development command environment file

```
rem Only for Windows Platform
set tools_dir=%TIO_HOME%\eclipse\plugins\newSPEditor_4.0.0\swd_cli
set INTERP=w32-ix86
set SPEDITOR_LIB_PATH=%tools_dir%\lib
set NLSPATH=%tools_dir%\msg_cat\C\spcatalog.cat;%NLSPATH%
set PATH=%PATH%;%tools_dir%\bin;%tools_dir%\lib
set PATHEXT=%PATHEXT%;.exe
set SWD_TDCLI=yes
```

Figure 9-8   Software Package Development CLI shortcut

To use the **wd** commands from a remote computer (not the Tivoli Provisioning Manager server):

1. Copy the %TIO_HOME%\eclipse\plugins\newSPEditor_4.0.0\swd_cli directory to the remote computer.
2. Set the environment.

In Tivoli Provisioning Manager Fix Pack 2 or later, the above steps are unnecessary because the **wd** commands are available under %TIO_HOME%\sie, on the Tivoli Provisioning Manager server.

9.6.4 Setting up a workflow to query the Tivoli Provisioning Manager Data Model

Currently there is no single command line to query the Tivoli Provisioning Manager data model, which is also known as the Data Center Model (DCM).

Although resources can be managed programmatically using Web Services Framework APIs, this approach may be too complicated and time consuming for many of you to use.

One method of querying the DCM is to wrap simple queries in workflows and use SOAP commands to execute these workflows. After the workflows are executed, you can review the query results. This technique is used throughout the end-to-end scenario we use in this chapter.

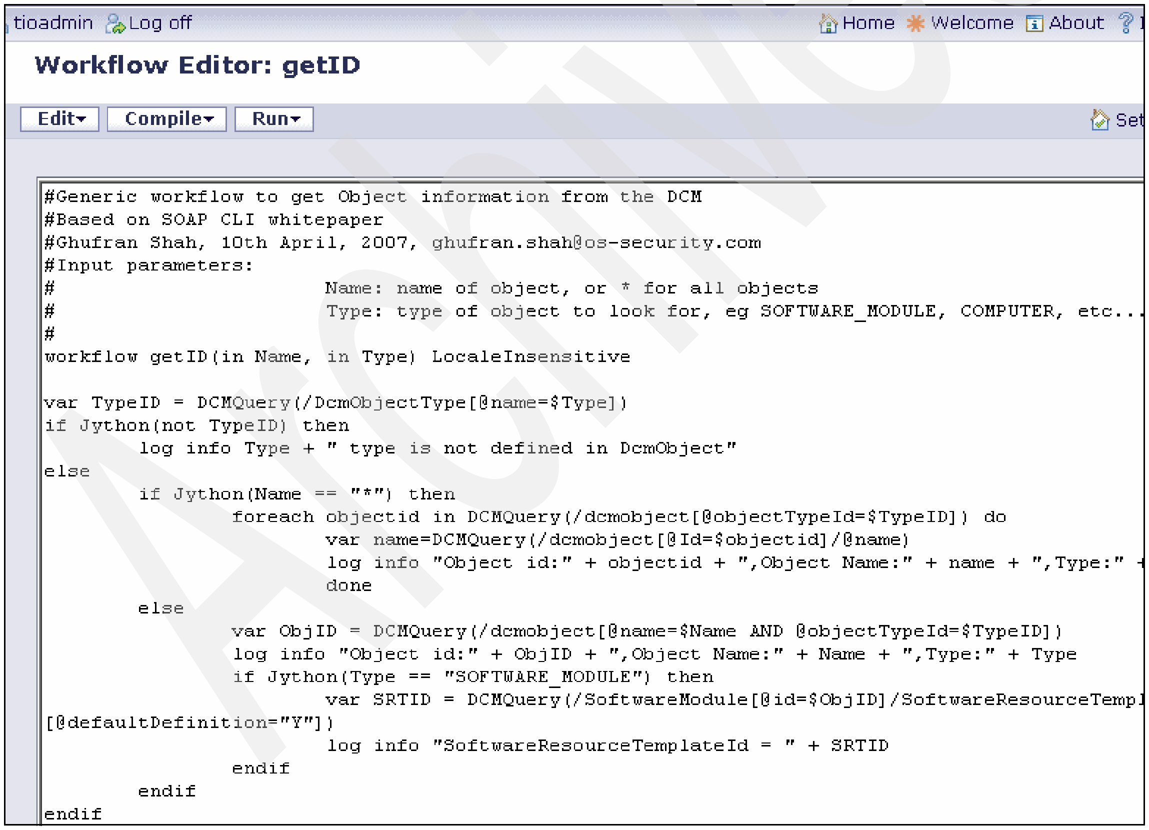
A common requirement is to retrieve the object identifier of one or more objects in the DCM. The workflow shown in Figure 9-9, takes two input parameters:

- Name - This is the name of the object you are looking for or a “*” for all objects.
- Type - This is the type of object you are looking for, for example, COMPUTER, SOFTWARE_MODULE, TASK, and so forth.

  Figure 9-10 on page 269 shows an example of running this workflow.

When the workflow completes, in the workflow execution status log, the object id, object type, and object name are displayed. For SOFTWARE_MODULES, the the default resource template ID of the software is also displayed.

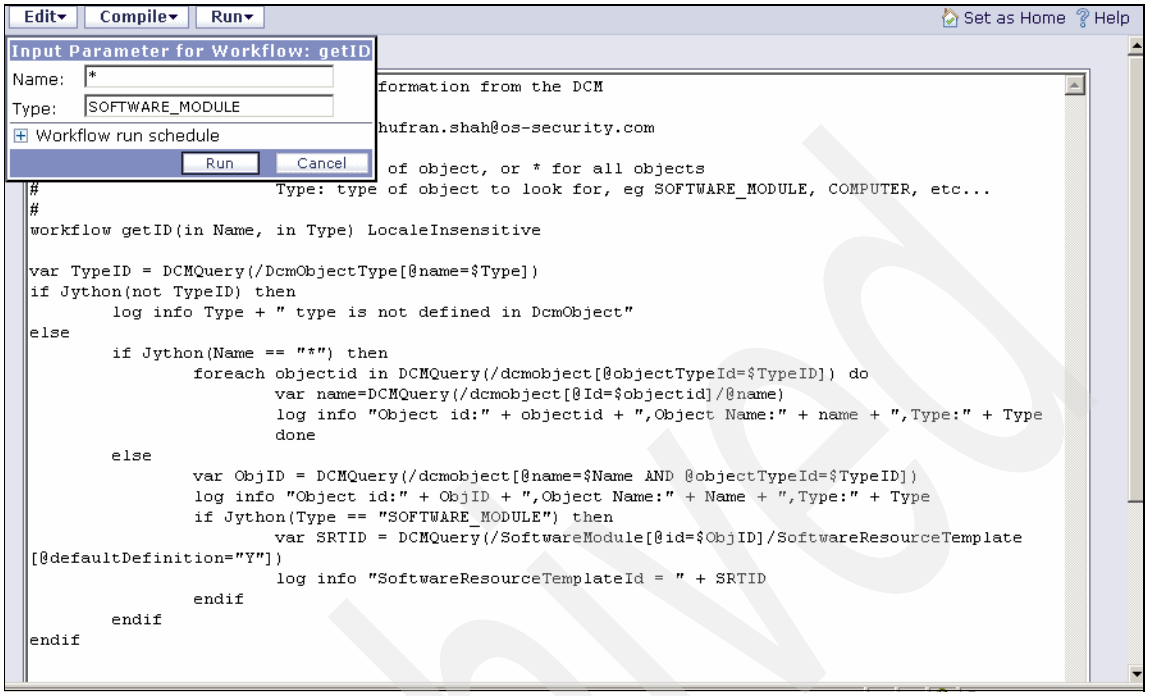
Figure 9-11 on page 270 shows an example of the output produced by this workflow.



```
#Generic workflow to get Object information from the DCM
#Based on SOAP CLI whitepaper
#Ghufran Shah, 10th April, 2007, ghufran.shah@os-security.com
#Input parameters:
#                       Name: name of object, or * for all objects
#                       Type: type of object to look for, eg SOFTWARE_MODULE, COMPUTER, etc..
#
workflow getID(in Name, in Type) LocaleInsensitive

var TypeID = DCMQuery(/DcmObjectType[@name=$Type])
if Jython(not TypeID) then
        log info Type + " type is not defined in DcmObject"
else
        if Jython(Name == "*") then
                foreach objectid in DCMQuery(/dcmobject[@objectTypeId=$TypeID]) do
                        var name=DCMQuery(/dcmobject[@Id=$objectid]/@name)
                        log info "Object id:" + objectid + ",Object Name:" + name + ",Type:" +
                        done
        else
                var ObjID = DCMQuery(/dcmobject[@name=$Name AND @objectTypeId=$TypeID])
                log info "Object id:" + ObjID + ",Object Name:" + Name + ",Type:" + Type
                if Jython(Type == "SOFTWARE_MODULE") then
                        var SRTID = DCMQuery(/SoftwareModule[@id=$ObjID]/SoftwareResourceTempl
[@defaultDefinition="Y"])
                        log info "SoftwareResourceTemplateId = " + SRTID
                endif
        endif
endif
```

Figure 9-9 Workflow to query DCM for object details

Figure 9-10 shows the Type input parameter workflow.

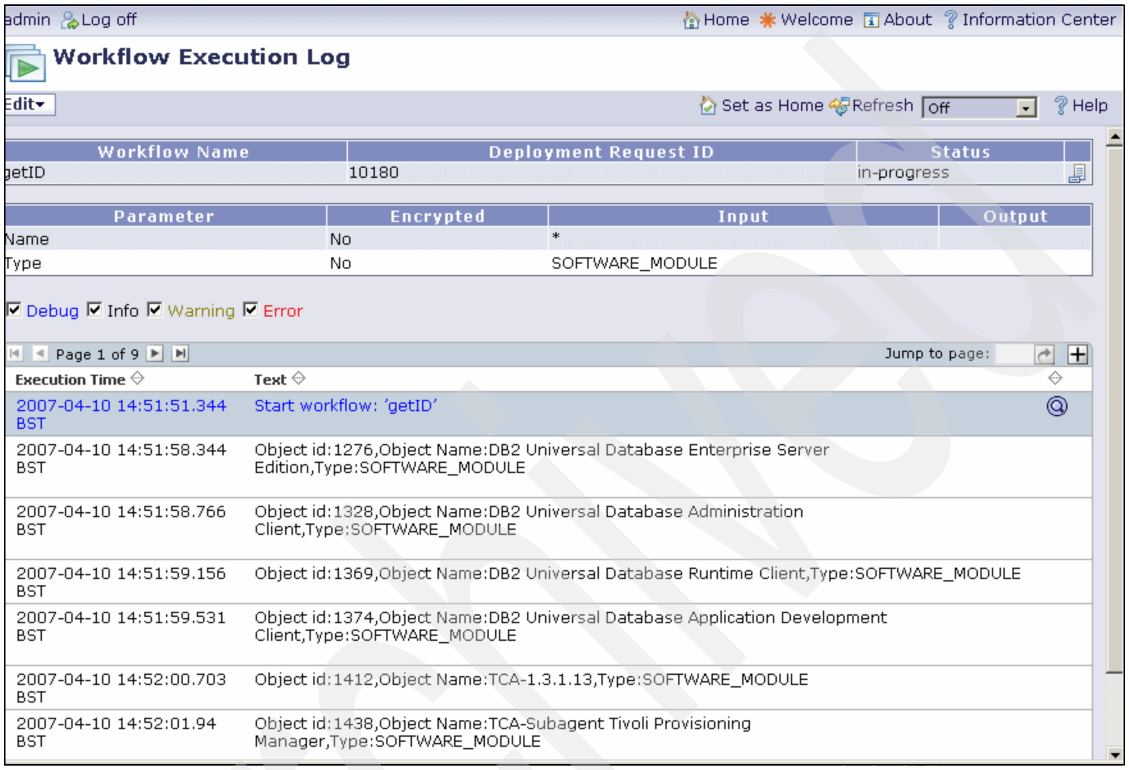


Name: *
Type: SOFTWARE_MODULE

```
formation from the DCM

hufran.shah@os-security.com

 of object, or * for all objects
#                       Type: type of object to look for, eg SOFTWARE_MODULE, COMPUTER, etc...
#
workflow getID(in Name, in Type) LocaleInsensitive

var TypeID = DCMQuery(/DcmObjectType[@name=$Type])
if Jython(not TypeID) then
        log info Type + " type is not defined in DcmObject"
else
        if Jython(Name == "*") then
                foreach objectid in DCMQuery(/dcmobject[@objectTypeId=$TypeID]) do
                        var name=DCMQuery(/dcmobject[@Id=$objectid]/@name)
                        log info "Object id:" + objectid + ",Object Name:" + name + ",Type:" + Type
                        done
        else
                var ObjID = DCMQuery(/dcmobject[@name=$Name AND @objectTypeId=$TypeID])
                log info "Object id:" + ObjID + ",Object Name:" + Name + ",Type:" + Type
                if Jython(Type == "SOFTWARE_MODULE") then
                        var SRTID = DCMQuery(/SoftwareModule[@id=$ObjID]/SoftwareResourceTemplate
[@defaultDefinition="Y"])
                        log info "SoftwareResourceTemplateId = " + SRTID
                endif
        endif
endif
```

Figure 9-10 Running the getID workflow from the Tivoli Provisioning Manager GUI

Figure 9-11is an example of how when the workflow completes, in the workflow execution status log, the object ID, object type, and object name are displayed. For SOFTWARE_MODULES, the default resource template ID of the software is displayed.


admin Log off
Home Welcome About Information Center
Workflow Execution Log
Edit
Set as Home Refresh Off Help

Workflow Name	Deployment Request ID	Status
getID	10180	in-progress

Parameter	Encrypted	Input	Output
Name	No	*	
Type	No	SOFTWARE_MODULE	

☑ Debug ☑ Info ☑ Warning ☑ Error

Page 1 of 9 Jump to page:

Execution Time	Text
2007-04-10 14:51:51.344 BST	Start workflow: 'getID'
2007-04-10 14:51:58.344 BST	Object id:1276,Object Name:DB2 Universal Database Enterprise Server Edition,Type:SOFTWARE_MODULE
2007-04-10 14:51:58.766 BST	Object id:1328,Object Name:DB2 Universal Database Administration Client,Type:SOFTWARE_MODULE
2007-04-10 14:51:59.156 BST	Object id:1369,Object Name:DB2 Universal Database Runtime Client,Type:SOFTWARE_MODULE
2007-04-10 14:51:59.531 BST	Object id:1374,Object Name:DB2 Universal Database Application Development Client,Type:SOFTWARE_MODULE
2007-04-10 14:52:00.703 BST	Object id:1412,Object Name:TCA-1.3.1.13,Type:SOFTWARE_MODULE
2007-04-10 14:52:01.94 BST	Object id:1438,Object Name:TCA-Subagent Tivoli Provisioning Manager,Type:SOFTWARE_MODULE

Figure 9-11 Sample output from the getID workflow

To run the getID workflow from the command line, use the syntax in Example 9-20:

Example 9-20 Running the getID workflow from the command line.

```
%TIO_HOME%\soapclient\tpmlteSoap>soapcli.cmd user password
http://<tpm_server>:8777/ws/pid/TpmLiteSoapService?wsdl
executeDeploymentRequest getID
"Name=software_name,Type=SOFTWARE_MODULE"
```

The following explanation applies to the syntax in Example 9-20:

- *user* and *password* are the credentials to access the Tivoli Provisioning Manager server.

- *tpm_server* is the Tivoli Provisioning Manager server hostname.
- *software_name* is the software definition name to query.

The output of this command provides the identifier of the deployment request that is created to run the workflow. This identifier is then passed into the standard "findLogDetails.cmd" script, to get the workflow execution log, which shows the output of the workflow.

Example 9-21 shows the syntax of the **findLogDetails.cmd** script.

Example 9-21 Running the findLogDetails.cmd script

```
%TIO_HOME%\soapclient\tpmlteSoap>findLogDetails.cmd user password
deployment_request_id
```

The following explanation applies to the syntax for the script in Example 9-21.

- *user* and *password* are the credentials to access the Tivoli Provisioning Manager server.
- *deployment_request_id* is the value returned by the getID workflow execution.

Example 9-22 shows the output of the **findLogDetails.cmd** command.

Example 9-22 Sample output of the findLogDetails.cmd script

```
Log Details:
Total: 5
Start workflow: 'getID'
The Object MySoftware of type SOFTWARE_MODULE has id = 24160
SoftwareResourceTemplateId = 24164
End workflow:
'getID'
```

When you want to retrieve the identifier of a task definition object, you must pass the following parameters to the getID workflow:

```
Name=task_definition_name,Type=TASK_DEFINITION
```

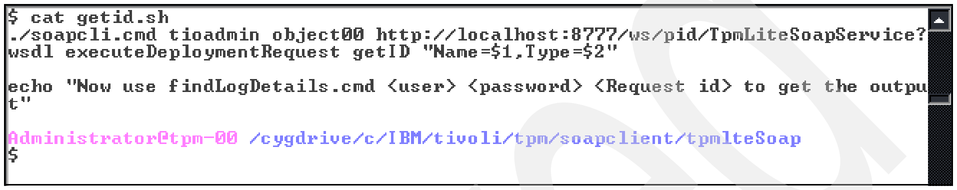
When you want to retrieve the identifier of a file repository object, you must pass the following parameters to the getID workflow:

```
Name=file_repository_name,Type=FILE_REPOSITORY
```

You can use the defined workflow to get the identifier of any object in the DCM. For a computer object, type the following:

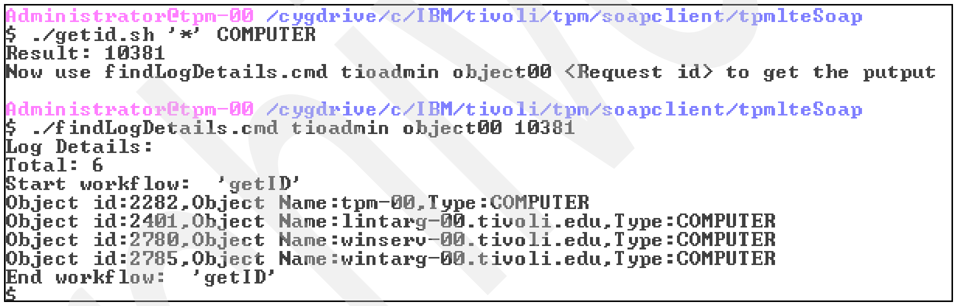
```
Name=computer_name,Type=COMPUTER
```

As you may use these commands regularly, we recommend that you start to build a library of "wrapper" scripts that you can use without having to constantly remember the syntax of each SOAP call. See Figure 9-12 and Figure 9-13 for examples.



```
$ cat getid.sh
./soapcli.cmd tioadmin object00 http://localhost:8777/ws/pid/TpmLiteSoapService?
wsdl executeDeploymentRequest getID "Name=$1,Type=$2"

echo "Now use findLogDetails.cmd <user> <password> <Request id> to get the outpu
t"

Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$
```

Figure 9-12 getid.sh - Example of a wrapper script to call the getID workflow



```
Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$ ./getid.sh '*' COMPUTER
Result: 10381
Now use findLogDetails.cmd tioadmin object00 <Request id> to get the putput

Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$ ./findLogDetails.cmd tioadmin object00 10381
Log Details:
Total: 6
Start workflow:  'getID'
Object id:2282,Object Name:tpm-00,Type:COMPUTER
Object id:2401,Object Name:lintarg-00.tivoli.edu,Type:COMPUTER
Object id:2780,Object Name:winserv-00.tivoli.edu,Type:COMPUTER
Object id:2785,Object Name:wintarg-00.tivoli.edu,Type:COMPUTER
End workflow:  'getID'
$
```

Figure 9-13 Example of running getid.sh and findLogDetails.cmd to query the DCM

9.7 End-to-end scenarios

In 9.6.4, "Setting up a workflow to query the Tivoli Provisioning Manager Data Model" on page 267, we described some of the building blocks of running workflows and interacting with the DCM on the command line. In this section, we describe an end-to-end scenario, from the preparation of a deployment package (software package) to its installation on an end-user workstation. We also describe the uninstall scenario.

The scenario consists of different use cases. We implemented each scenario using the command line. First we describe how the use case is typically implemented in Tivoli Configuration Manager using the w* commands, and then we explain how you can implement the use case in Tivoli Provisioning Manager for Software (or Tivoli Provisioning Manager) using the SOAP command line. A summary at the end of each use case explains alternatives to implement the scenario in Tivoli Provisioning Manager for Software (if any).

9.7.1 Creating a deployment package (software package)

This use case consists of packaging installation and uninstallation procedures into a format understandable by the Configuration Management tool (Tivoli Configuration Manager and Tivoli Provisioning Manager for Software): the software package.

Tivoli Configuration Manager implementation

You can create software packages using the Software Package Editor and then import it into the configuration management tool using the **wimpspo** command.

You can import any software package format with the same command line: SPD, SP, and SPB.

You can also manually create software packages in the ASCII definition using a text editor to create an SPD file. Next, you can use the SPD file to create a software package block (SPB) using the **wdcrtsp** command. You can then import the SPB using the **wimpso** command.

Tivoli Provisioning Manager for Software implementation

Tivoli Provisioning Manager for Software only lets you import SPBs; therefore, you first need to build an SPD. This is why you need to install the wd* commands.

Note: You might still use the Software Package Editor to automatically import software packages into Tivoli Provisioning Manager for Software; however, this chapter focuses on automating tasks using the command line.

You can use either of the following commands to create an SPB.

```
wdcrtsp -f %tmp%\sample\SampleSP.spd %tmp%\sample\SampleSP.spb

wdbldspb -o %tmp%\sample\SampleSP.sp %tmp%\sample\SampleSP.spb
```

The first command, **wdcrtsp**, takes an SPD file as input, while the second command, **wdbldspb**, takes an SP file as input.

When the SPB is available, copy it into a file repository by running the workflow FileRepository.PutFile. Because this workflow takes the identifier of the file repository and the identifier of the computer where the SPB resides (your local computer) as input, to retrieve those IDs, run the getID workflow that we described earlier in Figure 9-12 on page 272 and Figure 9-13 on page 272.

The following command copies the SPB into the file repository by running the FileRepository.PutFile workflow:

Example 9-23 Running the FileRepository.PutFile workflow from the command line

```
%TIO_HOME%\soapclient\tpmlteSoap> soapcli.cmd user password
http://<tpm_server>:8777/ws/pid/TpmLiteSoapService?wsdl
executeDeploymentRequest FileRepository.PutFile
"FileRepositoryID=fr_id,SourceID=s_id,SourcePath=src_path,SourceFileNam
e=spb_name,DestinationPath=dest_path,DestinationFileName=dest_file,Time
outInSeconds=300"
```

The following explanation describes the syntax for the FileRepository.PutFile workflow, which appears in Example 9-23.

- *user password* are the credentials to access the Tivoli Provisioning Manager server.
- <tpm_server> specifies the Tivoli Provisioning Manager server hostname.
- *fr_id* is the DCM identifier related to the file repository in which to copy the SPB file.
- *s_id* specifies the DCM identifier related to the server where the SPB file is located.
- *src_path* is the absolute path where the SPB to copy is located.
- *spb_name* is the SPB file name.
- *dest_path* is the relative path to the File Repository's root path into which the file is copied (that is, to copy it on the File Repository's root path use "/").
- *dest_file* is the destination file name.

Ensure that there are no blank spaces between the workflow arguments ("arg1=value1,arg2=value2"). Also ensure that the paths have "/" character as path-separator.

Now, the SPB is in the file repository and Tivoli Provisioning Manager for Software can reference it in a software definition, as described in the next section.

9.7.2 Importing a deployment package

This use case consists of creating a software package in the Configuration Management tool.

Tivoli Configuration Manager implementation

To import a software package in IBM Tivoli Configuration Manager, call **wimpspo** using either of the following commands.

```
wimpspo -c @ProfileManager:SamplePM -f %tmp%\sample\SampleSP.spd
@SampleSP.1.0

wimpspo -c @ProfileManager:SamplePM -f %tmp%\sample\SampleSP.spd -t
build -p %tmp%\sample\SampleSP.spb @SampleSP.1.0
```

The commands take a non-built software package as input, which is an SPD file. The first command imports the software package as SP format, and the second command imports the software package as SPB.

Tivoli Provisioning Manager for Software implementation

You can import any object into Tivoli Provisioning Manager for Software using the xmlimport command line tool by writing the XML file that describes the object.

The following XML is a template for defining a software product. You can investigate the XML model of other DCM assets by looking at the xmlimport.dtd file in your Tivoli Provisioning Manager for Software.installation on the $TIO_HOME/xml directory.

Example 9-24 XML template for defining a software product

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>
<software-module name="software_module_name" version="version"
vendor="vendor_name"
description="software_module_description" is-draft="false">
<installable-package name="installable_name"
description="Description_of_installable" version="version"
priority="1" file-repository="file_repository_name"
status="not_tested">
<file name=”spb_file_name” path="relative_path_to_root_path" />
<property component="KANAHA" name="BUILT" value="1" />
<software-requirement name="subagent.name"
type="TIVOLI_MANAGEMENT_AGENT"
enforcement="MANDATORY" hosting="true" accept-non-existing="true">
<software-requirement-value value="SIE" />
</software-requirement>
</installable-package>
<software-resource-template name="SRT_Name"
software-resource-type="INSTALLATION"
```

```
multiplicity-type="One" software-configuration-type="Regular"
is-selected="true"
is-default="true">
<template-param name="REMOVE_SPB_AFTER_PROCESS" value="FALSE"
parameter-type="String"
multiplicity-type="Zero_Or_One" is-hidden="false" is-changeable="false"
is-encrypted="false" />
<template-param name="FORCE" description="" value="TRUE_OR_FALSE"
parameter-type="Boolean" multiplicity-type="Zero_Or_One"
is-hidden="false"
is-changeable="true" is-encrypted="false">
<template-param-value value="TRUE" is-default="false" />
<template-param-value value="FALSE" is-default="true" />
</template-param>
…..
<template-param name="Default_Variable" value="VALUE"
parameter-type="String"
multiplicity-type="Zero_Or_One" is-hidden="false" is-changeable="true"
is-encrypted="false" />
<template-param name="Default_Variable1" value="VALUE1"
parameter-type="String"
multiplicity-type="Zero_Or_One" is-hidden="false" is-changeable="true"
is-encrypted="false" />
…..
</software-resource-template>
</software-module>
</datacenter>
```

The following list contains the values (shown in bold in the xml) that you configure to create a new software product using the command line:

- <software-module> tag:
  - name - Specifies the name of the software product to be created. The name must be a unique software product name.
  - version - Specifies the version of the software product.
  - vendor - Specifies who built the software product.
  - description - Gives a description of the software product.
- <installable-package> tag
  - name - Specifies the name of the Software Installable. It must be a unique Software Installable name.
  - description - Gives a description of the Software Installable.
  - version - Specifies the version of the Software Installable.

  - file-repository - Specifies the name of the file repository where the software package is located (LocalFileRepository).
  - <file> tag
    - name - Specifies the name of the SPB file.
    - path - Specifies the relative path to the File Repository's root path of the SPB file (that is, for SPB located in the File Repository's root path, use "/").
- <software-resource-template> tag
  - name - Specifies the name of the Software Resource Template associated to the Software Module that you want to create (that is, SoftResTemplSample).
  - <template-param> tag - In the FORCE option, choose true or false to force the operations performed using this software. For example, you can force the installation of the software already installed on a target computer.
  - <template-param> tag - Defines a list of default variables to use during the SPB installation. You must replace Default_Variable with the % sign followed by the name of the variable used by the SPB and VALUE with the string that expands at the SPB installation time.

To retrieve the list of default variables defined in an SPB, use the **wdexptsp** command available after setting up the wd* command line.

If you saved the XML file in %tmp%\sample\createSoftwareProduct.xml, you can create the software product by running the command in Example 9-25.

Example 9-25 Using the createByXmlImport.cmd script

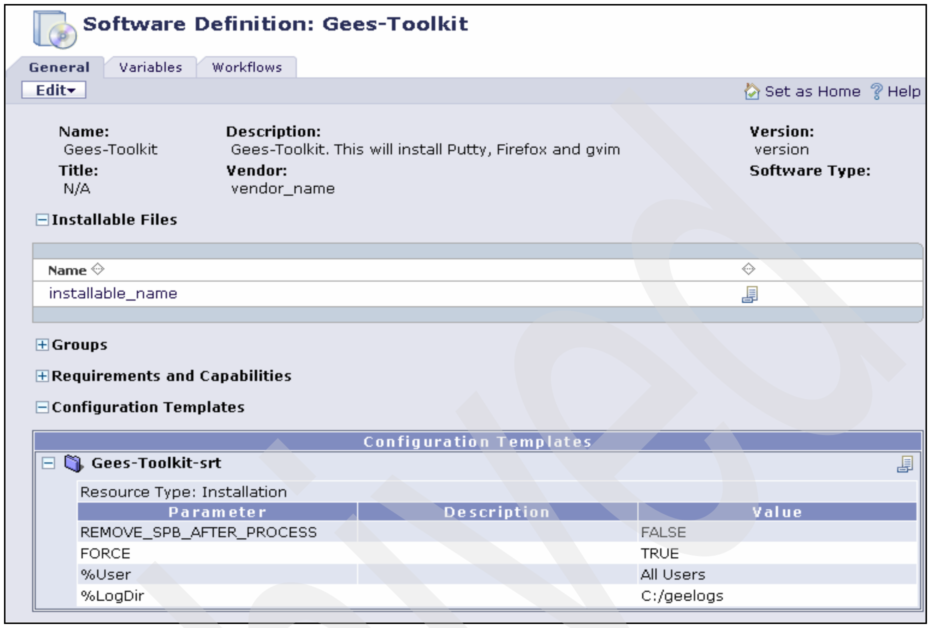
```
%TIO_HOME%\soapclient\tpmlteSoap>createByXmlImport.cmd user password
%tmp%\sample\createSoftwareProduct.xml
```

The XML in Example 9-26, creates the software product “Gees-Toolkit”, as shown in Figure 9-14 on page 279.

Example 9-26 Sample XML to create a software product

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>

<software-module name="Gees-Toolkit" version="version"
vendor="vendor_name" description="Gees-Toolkit. This will install
Putty, Firefox and gvim" is-draft="false">

   <installable-package name="installable_name"
description="Description_of_installable" version="version" priority="1"
file-repository="LocalFileRepository" status="not_tested">
      <file name="tpmtools.spb" path="myspbs" />
      <property component="KANAHA" name="BUILT" value="1" />
      <software-requirement name="subagent.name"
type="TIVOLI_MANAGEMENT_AGENT" enforcement="MANDATORY" hosting="true"
accept-non-existing="true">
      <software-requirement-value value="SIE" />
      </software-requirement>
</installable-package>

   <software-resource-template name="Gees-Toolkit-srt"
software-resource-type="INSTALLATION" multiplicity-type="One"
software-configuration-type="Regular" is-selected="true"
is-default="true">
      <template-param name="REMOVE_SPB_AFTER_PROCESS" value="FALSE"
parameter-type="String" multiplicity-type="Zero_Or_One"
is-hidden="false" is-changeable="false" is-encrypted="false" />
      <template-param name="FORCE" description="" value="TRUE"
parameter-type="Boolean" multiplicity-type="Zero_Or_One"
is-hidden="false" is-changeable="true" is-encrypted="false">
         <template-param-value value="TRUE" is-default="false" />
         <template-param-value value="FALSE" is-default="true" />
      </template-param>
      <template-param name="%User" value="All Users"
parameter-type="String" multiplicity-type="Zero_Or_One"
is-hidden="false" is-changeable="true" is-encrypted="false" />
      <template-param name="%LogDir" value="C:/geelogs"
parameter-type="String" multiplicity-type="Zero_Or_One"
is-hidden="false" is-changeable="true" is-encrypted="false" />
   </software-resource-template>
```

```
</software-module>
</datacenter>
```



Name: Gees-Toolkit
Description: Gees-Toolkit. This will install Putty, Firefox and gvim
Version: version
Title: N/A
Vendor: vendor_name
Software Type:

Installable Files

Name
installable_name

Groups

Requirements and Capabilities

Configuration Templates

Configuration Templates

Gees-Toolkit-srt

Resource Type: Installation

Parameter	Description	Value
REMOVE_SPB_AFTER_PROCESS		FALSE
FORCE		TRUE
%User		All Users
%LogDir		C:/geelogs

Figure 9-14 Sample software definition created—createByXmlImport.cmd

9.7.3 Distributing a deployment package

This use case consists of distributing (not installing) software installable images (the bulk data) to the endpoints. You can submit an installation task as soon as all endpoints have their copy of the data.

Tivoli Configuration Manager implementation

Perform a transactional install to distribution a software package in Tivoli Configuration Manager. Example 9-27 shows a command for distributing the software package jiga-soft on the computer jiga-box.

Example 9-27 Using winstsp

```
winstsp -ty @SoftwarePackage:jiga-soft @Endpoint:jiga-box
```

Tivoli Provisioning Manager for Software implementation

Tivoli Provisioning Manager for Software downloads binaries to target endpoints with a distribution operation. You must use this operation to perform a transactional install on Tivoli Provisioning Manager for Software.

In Tivoli Provisioning Manager for Software, operations are called tasks, and tasks are stored in the DCM. To create a software distribution task to a group of endpoints, we use the **xmlimport** command to import the task definition shown in Example 9-28.

You can schedule a task for execution. If you do not need to schedule the execution, you can submit it later using the SOAP command line.

To create and submit a software distribution task, configure the following XML template shown in Example 9-28.

Example 9-28   XML template to create and submit a software distribution task

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>

<task-job name="task_job_name" type="DistributeSoftwareTask">
<task-job-item workflow-name="SoftwareModule.DistributeInstallable"
device-parameter-name="DeviceID">
<task-argument position="0" argument-name="DeviceID"
argument-value="software_module_id" encrypted="false" />
<task-argument position="0" argument-name="SoftwareModuleID"
argument-value="software_module_id" encrypted="false" />
</task-job-item>
</task-job>

<tpm-task name="name" type="TpmTask" locale="en_US"
task-job-name="task_job_name" scheduled-time="scheduled_time"
created-by-user="user"
last-updated-by="user" concurrency-level="1">
<task-meta-target dcm-object-name="computer_name1" />
<task-meta-target dcm-object-name="computer_name2" />
</tpm-task>
</datacenter>
```

The following list contains the values that you must configure:

- <task-job> tag
  – name - Specifies the name of the new task job to be created.
- <task-argument> tag for both DeviceID and SoftwareModuleID argument-name
  – argument-value - Specifies the identifier related to the software module to be distributed. (Use the getID workflow to retrieve the SoftwareModuleID.)

- <tpm-task> tag
  - name - Specifies the name of the Tivoli Provisioning Manager task to be created.
  - task-job-name - Specifies the name of the task job to submit (that is, my_task_job).
  - scheduled-time - Specifies when the task job will be submitted. The time-format is the following: YYYY-MM-DD HH:MM:mm:SS.d (that is, 2006-11-29 18:00:00.0) If this value is omitted, the task can be submitted only upon the request of the user running the TpmTaskDefinition.execute workflow. Read further for more details.
  - created-by-user - Specifies the name of the user who created the task (that is, tioappadmin).
  - last-updated-by - Specifies the name of the user who updated the task (that is, tioappadmin).
- <task-meta-target> tag
  - dcm-object-name - Specifies the name of the computer on which the job will be submitted. This computer must already exist as a computer object in the DCM. Multiple computers can be specified by having multiple <task-meta-target> tags.

After configuring the XML with actual values, use *createByXmlImport* to define the XML in the DCM, as shown in Example 9-25 on page 277.

If you specified the scheduled-time, the task runs at that time. If you did not specify the scheduled-time, then you can still run the task by invoking the TpmTaskDefinition.execute workflow as shown in Example 9-29.

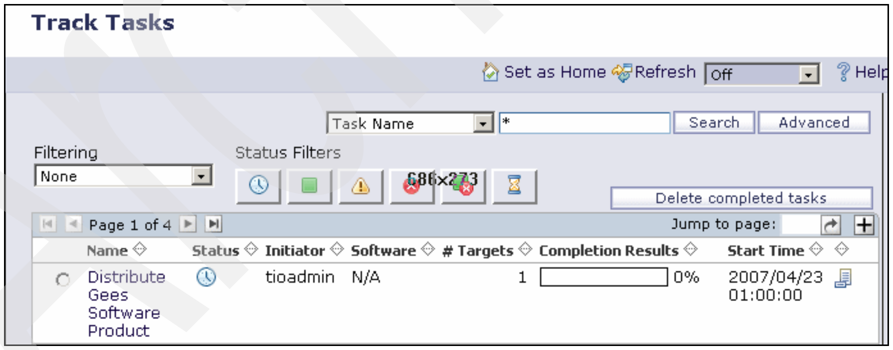
Example 9-29 Invoking the TpmTaskDefinition.execute workflow

```
%TIO_HOME%\soapclient\tpmlteSoap> soapcli.cmd user password
http://<tpm_server>:8777/ws/pid/TpmLiteSoapService?wsdl
executeDeploymentRequest TpmTaskDefinition.Execute
"TaskDefinitionID=ID"

(The ID value in bold in the snippet is the identifier of the task to
submit. Use the getID to obtain the ID value by task name.)
```

The XML shown in Example 9-30 creates and submits the task “Distribute Gees Software Product”, as shown in Figure 9-15.

Example 9-30 Sample XML to create and submit a Software Distribution task on a target

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>

<task-job name="DistributeToolkit" type="DistributeSoftwareTask">
   <task-job-item workflow-name="SoftwareModule.DistributeInstallable"
device-parameter-name="DeviceID">
      <task-argument position="0" argument-name="DeviceID"
argument-value="3281" encrypted="false" />
</task-job-item>

</task-job>

<tpm-task name="Distribute Gees Software Product" type="TpmTask"
locale="en_US" task-job-name="DistributeToolkit" scheduled-time="now"
created-by-user="tioadmin" last-updated-by="tioadmin"
concurrency-level="1">
   <task-meta-target dcm-object-name="wintarg-00.tivoli.edu" />
</tpm-task>

</datacenter>
```



Name	Status	Initiator	Software	# Targets	Completion Results	Start Time
Distribute Gees Software Product		tioadmin	N/A	1	0%	2007/04/23 01:00:00

Figure 9-15 Software Distribution task created by an XML Import

9.7.4 Installing a deployment package

This use case consists of running installation procedures on the target endpoint.

Tivoli Configuration Manager implementation

In the previous scenario, we performed a transactional install. From an IBM Tivoli Configuration Manager perspective, to install a software package is to commit the installation by using the **winstsp** command.

Example 9-31 Using winstsp

```
winstsp -ty @SoftwarePackage:jiga-soft @Endpoint:jiga-box
```

Tivoli Provisioning Manager for Software implementation

Create an install task in the DCM, and schedule it for execution to install a software product in Tivoli Provisioning Manager for Software. The steps are the same as the steps we described in the previous use case, and the only thing that changes is the XML file to customize.

Example 9-32 XML template for installing a software package

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>
<task-job name="task_job_name" type="InstallSoftwareTask">
<task-job-item workflow-name="SoftwareModule.Install"
device-parameter-name="DeviceID">
<task-argument position="0" argument-name="DeviceID"
argument-value="software_module_id" encrypted="false" />
<task-argument position="0" argument-name="SoftwareModuleID"
argument-value="software_module_id" encrypted="false" />
<task-argument position="3" argument-name="SoftwareResourceTemplateID"
argument-value="software_resource_template_ID" encrypted="false" />
</task-job-item>
</task-job>
<tpm-task name="tpm_task_name" type="TpmTask" locale="en_US"
task-job-name="task_job_name" scheduled-time="scheduled_time"
created-by-user="user"
last-updated-by="user" concurrency-level="1">
<task-meta-target dcm-object-name="computer_name1" />
<task-meta-target dcm-object-name="computer_name2" />
</tpm-task>
</datacenter>
```

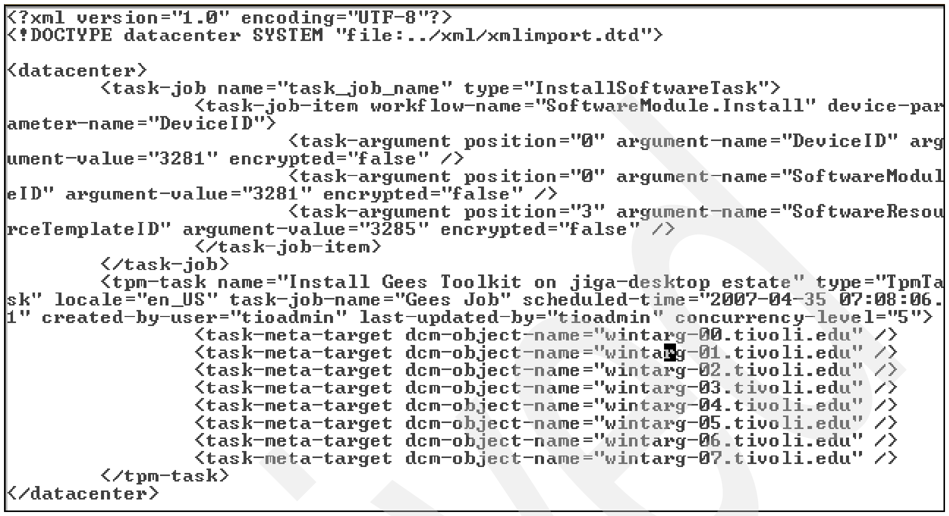
The following list contains the values that you must configure to create a new task that installs a software product on a list of computers by the command line:

- <task-job> tag
  - name - Specifies the name of the new task job to be created.
- <task-argument> tag for both DeviceID and SoftwareModuleID argument-name
  - argument-value - Specifies the identifier related to the software module to be installed.
- <task-argument> tag for SoftwareResourceTemplateID argument-name
  - argument-value - Specifies the identifier related to the software resource template belonging to the software module to be installed.
- <tpm-task> tag
  - name - Specifies the name of the Tivoli Provisioning Manager for Software task to be created.
  - task-job-name - Specifies the name of the task job to submit.
  - scheduled-time - Specifies when the task job is submitted. The time-format is the following: YYYY-MM-DD HH:MM:mm:SS.d (that is, 2006-11-29 18:00:00.0). If this value is omitted, the task can be submitted only upon the request of the user by running the TpmTaskDefinition.execute workflow.
  - created-by-user - Specifies the name of the user who created the task.
  - last-updated-by - Specifies the name of the user who updated the task.
- <task-meta-target> tag
  - dcm-object-name - Specifies the name of the computer on which the job will be submitted. This computer must already exist as a computer object in the DCM. You can specify multiple computers by having multiple <task-meta-target> tags.

After you configure this XML file, you can create the task in the DCM as before by using the *createByXmlImport* command, as shown in Example 9-25 on page 277.

If you did not specify scheduling information, you can invoke the workflow TpmTaskDefinition.execute, as shown in Example 9-29 on page 281.

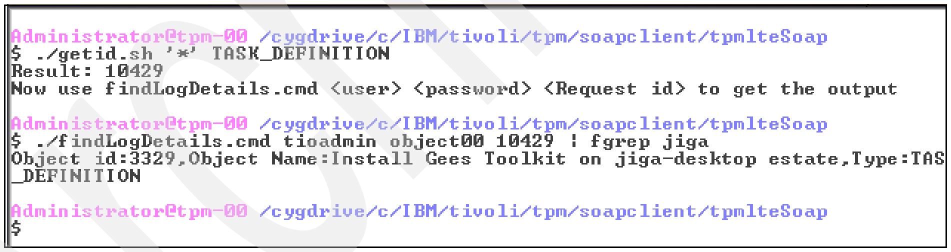
The XML in Figure 9-16 creates and schedules a task to install a software package on multiple targets.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">

<datacenter>
        <task-job name="task_job_name" type="InstallSoftwareTask">
                <task-job-item workflow-name="SoftwareModule.Install" device-par
ameter-name="DeviceID">
                        <task-argument position="0" argument-name="DeviceID" arg
ument-value="3281" encrypted="false" />
                        <task-argument position="0" argument-name="SoftwareModul
eID" argument-value="3281" encrypted="false" />
                        <task-argument position="3" argument-name="SoftwareResou
rceTemplateID" argument-value="3285" encrypted="false" />
                </task-job-item>
        </task-job>
        <tpm-task name="Install Gees Toolkit on jiga-desktop estate" type="TpmTa
sk" locale="en_US" task-job-name="Gees Job" scheduled-time="2007-04-35 07:08:06.
1" created-by-user="tioadmin" last-updated-by="tioadmin" concurrency-level="5">
                <task-meta-target dcm-object-name="wintarg-00.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-01.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-02.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-03.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-04.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-05.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-06.tivoli.edu" />
                <task-meta-target dcm-object-name="wintarg-07.tivoli.edu" />
        </tpm-task>
</datacenter>
```

Figure 9-16 Sample XML to install a software package on multiple targets

To ensure that the task was created successfully, use the **`getid.sh`** and **`findLogDetails.cmd`** scripts, as shown in Figure 9-17.



```
Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$ ./getid.sh '*' TASK_DEFINITION
Result: 10429
Now use findLogDetails.cmd <user> <password> <Request id> to get the output

Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$ ./findLogDetails.cmd tioadmin object00 10429 | fgrep jiga
Object id:3329,Object Name:Install Gees Toolkit on jiga-desktop estate,Type:TAS
_DEFINITION

Administrator@tpm-00 /cygdrive/c/IBM/tivoli/tpm/soapclient/tpmlteSoap
$
```

Figure 9-17 Verifying that a task was created successfully in the DCM

9.7.5 Monitoring tasks

After you schedule a task, you probably want to monitor its execution.

Tivoli Configuration Manager implementation

IBM Tivoli Configuration Manager users use the **`wmdist`** command to monitor software distribution operations, as shown in Example 9-33 on page 286.

Example 9-33   Tivoli Configuration Manager distribution status tracking

```
wmdist –l -a

Name Distribution ID Targets Completed Successful Failed
=======================================================================
jiga-soft (install) 1125417541.786 2 2(100%) 2(100%) 0( 0%)

An install operation with distribution ID 1125417541.786 has been
successfully performed on two target endpoints.
```

Tivoli Provisioning Manager for Software implementation

You can monitor the status of the operation using the Task Details page of the Web User Interface, but you cannot use the SOAP command line. To list all of the submitted tasks, click **Task Management** → **Track Tasks**, and then click the specific task to get detailed information.

9.7.6 Uninstalling a software package

This use case consists of running back-out procedures on the target endpoint, which you can perform to take the endpoint to the back-level in case the installation malfunctions or in case the software is no longer needed and must be uninstalled from the endpoints.

Tivoli Configuration Manager implementation

The Tivoli Configuration Manager operation for uninstalling a software package is provided with the **`wremovsp`** command.

Example 9-34   Using wremovsp

```
wremovsp @SoftwarePackage:jiga-soft @Endpoint:jiga-box

(This operation uninstalls SampleSP from computer Computer-EP by
running uninstall procedures provided in SPB.)
```

Tivoli Provisioning Manager for Software implementation

You must create and schedule an uninstall task for removing software from computers, by customizing an XML file. If you do not specify scheduling information, you have to run the TpmTaskDefinition.execute workflow to get the job done.

Example 9-35 XML template for unininstallng a software package

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>
<task-job name="task_job_name" type="UninstallSoftwareTask">
<task-job-item workflow-name="SoftwareModule.Uninstall"
device-parameter-name="DeviceID">
<task-argument position="0" argument-name="DeviceID"
argument-value="software_module_id "
encrypted="false" />
<task-argument position="0" argument-name="SoftwareModuleID"
argument-value="software_module_id"
encrypted="false" />
</task-job-item>
</task-job>

<tpm-task name="tpm_task_name" type="TpmTask" locale="en_US"
task-job-name="task_job_name"
scheduled-time="scheduled_time" created-by-user="user"
last-updated-by="user" concurrency-level="1">
<task-meta-target dcm-object-name="computer_name1" />
<task-meta-target dcm-object-name="computer_name2" />
</tpm-task>
</datacenter>
```

The following list contains the values that for you to configure to create a new task that uninstalls a software module on a list of computers using the command line:

- <task-job> tag
  - name - Specifies the name of the new task job to be created.
- <task-argument> tag for both DeviceID and SoftwareModuleID argument-name
  - argument-value - Specifies the identifier related to the software module to be uninstalled.
- <tpm-task> tag
  - name - Specifies the name of the Tivoli Provisioning Manager for Software task to be created.
  - task-job-name - Specifies the name of the task job to submit.
  - scheduled-time - Specifies when the task job will be submitted. The time-format is the following: YYYY-MM-DD HH:MM:mm:SS.d (that is, 2006-11-29 18:00:00.0) If this value is omitted, the task can be submitted

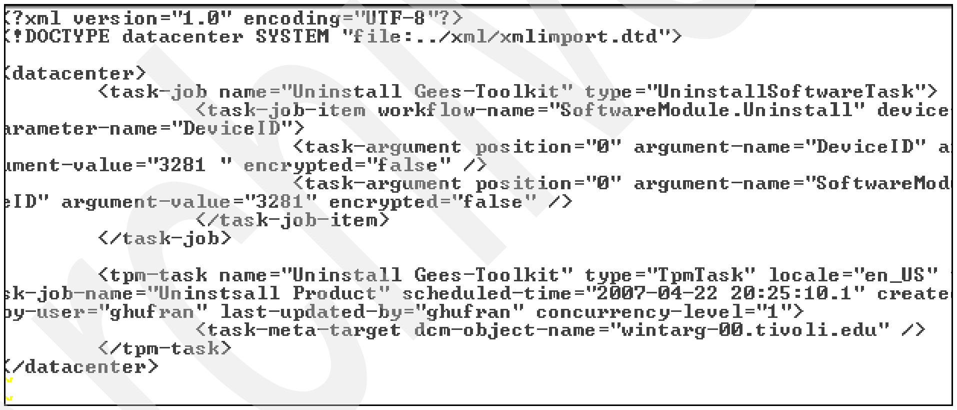
only upon the request of the user by running TpmTaskDefinition.execute workflow.

  – created-by-user - Specifies the name of the user who created the task.
  – last-updated-by - Specifies the name of the user who updated the task.

- <task-meta-target> tag
  – dcm-object-name - Specifies the name of the computer on which the job will be submitted. This computer must already exist as a computer object in the DCM. You can specify multiple computers by having multiple <task-meta-target> tags.

After you configure this XML file, you can create the task in the DCM, as before, by using the **createByXmlImport** command (as shown in Example 9-25 on page 277). If you did not specify scheduling information, you can invoke the workflow TpmTaskDefinition.execute, as shown in Example 9-29 on page 281.

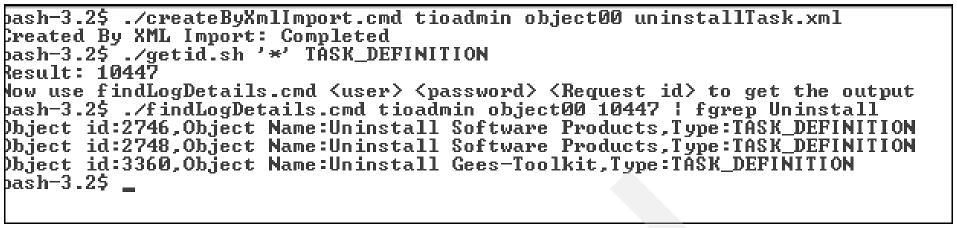
The XML in Figure 9-18 creates and schedules a task to uninstall a software package from one target.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">

<datacenter>
        <task-job name="Uninstall Gees-Toolkit" type="UninstallSoftwareTask">
                <task-job-item workflow-name="SoftwareModule.Uninstall" device
arameter-name="DeviceID">
                        <task-argument position="0" argument-name="DeviceID" a
ument-value="3281 " encrypted="false" />
                        <task-argument position="0" argument-name="SoftwareMod
eID" argument-value="3281" encrypted="false" />
                </task-job-item>
        </task-job>

        <tpm-task name="Uninstall Gees-Toolkit" type="TpmTask" locale="en_US"
sk-job-name="Uninstsall Product" scheduled-time="2007-04-22 20:25:10.1" create
by-user="ghufran" last-updated-by="ghufran" concurrency-level="1">
                <task-meta-target dcm-object-name="wintarg-00.tivoli.edu" />
        </tpm-task>
</datacenter>
```

Figure 9-18   Sample XML to uninstall a software package.

To ensure that the task was created successfully, you can use the **getid.sh** and **findLogDetails.cmd** scripts, as shown in Figure 9-19 on page 289.



```
bash-3.2$ ./createByXmlImport.cmd tioadmin object00 uninstallTask.xml
Created By XML Import: Completed
bash-3.2$ ./getid.sh '*' TASK_DEFINITION
Result: 10447
Now use findLogDetails.cmd <user> <password> <Request id> to get the output
bash-3.2$ ./findLogDetails.cmd tioadmin object00 10447 | fgrep Uninstall
Object id:2746,Object Name:Uninstall Software Products,Type:TASK_DEFINITION
Object id:2748,Object Name:Uninstall Software Products,Type:TASK_DEFINITION
Object id:3360,Object Name:Uninstall Gees-Toolkit,Type:TASK_DEFINITION
bash-3.2$ _
```

Figure 9-19 Verifying that a task was created successfully in the DCM

9.8 Summary

In this section, we followed the implementation of an end-to-end software distribution scenario in Tivoli Provisioning Manager for Software with a two-fold purpose:

- To introduce concepts related to software management in Tivoli Provisioning Manager for Software.
- To describe what features Tivoli Provisioning Manager for Software makes available through the command line and how they can be accessed.

We discussed how to define a software product in Tivoli Provisioning Manager for Software, for example, you need to produce three entities:

- A software definition – name and version of the software product.
- An installable image – in our scenario we only described how to define a software whose installable is an SPB, but you can define any file as the installable, and it will be distributed to the endpoints using the dynamic content delivery service component embedded in Tivoli Provisioning Manager for Software.
- One or more Software Resource Templates (SRT) – the SRT defines a configuration of the software product. You might define several configurations of the software by associating SRTs. In our scenario we explained how to map Tivoli Configuration Manager variables into SRTs.

We also discussed how to create objects. Objects are created in Tivoli Provisioning Manager for Software by importing them into the DCM. After you write information about a software product into an XML file, the XML file is loaded into the XML using the **`xmlimport`** command.

Finally, we also discussed how to run workflows from the command line.

These form the building blocks for automating common Tivoli Provisioning Manager for Software tasks using the command line.

10

Working in a firewall environment

In this chapter, we discuss the working of Tivoli Provisioning Manager for Software in an environment that has a firewall. We introduce the different components that are available in the product to transport the data streams safely and securely over one or more firewalls. We recapitulate the different ports that are used in the Software Distribution Infrastructure (SDI) of a standard Tivoli Provisioning Manager for Software infrastructure.

This chapter has the following sections:

- “Ports used in the Software Distribution Infrastructure” on page 292
- “Tivoli Provisioning Manager for Software firewall support” on page 293
- “Installing and configuring firewall components” on page 299

10.1 Ports used in the Software Distribution Infrastructure

In Table 10-1, we list the default ports used in SDI on the Tivoli Provisioning Manager for Software server management side.

Table 10-1 SDI default ports server side

Port	Management side	Description
8777	Tivoli Provisioning Manager	Web service port that the Tivoli Provisioning Manager Authenticator uses in dynamic content delivery service.
9045	Dynamic content delivery service/device management service	Server authentication
9046	Dynamic content delivery service/device management service	Server and client authentication
9511	Agent Manager	Agent registration
9512	Agent Manager	Report agent health and IP address. Obtain and renew certificates. Query agent information.
9513	Agent Manager	Agent recovery server (unsecure)

Table 10-2 shows the default ports used on the Tivoli Common Agent.

Table 10-2 Default ports on TCA—endpoint side

Port	Usage	Description
9510	Tivoli Common Agent	The port on which the common agent listens.
2100	Dynamic content delivery service	The management center uses this port to send messages to the depot servers. Depot servers listen for these messages on this port.

Port	Usage	Description
2101	Dynamic content delivery service	This is the client's peer port. Clients listen on this port for requests from other clients. It is only used if peering is enabled.
9514	TCA	Non-stop service (local communication)
9515	TCA	Non-stop service (local communication)

The ports described in Table 10-3 are used for discovery purposes and may be used when installing the common agent from Tivoli Provisioning Manager for Software. They are not essential to manage the installed agents.

Table 10-3 Discovery ports

Port	Description
139	Port used for discovery and Remote Execution Access (RXA) on SMB systems
22	Port used by discovery and RXA execution on SSH systems

10.2 Tivoli Provisioning Manager for Software firewall support

Tivoli Provisioning Manager for Software provides basic firewall support. As described in 10.1, “Ports used in the Software Distribution Infrastructure” on page 292, Tivoli Provisioning Manager for Software uses a documented limited fixed number of ports. All ports are configurable and all communication is encrypted and authenticated.

The best solution is to allow the SDI ports traversal through the firewall. Open the ports listed in Table 10-4 on page 294 on the firewall between Tivoli Provisioning Manager for Software server components and the Tivoli Common Agents.

Table 10-4 Scalable Distribution Infrastructure ports

Port	Direction	Usage
9045	Inbound to Tivoli Provisioning Manager for Software server	Device Management Service (DMS)/dynamic content delivery service
9046	Inbound to Tivoli Provisioning Manager for Software server	DMS/dynamic content delivery service
9011	Inbound to Tivoli Provisioning Manager for Software server	Agent Manager
9012	Inbound to Tivoli Provisioning Manager for Software server	Agent Manager
9013	Inbound to Tivoli Provisioning Manager for Software server	Agent Manager
2100	Inbound to depot	common agent connecting depot
2101	Inbound to peer	Optional—peers are most likely placed on same side of the firewall
2100	Outbound to every depot	Management Center to depot
9510	Outbound to TCA	Optional, ping agent, install bundle

In some circumstances, opening these ports is not desirable, or in the case of traversal through several firewalls, impossible. For these customers, Tivoli Provisioning Manager for Software provides basic firewall support. Customers familiar with Tivoli Security Compliance Manager will recognize the solution, as it is based off the components of that product.

To support Tivoli Management Framework endpoints, no additional configuration is needed when the Tivoli Security Firewall Toolkit (TSFT) is in place. Tivoli Provisioning Manager for Software initiated software distribution, inventory or task related operations, use the TSFT unmodified.

10.2.1 Firewall components

The firewall support of Tivoli Provisioning Manager for Software consists of three components, which are all packaged in the same Java JAR file:

Gateway Manager	Placed on the trusted side of the firewall, and establishes connections between the Tivoli Provisioning Manager for Software server components and the Gateway Service or the Proxy Relay.
Gateway Service	Placed on the non-trusted side of the firewall, and establishes the connection between the Tivoli Common Agents and the Gateway Manager or the Proxy Relay.
Proxy Relay	Placed between firewalls, and establishes connections between Gateway Manager and Gateway Service.

Each Gateway Manager can connect to and manage one or more Gateway Services. In addition, one or more Gateway Managers can manage each Gateway Service. Thus Gateway Managers and Gateway Services can have many to many connections. Multiple Proxy Relays can be chained to tunnel over multiple network discontinuities.

The Gateway Manager or Proxy Relay communicates from one port, by default 1960, to a Gateway Service or Proxy Relay on one port, which is by default 1961.

Communication is always initiated outbound, from Gateway Manager to Gateway Service or Proxy Relay and from Proxy Relay to Gateway Service or another Proxy Relay.

By default, the Gateway Service listens on incoming connections from the Gateway Manager or Proxy Relay on port 1961. The Proxy Relay listens on incoming connections from Gateway Manager or another Proxy Relay on port 1960.

Gateway Manager and Proxy Relay use ephemeral ports as a source port for outbound connections.

Note: As from Tivoli Provisioning Manager for Software Fix Pack 2, you can change the inbound Gateway Service port from the default value of 1961, by providing a different port on the command line.

When the Gateway Manager and Gateway Service are operating correctly, there is a persistent TCP/IP connection from the manager to the service. This Command Channel allows the Gateway Service to notify the Gateway Manager when it receives a new connection request. A periodic heartbeat signal is sent to keep the connection alive. If the Command Channel is closed, the Gateway Manager attempts to reconnect, periodically, every 60 seconds.

When the Gateway Service receives a connection request from a common agent for a particular service, it forwards this request through the Command Channel to the Gateway Manager, which in turn creates the connection to the service provider. The input and output streams are then on both ends connected, and communication flows by a thread that reads data from an input stream and writes that data to another output stream. The Gateway Service automatically stops listening on that particular Gateway Manager's service ports when the connection is broken.

You can configure the Gateway Service to advertise a particular service. To utilize this feature, you must enable user datagram protocol (UDP) broadcasting for the local subnet. A common agent can discover a service by sending a broadcast UDP packet that contains the service's name. If the Gateway Service receives that UDP packet, and it currently advertises that service name, it responds back with an addressed UDP packet that contains the port number on which the service is listening. The endpoint can then use the source address of the UDP packet and the port contained within the packet to connect to the given service.

Note: Instead of UDP discovery, you can configure the common agent to connect to the Tivoli Provisioning Manager for Software Web services, by pointing the common agent directly to the Gateway Service, which does prevent ‘dynamic’ discovery of the nearest Gateway Services when you move mobile computers between sites or networks.

The Proxy Relay acts as a go-between Gateway Manager and Gateway Service, which uses the same mechanisms, except UDP discovery, which allows it to hop several firewalls.

Note: The dynamic content delivery service management center—by default running on the Tivoli Provisioning Manager for Software server—must be able to directly communicate to all depot servers, which you cannot tunnel through the firewall support components. The communication requirement is uni-directional, from the Tivoli Provisioning Manager for Software server to the depot.

Note: Peer transfers are not supported using the firewall components. Direct connectivity is required between peers.

Each active communication channel uses one thread on a Gateway Manager, Gateway Service, or Proxy Relay. All traffic over the Command Channel is secure sockets layer (SSL) encrypted.

10.2.2 Example architectures

In this section, we show examples of possible firewall configurations.

The environment on Figure 10-1, is the simplest firewall to set up, where one Gateway Manager talks to one Gateway Service. Port 1961 (by default) needs to be open between Gateway Manager and Gateway Service systems. If depots are located on the non-trusted left side of the firewall, then port 2100 needs to be opened between the Tivoli Provisioning Manager for Software server and each depot server.

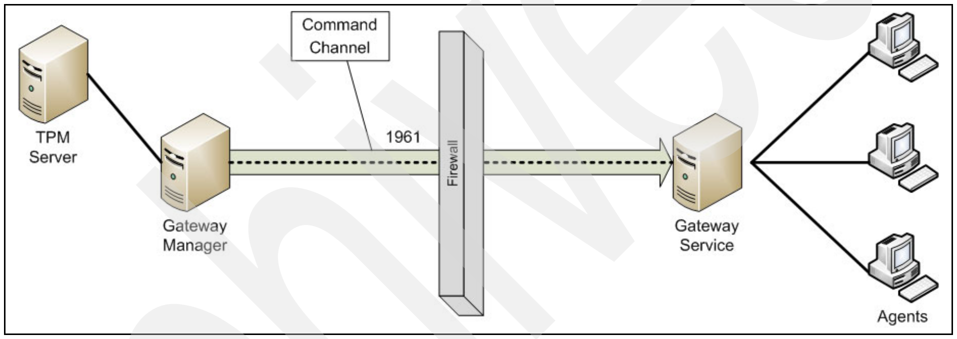


Figure 10-1 Single firewall between Tivoli Provisioning Manager for Software and common agents

In the environment depicted in Figure 10-2 on page 298, depots must be located in the data center, on the Tivoli Provisioning Manager for Software side of the firewall. Alternately, you can place depots in the DMZ when port 2100 is opened on the firewall that separates the data center and the DMZ. You cannot place depots on the agents' side because the dynamic content delivery service management center must talk directly to port 2100 of every depot.

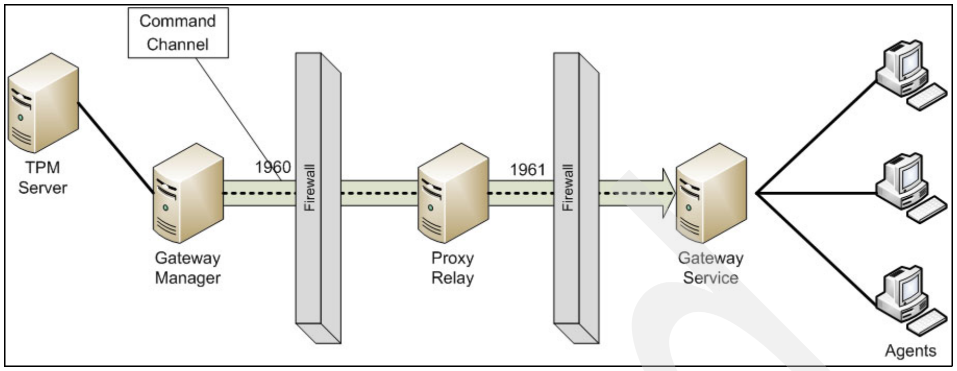


Figure 10-2 Two firewalls with DMZ

Figure 10-3 shows a more complex environment, where a single Gateway Manager handles connections to multiple Gateway Services.

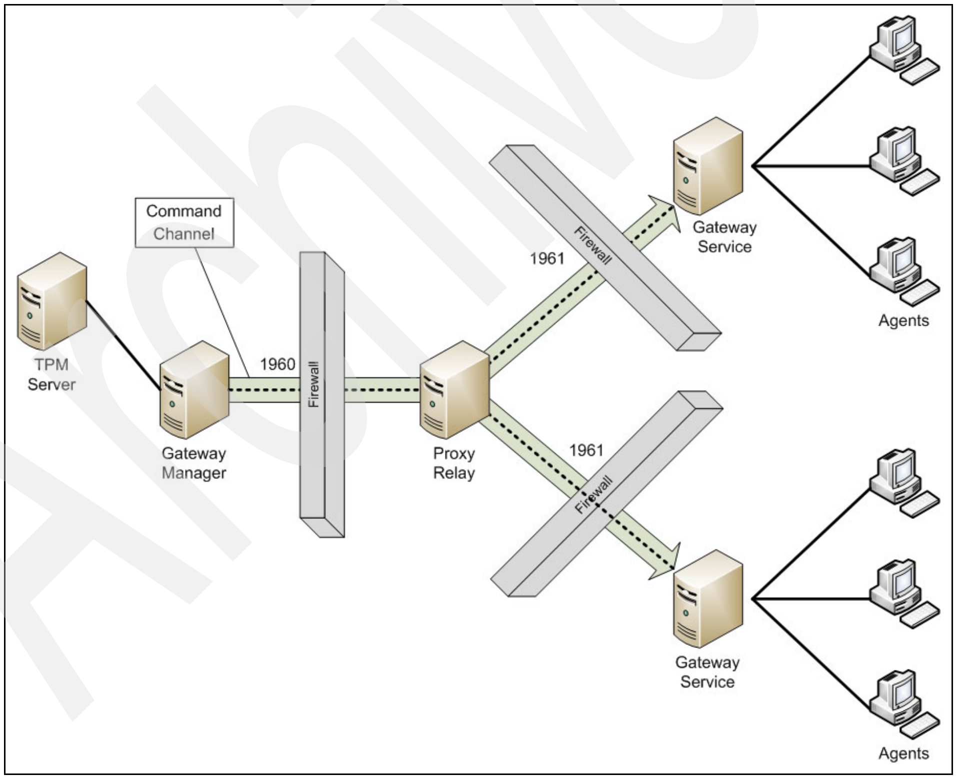


Figure 10-3 Multiple Gateway Service machines with DMZ

10.3 Installing and configuring firewall components

In this section, we describe how to install and configure the different components in the Scalable Distribution Infrastructure to allow communication in a firewalled environment.

All components, Gateway Manager, Gateway Service and Proxy Relay, are packaged in the same Java jar file, which is called proxy.jar and located in the $TIO_HOME/repository/FirewallProxy directory.

In the following scenario, we first use oslo.itsc.austin.ibm.com as a Gateway Manager and barcelona.itsc.austin.ibm.com as a Gateway Service in an environment similar to Figure 10-1 on page 297. Later we introduce ankara.itsc.austin.ibm.com as a Proxy Relay to build an environment as shown in Figure 10-2 on page 298.

In our lab, we do not have a firewall environment; instead, all systems can directly connect to each other, but this is not relevant to the different scenarios.

Table 10-5 Firewall components used

Host	IP address	OS	Component
oslo	9.3.5.49	SUSE	Gateway Manager
barcelona	9.3.5.89	RHEL	Gateway Service
ankara	9.3.5.55	RHEL	Proxy Relay

Table 10-6 shows the Tivoli Provisioning Manager for Software components we use in the following scenario.

Table 10-6 Tivoli Provisioning Manager for Software components

Host	IP address	Component
jiga.itsc.austin.ibm.com	9.3.4.163	Tivoli Provisioning Manager for Software server
nice.itsc.austin.ibm.com	9.3.5.88	SDI depot server
izmir.itsc.austin.ibm.com	9.3.5.74	TCA

Note: Make sure you use a recent JDK for running the firewall components. We had issues downloading files from the depot with old Java 1.4.2 versions. The following scenarios are run with **`Classic VM (build 1.4.2, J2RE 1.4.2 IBM build cxia32142-20070321 (SR8) (JIT enabled: jitc))`**, and at least SR5 is recommended.

10.3.1 Setting up the Gateway Manager

Usage for the Gateway Manager is `java -jar proxy.jar -gwmanager [configuration.xml]`. If the optional configuration is not provided on the command line, a default of `gateway.xml` is assumed. Make sure that the configuration XML-file is present in the current directory.

The gateway configuration file describes the Gateway Service(s) to which the Gateway Manager connects and the services that are tunneled.

Table 10-7 contains the elements for the XML file.

Table 10-7 Gateway XML elements

Element	Content
gatewaymanager	A single element that describes the Gateway Manager.
gatewayservice	One or more elements that are contained in the gatewaymanager, describing name, IP address, and the port number of a remote Gateway Service and optionally a routepath pointing to Proxy Relays.
service	One or more elements that are contained in gatewayservice, describing name, IP address, and the port number of a tunneled resource, and whether it is advertised and optionally a routepath pointing to Proxy Relays

Only the SDI managers are tunneled, and Table 10-8 lists the components.

Table 10-8 SDI components and ports tunneled

Component	Port
Dynamic content delivery service/device management service	9045
Dynamic content delivery service/device management service	9046
Dynamic content delivery service depot	2100
Agent Manager	9511

Component	Port
Agent Manager	9512
Agent Manager	9513

1. Copy the proxy.jar file on the Tivoli Provisioning Manager for Software server from $TIO_HOME/repository/FirewallProxy to the Gateway Manager system. In this case, we create a directory called /opt/tivoli/gw_manager on oslo, and copy the jar file there.
2. Create the configuration XML file. Route the SDI managers to our Tivoli Provisioning Manager for Software server running on jiga.itsc.austin.ibm.com and living on IP address 9.2.4.163. Route the depot port 2100 to our depot server on nice.itsc.austin.ibm.com. Create the Gateway Service link to barcelona, and listen on the non-default port number of 19061. Tell it to advertise all service names. Name the file gatewaymanager.xml.

Note: You need to add a service element for each depot you have behind the firewall in respect to the common agents. Tivoli Provisioning Manager for Software registers depots by host name in the dynamic content delivery service management center and uses these registered names in download plans. The UDP discovery requests the depot service by host name, so we advertise it by host name.

Example 10-1 gatewaymanager.xml

```
<!DOCTYPE gatewaymanager>

<gatewaymanager>
   <gatewayservice name="barcelona" type="remote"
      host="9.3.5.89" port="19061">

      <service name="nice.itsc.austin.ibm.com:2100" port="2100"
advertise="true">
         <resource host="9.3.5.88" port="2100" />
      </service>

      <service name="9.3.4.163:9511" port="9511" advertise="true">
         <resource host="9.3.4.163" port="9511" />
      </service>

      <service name="9.3.4.163:9512" port="9512" advertise="true">
         <resource host="9.3.4.163" port="9512" />
      </service>
      <service name="9.3.4.163:9513" port="9513" advertise="true">
         <resource host="9.3.4.163" port="9513" />
      </service>

      <service name="9.3.4.163:9045" port="9045" advertise="true">
         <resource host="9.3.4.163" port="9045" />
      </service>

      <service name="9.3.4.163:9046" port="9046" advertise="true">
         <resource host="9.3.4.163" port="9046" />
      </service>

   </gatewayservice>
</gatewaymanager>
```

3. Next we need to create an SSL certificate to encrypt traffic over the Command Channel between Gateway Manager and Service machines. Use the ikeyman utility of the IBM JDK 1.4.2 to create the client key file. Start the ikeyman utility from the $JAVA_HOME/bin directory.
   a. Select **Key Database File** → **New**, and create the gateway.private file in the /opt/tivoli/gw_manager file. Make sure you select **JKS** as the database type, and press **OK**.

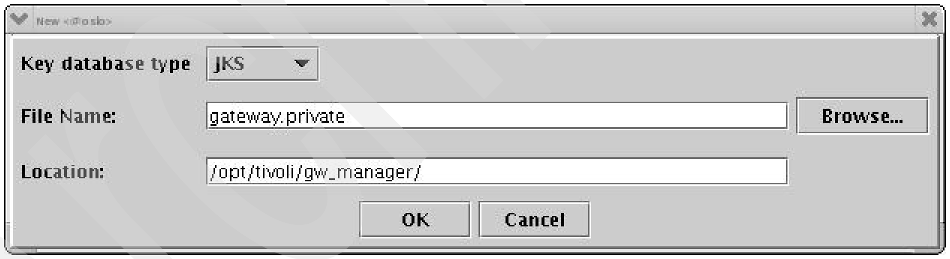
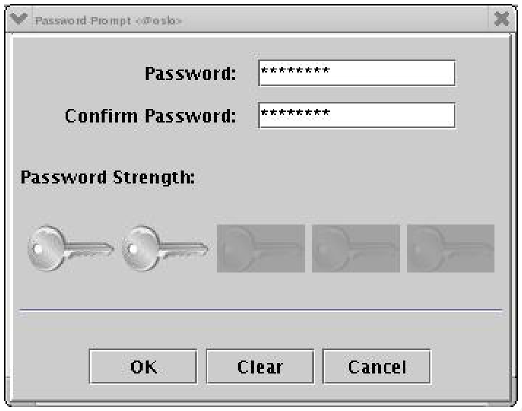

Key database type: JKS
File Name: gateway.private
Location: /opt/tivoli/gw_manager/

Figure 10-4   Create gateway.private key file

   b. Type **`password`** as the password, confirm it, and press **OK**.


Password Prompt <@oslo>

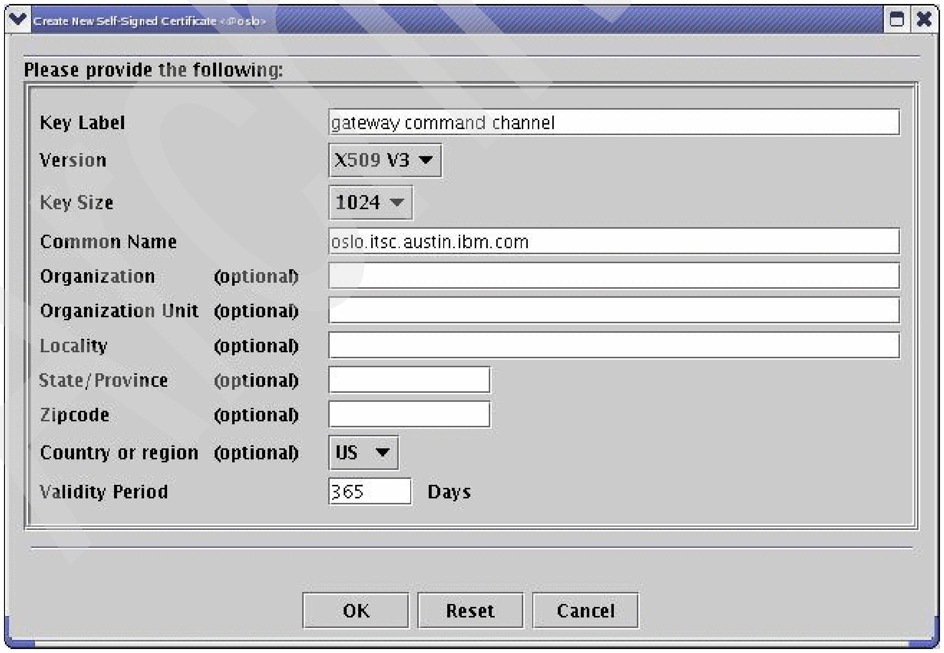
Password: ********

Confirm Password: ********

Password Strength:

Figure 10-5 Type the password

c. Select **Create** → **New Self-signed Certificate**, and provide all of the required fields, label and common name**.** Make sure that the default **version X509 V3** and **key size 1024** are not changed. Note that the certificate has a validity period. Accept the default of one year expiry date—it is necessary to renew the certificate before the year expires—and select **OK**.


Create New Self-Signed Certificate <@oslo>

Please provide the following:

Key Label		gateway command channel
Version		X509 V3
Key Size		1024
Common Name		oslo.itsc.austin.ibm.com
Organization	(optional)	
Organization Unit	(optional)	
Locality	(optional)	
State/Province	(optional)	
Zipcode	(optional)	
Country or region	(optional)	US
Validity Period		365 Days

Figure 10-6 Create certificate

d. Select **Key Database File** → **Exit** to exit the ikeyman utility. The gateway.private is created in the /opt/tivoli/gw_manager directory.

Example 10-2 Contents of Gateway Manager directory

```
oslo:/opt/tivoli/gw_manager # ls -ltr
total 120
-rw-r--r--  1 root root 88674 May 16 09:05 proxy.jar
-rw-r--r--  1 root root   910 May 24 01:08 gatewaymanager.xml
-rw-r--r--  1 root root 26530 May 24 01:25 gateway.private
drwxr-xr-x  7 root root   232 May 24 01:30 ..
drwxr-xr-x  2 root root   152 May 24 01:31 .
```

4. The Gateway Manager expects the **`gateway.private`** file to be present in the current directory, where it reads the key file at startup and creates a new file **`gateway.public,`** if it is not present. Copy this .public file to the current directory of the Gateway Service. You can rename the file, but you must retain the .public file extension.
5. Start the Gateway Manager by typing **`java -jar proxy.jar -gwmanager gatewaymanager.xml`** in the /opt/tivoli/gw_manager directory.

Example 10-3 Start up of Gateway Manager and generation of gateway.public

```
oslo:/opt/tivoli/gw_manager # java -jar proxy.jar -gwmanager
gatewaymanager.xml
May 24, 2007 1:38:00 AM com.ibm.jac.proxy.gateway.GatewayManager
initialize
INFO: Configured 1 gateway service
May 24, 2007 1:38:00 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Gateway Manager service starting for 9.3.5.89:19061
May 24, 2007 1:38:00 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Configuring gateway service for 6 service(s)
May 24, 2007 1:38:00 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Connecting to gateway service
May 24, 2007 1:38:00 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
WARNING: Connection closed: java.net.ConnectException: Connection
refused
oslo:/opt/tivoli/gw_manager # ls -ltr
total 148
-rw-r--r--  1 root root 88674 May 16 09:05 proxy.jar
-rw-r--r--  1 root root   910 May 24 01:08 gatewaymanager.xml
-rw-r--r--  1 root root 26530 May 24 01:25 gateway.private
```

```
drwxr-xr-x  7 root root   232 May 24 01:30 ..
drwxr-xr-x  2 root root   184 May 24 01:37 .
-rw-r--r--  1 root root 25822 May 24 01:37 gateway.public
oslo:/opt/tivoli/gw_manager # cp gateway.public oslo.public
```

6. Press Ctrl+C to stop the Gateway Manager after it starts up. The Gateway Manager will try to connect to its Gateway Service, but we have not yet created that service. Notice that the gateway.public file was created, and so copy the file to oslo.public.

Make sure that the Gateway Manager is started automatically after the system restarts. You can create an init script on Unix and Linux, or you can use the SRVANY utility to run the Gateway Server as a service on Windows systems.

10.3.2 Setting up the Gateway Service

Usage for the Gateway Manager is **`java -jar proxy.jar -gwservice [portnumber]`**. If the optional portnumber is not typed on the command line, then a default portnumber of **`1961`** is assumed.

Note: Changing the default portnumber for the Gateway Service is only available from fix pack 2 onwards.

There is no configuration XML file for a Gateway Service machine. The Gateway Manager contacts the Gateway Service and communicates the services it needs to advertise and tunnel.

We do need a copy of the gateway.public file from the Gateway Manager system to securely communicate to the Gateway Manager and of course the proxy.jar itself. The Gateway Service picks up every file with the .public extension in the current directory, so it can securely connect to multiple Gateway Managers.

We use barcelona, acting as our Gateway Service, running from the /opt/tivoli/gw_service directory, and connecting to oslo.

Note: A Gateway Service does not initiate outbound connection; instead, it waits for a Gateway Manager to connect to it.

We start the Gateway Service using the non-default port 19061 by typing **`java -jar proxy.jar -gwservice 19061`** in the /opt/tivoli/gw_service directory. We are not using the default port as configured in the gatewaymanager.xml, shown in Example 10-1 on page 301.

Example 10-4 Start up of Gateway Service

```
[root@barcelona gw_service]# ls -ltr
total 120
-rwxr-xr-x  1 root root 88674 May 16 13:46 proxy.jar
-rw-r--r--  1 root root 25822 May 24 10:09 oslo.public
[root@barcelona gw_service]# java -jar proxy.jar -gwservice 19061
May 24, 2007 10:13:08 AM com.ibm.jac.proxy.gateway.SSLUtils
getClientSSLContext
INFO:   Added public key for verisign class 1 public primary
certification authority - g3
..<snip>..
May 24, 2007 10:13:09 AM com.ibm.jac.proxy.gateway.SSLUtils
getClientSSLContext
INFO:   Added public key for gateway command channel
..<snip>..
May 24, 2007 10:13:09 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Gateway service starting
May 24, 2007 10:13:09 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Advertising service starting
May 24, 2007 10:13:09 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Listening command socket on port 19061
```

Notice the ‘gateway command channel’ public key, which was added from the oslo.public file, and you will see the Gateway Service waiting for the Gateway Manager to connect.

Start the Gateway Manager again, and see it connecting to the Gateway Service in Example 10-5.

Example 10-5 Gateway Manager handshake

```
oslo:/opt/tivoli/gw_manager # java -jar proxy.jar -gwmanager
gatewaymanager.xml
May 24, 2007 1:48:50 AM com.ibm.jac.proxy.gateway.GatewayManager
initialize
INFO: Configured 1 gateway service
May 24, 2007 1:48:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Gateway Manager service starting for 9.3.5.89:19061
May 24, 2007 1:48:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Configuring gateway service for 6 service(s)
May 24, 2007 1:48:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
```

```
INFO: Connecting to gateway service
May 24, 2007 1:48:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Sending request for command channel
Start handshake
Finish handshake
May 24, 2007 1:48:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Sending services
May 24, 2007 1:48:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Waiting for reply
May 24, 2007 1:48:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: OK status returned
```

The Gateway Manager and Service are initialized and ready for operation if the line `INFO: OK status returned` appears on the Gateway Manager's console.

In Example 10-6, the Gateway Service accepts the communication and advertising and forwards the configured services.

Example 10-6 Gateway Service handshake

```
..
INFO: Accepted connection from 9.3.5.49:56220
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Command channel request
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Gateway service thread starting
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Command channel request
Start handshake
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.SSLUtils$2
checkServerTrusted
INFO: Trusted CN=oslo.itsc.austin.ibm.com, C=US
Finish handshake
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 9046
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9046 for port 9046
```

```
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 2100
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service nice.itsc.austin.ibm.com:2100 for port 2100
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 9511
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9511 for port 9511
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 9045
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9045 for port 9045
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 9513
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9513 for port 9513
May 24, 2007 10:16:26 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56220>Forwarding port 9512
May 24, 2007 10:16:26 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9512 for port 9512
```

In Example 10-7, see how we can verify the UDP discovery with the **-locate** option of the proxy.jar, from anywhere within the subnet of the Gateway Service.

Example 10-7 Locate a service

```
[root@barcelona gw_service]# java -jar proxy.jar -locate 9.3.4.163:9046
GatewayServiceHost = barcelona.itsc.austin.ibm.com
ServicePort =9046
[root@barcelona gw_service]# java -jar proxy.jar -locate
nice.itsc.austin.ibm.com:2100
GatewayServiceHost = barcelona.itsc.austin.ibm.com
ServicePort =2100
[root@barcelona gw_service]# java -jar proxy.jar -locate
jiga.itsc.austin.ibm.com:9046
Timeout waiting for response
```

Specify exactly what is configured in the gatewaymanager.xml. If it is not exactly matched, the Gateway Service fails to respond and the locate operation will time-out.

Example 10-8 Gateway Service console log for UDP discovery

```
May 24, 2007 10:23:42 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service 9.3.4.163:9046 from /9.3.5.89:33421
May 24, 2007 10:23:42 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 9046
May 24, 2007 10:23:47 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service nice.itsc.austin.ibm.com:2100 from
/9.3.5.89:33422
May 24, 2007 10:23:47 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 2100
May 24, 2007 10:24:04 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service jiga.itsc.austin.ibm.com:9046 from
/9.3.5.89:33423
```

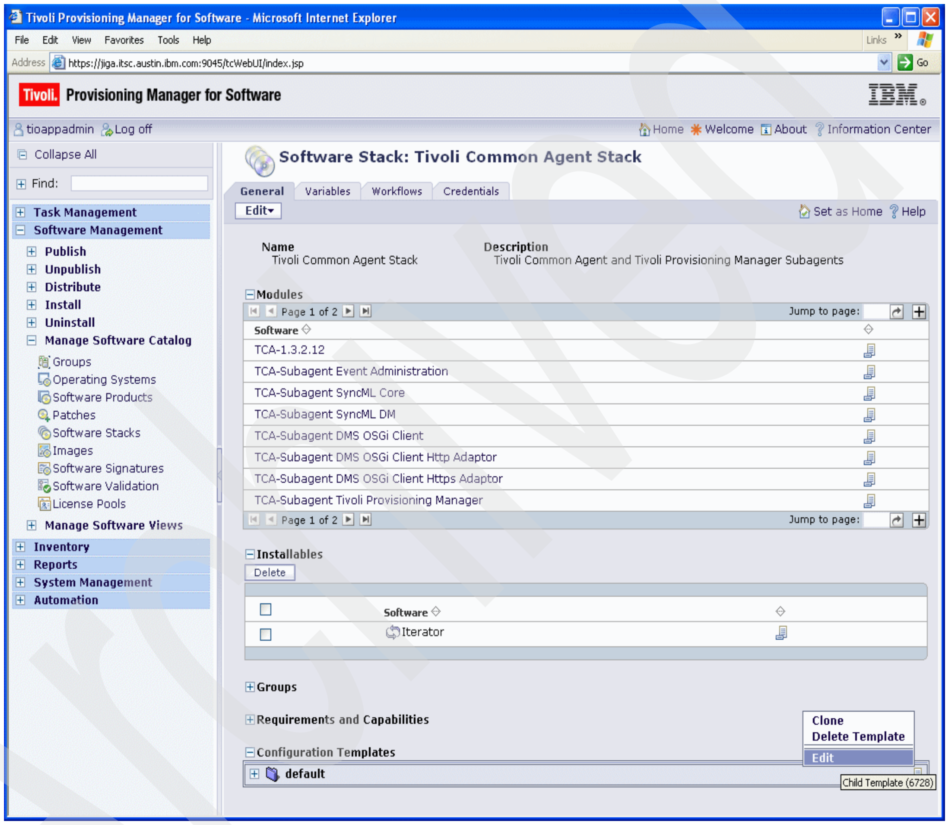
10.3.3 Configuring the common agent for firewall support

To allow a common agent to discover Tivoli Provisioning Manager for Software services on the other side of the firewall, we must enable firewall support on the common agent.

Configuring new agents

When deploying new agents, we enable firewall support by modifying Tivoli Common Agent Stack Configuration Templates.

1. Select **Software Management** → **Manage Software Catalog** → **Software Stacks**, and click Tivoli Common Agent Stack. Expand the **Configuration Templates**, and select **Edit** from the more action button, which is located to the right of the default Configuration Template.



Tivoli Provisioning Manager for Software - Microsoft Internet Explorer

Address https://jiga.itsc.austin.ibm.com:9045/tcWebUI/index.jsp

Provisioning Manager for Software

tioappadmin Log off | Home Welcome About Information Center

Collapse All
Find:
- Task Management
- Software Management
  - Publish
  - Unpublish
  - Distribute
  - Install
  - Uninstall
  - Manage Software Catalog
    - Groups
    - Operating Systems
    - Software Products
    - Patches
    - Software Stacks
    - Images
    - Software Signatures
    - Software Validation
    - License Pools
  - Manage Software Views
- Inventory
- Reports
- System Management
- Automation

Software Stack: Tivoli Common Agent Stack

General | Variables | Workflows | Credentials

Edit▾ | Set as Home | Help

Name: Tivoli Common Agent Stack
Description: Tivoli Common Agent and Tivoli Provisioning Manager Subagents

Modules (Page 1 of 2, Jump to page:)

Software
TCA-1.3.2.12
TCA-Subagent Event Administration
TCA-Subagent SyncML Core
TCA-Subagent SyncML DM
TCA-Subagent DMS OSGi Client
TCA-Subagent DMS OSGi Client Http Adaptor
TCA-Subagent DMS OSGi Client Https Adaptor
TCA-Subagent Tivoli Provisioning Manager

Installables
Delete

	Software
☐	Iterator

Groups

Requirements and Capabilities

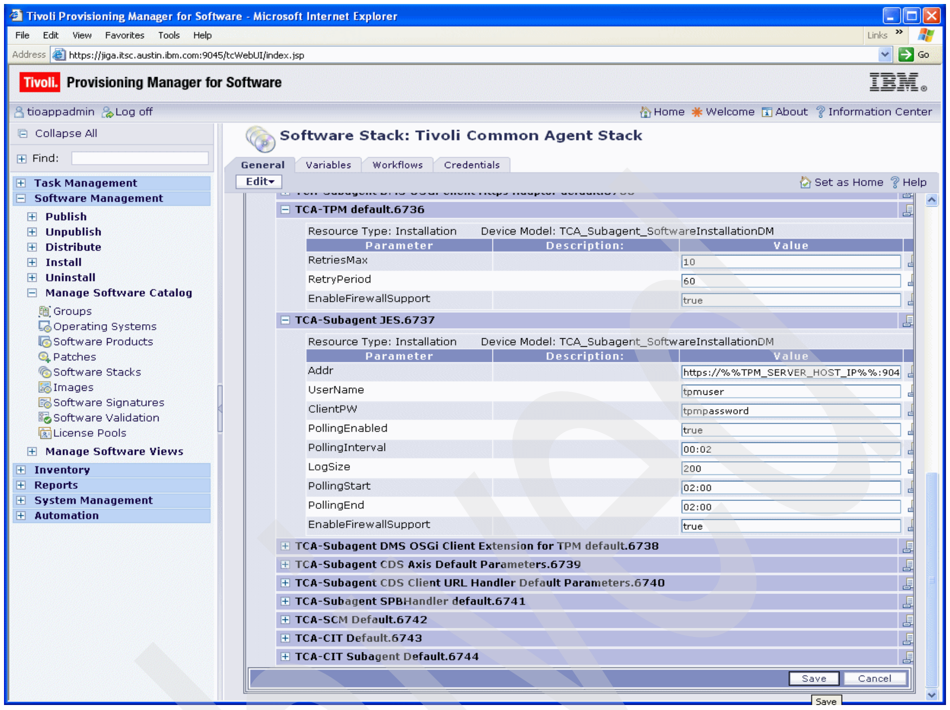
Configuration Templates
default

Clone
Delete Template
Edit

Child Template (6728)

Figure 10-7 Edit Tivoli Common Agent Configuration Template

2. Change the **TCA-Subagent JES** and **TCA-TPM** configurations, by modifying **EnableFirewallSupport** to **true**, and press the Save button, which is located in the bottom-right corner.


Software Stack: Tivoli Common Agent Stack

TCA-TPM default.6736

Resource Type: Installation   Device Model: TCA_Subagent_SoftwareInstallationDM

Parameter	Description:	Value
RetriesMax		10
RetryPeriod		60
EnableFirewallSupport		true

TCA-Subagent JES.6737

Resource Type: Installation   Device Model: TCA_Subagent_SoftwareInstallationDM

Parameter	Description:	Value
Addr		https://%%TPM_SERVER_HOST_IP%%:904
UserName		tpmuser
ClientPW		tpmpassword
PollingEnabled		true
PollingInterval		00:02
LogSize		200
PollingStart		02:00
PollingEnd		02:00
EnableFirewallSupport		true

TCA-Subagent DMS OSGi Client Extension for TPM default.6738
TCA-Subagent CDS Axis Default Parameters.6739
TCA-Subagent CDS Client URL Handler Default Parameters.6740
TCA-Subagent SPBHandler default.6741
TCA-SCM Default.6742
TCA-CIT Default.6743
TCA-CIT Subagent Default.6744

Figure 10-8 Enable firewall support on Tivoli Common Agent stack

All new agents deployed with this Tivoli Common Agents stack finds the Tivoli Provisioning Manager for Software services by UDP discovery and connects to them over a Gateway Service.

Note: Do not forget to also modify Job Execution Service (JES) and Tivoli Provisioning Manager for Software common agent sub-agents of the CDS_Depot_Stack, if you intend to deploy depots behind the firewall.

Configuring existing agents

Alternatively, we can enable firewall support manually on existing common agents, by modifying property files for JES and Tivoli Provisioning Manager subagents. We edit the following files and set EnableFirewallSupport to true:

- $INSTALL_DIR/runtime/agent/subagent/jes.properties
- $INSTALL_DIR/runtime/agent/subagent/jes.properties

Example 10-9 JES and Tivoli Provisioning Manager property firewall support values

```
C:\Program Files\tivoli\ep\runtime\agent\subagents>type jes.properties
#Fri Apr 27 15:58:14 CDT 2007
Addr=https\://9.3.4.163\:9045/dmserver/SyncMLDMServlet
PollingEnd=02\:00
PollingInterval=00\:02
PollingStart=02\:00
UserName=tpmuser
LogSize=200
PollingEnabled=true
EnableFirewallSupport=true
ClientPW=tpmpassword

C:\Program Files\tivoli\ep\runtime\agent\subagents>type tpm.properties
#Fri Apr 27 15:58:04 CDT 2007
RetryPeriod=60
RetriesMax=10
EnableFirewallSupport=true
```

Restart the agent after you change the values, by typing **`%INSTALLDIR%\runtime\agent\endpoint.bat stop`** followed by **`start`**.

Example 10-10 Stopping and starting the common agent

```
C:\Program Files\tivoli\ep\runtime\agent>endpoint stop
Stopping service IBMTivoliCommonAgent0
Waiting for service IBMTivoliCommonAgent0 to stop (will wait a maximum
of 5 minu
tes)
Stopped service IBMTivoliCommonAgent0

C:\Program Files\tivoli\ep\runtime\agent>endpoint start
Starting service IBMTivoliCommonAgent0
Started service IBMTivoliCommonAgent0
```

Configuring agents without UDP discovery

When UDP traffic is blocked between the common agent and the Gateway Service system, the discovery mechanism will not function; furthermore, the common agent cannot automatically find routing information to access the Tivoli Provisioning Manager for Software components behind network discontinuities.

It is possible to hard code the network addresses and ports in the appropriate property files, and point the common agent directly to the Gateway Service to yet allow traversal over firewalls.

You can force the dynamic content delivery service client to pass through the firewall components, by creating the following file:
`$INSTALL_DIR/runtime/agent/subagent/cds/client/proxy.conf`

Enter proxy address port pairs in the following syntax:
targethost:targetport=proxyhost:proxyport

Proxy the Tivoli Provisioning Manager for Software server's port 9046 and all depots' port 2100 to the nearest Gateway Service system on the correct advertised ports.

You can force the JES/DMS component to communicate directly to the Gateway Service.

- Before you install the common agent, modify the software resource template (SRT) for the JES subagent in the Tivoli Common Agent stack. Replace %%TPM_SERVER_HOST_IP%% in the value of the *Addr* parameter with the IP address of the Gateway Service system.
- On an already installed common agent, stop the endpoint and edit the $INSTALL_DIR/runtime/agent/subagent/jes.properties file. Replace the Tivoli Provisioning Manager for Software server's IP address with the IP address of the Gateway Service system. Additionally remove the file OSGiAgentTree.bin and OSGiAgentTree.bin.bak from the $INSTALL_DIR/runtime/base/workspace directory. Then start the agent.

A deployed common agent stores the address of the Tivoli Provisioning Manager for Software server in the file $INSTALL_DIR/runtime/agent/config/endpoint.properties. Change all occurrences of the Tivoli Provisioning Manager for Software server's IP address to the IP address of the Gateway Service. Stop and start the agent after you modify the property file.

Note: In the previously mentioned scenarios, we connected the common agent to Tivoli Provisioning Manager for Software components, which all reside on the Tivoli Provisioning Manager for Software server. In large complex environments, you can install the different components on separate physical machines. In that case, you must point the Gateway Manager configuration XML file, or local modifications, to the system that is actually providing the service.

10.3.4 Communicating over the firewall components

When you start a common agent with firewall support, you can see the connection requests—whether broadcasted or direct—coming into the Gateway Service, which forwards the traffic to the Gateway Manager. The communication is tunneled over the ProxyPumps.

In Example 10-11 and Example 10-12 on page 317, the common agent izmir, with the IP address 9.3.5.74, starts up and sends out UDP discovery diagrams, which report to the Agent Manager service on port 9512 (and possibly 9511 and 9513), and requests jobs from the device management service server on port 9045. A software distribution job is received, the common agent requests a download plan from the dynamic content delivery service management center on port 9046, and then downloads the SPB from the depot on nice.itsc.austin.ibm.com at port 2100.

Note: A common agent will first try directly to its depot. If this succeeds, the common agent will not communicate over the firewall components. If the dynamic content delivery service management center or a depot is not reachable, then the dynamic content delivery service client falls back to UDP discovery and connects through the firewall components.

In our lab environment, without physical firewalls, we installed a software firewall on the common agent to block all TCP traffic to and from the Tivoli Provisioning Manager for Software server and the depot.

Example 10-11 Gateway Service forwarding to Gateway Manager

```
May 24, 2007 11:53:23 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service 9.3.4.163:9045 from /9.3.5.74:1157
May 24, 2007 11:53:23 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 9045
May 24, 2007 11:53:23 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service barcelona.itsc.austin.ibm.com:9045 from
/9.3.5.74:1158
May 24, 2007 11:53:34 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service 9.3.4.163:9512 from /9.3.5.74:1159
May 24, 2007 11:53:34 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 9512
```

```
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56730
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.5.74,port=1161,localport=9512] -->
Socket[addr=/9.3.5.49,port=56730,localport=19061]
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.89:9512
remote=/9.3.5.74:1161]->java.nio.channels.SocketChannel[connected
local=/9.3.5.89:19061 remote=/9.3.5.49:56730]
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56732
May 24, 2007 11:53:34 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:53 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:53 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56736
May 24, 2007 11:53:53 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:54 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.5.49,port=56736,localport=19061] -->
Socket[addr=/9.3.5.74,port=1164,localport=9045]
May 24, 2007 11:53:54 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.89:19061
remote=/9.3.5.49:56736]->java.nio.channels.SocketChannel[connected
local=/9.3.5.89:9045 remote=/9.3.5.74:1164]
May 24, 2007 11:53:54 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service 9.3.4.163:9046 from /9.3.5.74:1166
May 24, 2007 11:53:54 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 9046
May 24, 2007 11:53:54 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
```

```
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:54 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56738
May 24, 2007 11:53:54 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:55 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Request for service nice.itsc.austin.ibm.com:2100 from
/9.3.5.74:1170
May 24, 2007 11:53:55 AM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser run
INFO: Responding with 2100
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56740
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:56 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.5.49,port=56740,localport=19061] -->
Socket[addr=/9.3.5.74,port=1171,localport=2100]
May 24, 2007 11:53:56 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.89:19061
remote=/9.3.5.49:56740]->java.nio.channels.SocketChannel[connected
local=/9.3.5.89:2100 remote=/9.3.5.74:1171]
May 24, 2007 11:53:58 AM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.49:56698>accepted connection from /9.3.5.74
May 24, 2007 11:53:58 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.49:56742
May 24, 2007 11:53:58 AM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Data channel request
May 24, 2007 11:53:58 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.5.74,port=1172,localport=9045] -->
Socket[addr=/9.3.5.49,port=56742,localport=19061]
May 24, 2007 11:53:58 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.89:9045
remote=/9.3.5.74:1172]->java.nio.channels.SocketChannel[connected
local=/9.3.5.89:19061 remote=/9.3.5.49:56742]
```

The common agent caches discovery responses. Dynamic content delivery service connection information is cached in $INSTALL_DIR/runtime/agent/subagents/cds/client/proxy.cache. If this information is still valid, no UDP discovery is performed.

Example 10-12   Gateway Manager handling Gateway Service request

```
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.4.163:9512
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.4.163,port=9512,localport=56729] -->
Socket[addr=/9.3.5.89,port=19061,localport=56730]
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56729
remote=/9.3.4.163:9512]->java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56730 remote=/9.3.5.89:19061]
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.4.163:9512
May 24, 2007 11:53:30 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.4.163:9045
May 24, 2007 11:53:50 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:50 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.4.163,port=9045,localport=56735] -->
Socket[addr=/9.3.5.89,port=19061,localport=56736]
May 24, 2007 11:53:50 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56735
remote=/9.3.4.163:9045]->java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56736 remote=/9.3.5.89:19061]
May 24, 2007 11:53:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.4.163:9046
```

```
May 24, 2007 11:53:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.5.88:2100
May 24, 2007 11:53:51 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:52 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.5.88,port=2100,localport=56739] -->
Socket[addr=/9.3.5.89,port=19061,localport=56740]
May 24, 2007 11:53:52 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56739
remote=/9.3.5.88:2100]->java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56740 remote=/9.3.5.89:19061]
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening channel to resource 9.3.4.163:9045
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Opening data channel to gateway service
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.4.163,port=9045,localport=56741] -->
Socket[addr=/9.3.5.89,port=19061,localport=56742]
May 24, 2007 11:53:55 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56741
remote=/9.3.4.163:9045]->java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56742 remote=/9.3.5.89:19061]
May 24, 2007 11:54:01 AM com.ibm.jac.proxy.ProxyPump readChannel
INFO: Closing Socket[addr=/9.3.4.163,port=9046,localport=56737] -->
Socket[addr=/9.3.5.89,port=19061,localport=56738]
May 24, 2007 11:54:01 AM com.ibm.jac.proxy.ProxyPump closeChannelPair
INFO: Closing channel pair java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56737
remote=/9.3.4.163:9046]->java.nio.channels.SocketChannel[connected
local=/9.3.5.49:56738 remote=/9.3.5.89:19061]
```

10.3.5 Introducing the Proxy Relay

When several network discontinuities exist between the Tivoli Provisioning Manager for Software services and the common agents, for example, as shown in Figure 10-2 on page 298 or Figure 10-3 on page 298, one or more *Proxy Relay systems* must be put in place to traverse the multiple firewalls.

Usage for the Proxy Relay is **`java -jar proxy.jar -relay`**. Proxy Relays provide IP based access control lists (ACLs) for incoming and outgoing connections. The configuration of the Proxy Relay is done with the *proxyrelay.xml* file, which is read from the current directory.

Table 10-9 describes the contents of the proxyrelay.xml elements.

Table 10-9 Proxyrelay.xml elements

Element	Content
proxyrelay	A single element with one parameter and one port, which defaults to 1960 if not present.
inboundACL	One element that is contained in proxyrelay and describes one or more entries. Each entry describes a Gateway Manager or another Proxy Relay.
outboundACL	One element that is contained in proxyrelay and describes one or more entries. Each entry describes a Gateway Service or another Proxy Relay.

The format of the ACL consists of a base IP address, followed by an optional subnet mask, and a port range. If you want a range of addresses, consider using the optional subnet mask, which starts with a forward slash and is any number between 1 and 32. The number indicates how many of the IP address' 32 bits are significant. If the subnet mask is not specified, the default behavior is to match all 32 bits. Additionally, you can specify a port number or port range in the ACL. The port entry is delimited by a colon. You can specify multiple ports as a range that is separated by a hyphen (for example, 1960-1970), or you can separate the range as separate entries delimited by a comma (for example, 1950, 1960). If no ports are specified, by default, the ACL allows ports 1950 and 1960. See Table 10-10 on page 320, for examples of ACL formats.

Table 10-10 ACL format

IP address	Subnet mask	Port range
192.168.100.1	/24	:1950, 1960
9.0.0.0	/8	:1960
10.10.10.10	/32	:1960-1965

To route the Gateway Manager to a Gateway Server over a Proxy Relay, you need to adapt the Gateway Manager XML configuration. Add a routepath element to the specific Gateway Service. In our setup, we copy our existing *gatewaymanager.xml* (see Example 10-1 on page 301) to *gatewayproxy.xml*, and we route the connection to our Gateway Service on barcelona through the Proxy Relay on ankara with IP address 9.3.5.55. We route through the non-default port 19061.

Example 10-13 Gateway Manager XML configuration with Proxy Relay routing

```
<!DOCTYPE gatewaymanager>

<gatewaymanager>
      <gatewayservice name="barcelona" type="remote"
         host="9.3.5.89" port="19061">
         <routepath>
            <pathElement host="9.3.5.55" port="19060"/>
         </routepath>

         <service name="nice.itsc.austin.ibm.com:2100" port="2100"
advertise="true">
            <resource host="9.3.5.88" port="2100" />
         </service>

         <service name="9.3.4.163:9511" port="9511" advertise="true">
            <resource host="9.3.4.163" port="9511" />
         </service>

         <service name="9.3.4.163:9512" port="9512" advertise="true">
            <resource host="9.3.4.163" port="9512" />
         </service>

         <service name="9.3.4.163:9513" port="9513" advertise="true">
            <resource host="9.3.4.163" port="9513" />
         </service>

         <service name="9.3.4.163:9045" port="9045" advertise="true">
```

```
            <resource host="9.3.4.163" port="9045" />
         </service>

         <service name="9.3.4.163:9046" port="9046" advertise="true">
            <resource host="9.3.4.163" port="9046" />
         </service>

   </gatewayservice>
</gatewaymanager>
```

In Example 10-14, we start our Gateway Manager with this new configuration.

Example 10-14   Start Gateway Manager

```
oslo:/opt/tivoli/gw_manager # java -jar proxy.jar -gwmanager
gatewayproxy.xml
May 24, 2007 2:38:05 PM com.ibm.jac.proxy.gateway.GatewayManager
initialize
INFO: Configured 1 gateway service
May 24, 2007 2:38:05 PM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Gateway Manager service starting for 9.3.5.89:19061
May 24, 2007 2:38:06 PM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Configuring gateway service for 6 service(s)
May 24, 2007 2:38:06 PM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
INFO: Connecting to gateway service
May 24, 2007 2:38:06 PM com.ibm.jac.proxy.gateway.GatewayManagerThread
run
WARNING: Connection closed: java.net.ConnectException: Connection
refused
```

We have not yet set up our Proxy Relay on ankara, so the connection failed.

On ankara, create the directory /opt/tivoli/proxy, and copy the proxy.jar and proxyrelay.xml file to the directory, and start the Proxy Relay.

In the `proxyrelay.xml` configuration file, we define the address and port of the Gateway Manager and Gateway Service as inbound and outbound ACL respectively. We use the same non-default ports as above.

Example 10-15 Proxyrelay.xml configuration file

```
<!DOCTYPE proxyrelay>

<proxyrelay port="19060">
        <inboundACL>
                <entry>9.3.5.49:19060</entry>
        </inboundACL>
        <outboundACL>
                <entry>9.3.5.89:19061</entry>
        </outboundACL>
</proxyrelay>
```

We start the Proxy Relay with the command **`java -jar proxy.jar -relay`**.

Example 10-16 Proxy Relay starting

```
[root@ankara proxy]# java -jar proxy.jar -relay
May 24, 2007 3:03:23 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Proxy is starting.
May 24, 2007 3:03:23 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Proxy listening on port 19060
May 24, 2007 3:03:23 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Proxy socket timeout:  30000ms
May 24, 2007 3:03:23 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Inbound ACL :  [9.3.5.49/32 19060]
May 24, 2007 3:03:23 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Outbound ACL:  [9.3.5.89/32 19061]
May 24, 2007 3:04:08 PM com.ibm.jac.proxy.ProxyRelay run
INFO: Connection from Socket[addr=/9.3.5.49,port=57592,localport=19060]
May 24, 2007 3:04:09 PM com.ibm.jac.proxy.ProxyRelay validateConnection
INFO: Inbound connection from 9.3.5.49 to port 19060 allowed
(9.3.5.49/32:19060)
May 24, 2007 3:04:09 PM com.ibm.jac.proxy.ProxyRelay validateConnection
INFO: Outbound connection to 9.3.5.89:19061 allowed (9.3.5.89/32:19061)
```

We see the Proxy Relay accepting inbound connections from the Gateway Manager and outbound connections to the Gateway Service. When the Gateway Manager retries to contact the Gateway Service, the actual tunnel is being set up and data transfer now flows as shown in 10.3.4, “Communicating over the firewall components” on page 314, from Gateway Service to Gateway Manager.

In Example 10-17, we see the Gateway Service talking to the Proxy Relay.

Example 10-17 Relayed Gateway Service communication

```
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Accepted connection from 9.3.5.55:35072
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayService run
INFO: Command channel request
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Gateway service thread starting
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Command channel request
Start handshake
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.SSLUtils$2
checkServerTrusted
INFO: Trusted CN=oslo.itsc.austin.ibm.com, C=US
Finish handshake
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Forwarding port 9513
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9513 for port 9513
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Forwarding port 9511
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9511 for port 9511
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Forwarding port 9046
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9046 for port 9046
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Forwarding port 9045
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9045 for port 9045
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
```

```
INFO: <9.3.5.55:35072>Forwarding port 2100
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service nice.itsc.austin.ibm.com:2100 for port 2100
May 24, 2007 2:39:09 PM com.ibm.jac.proxy.gateway.GatewayServiceThread
log
INFO: <9.3.5.55:35072>Forwarding port 9512
May 24, 2007 2:39:09 PM
com.ibm.jac.proxy.gateway.GatewayServiceAdvertiser addService
INFO: Advertising service 9.3.4.163:9512 for port 9512
```

We can have multiple Proxy Relays to hop multiple network discontinuities.

11

Troubleshooting and maintenance

In this chapter, we provide tips for troubleshooting your Tivoli Provisioning Manager for Software and Tivoli Configuration Manager coexistence or migration environment. It contains the following sections:

11.1 Tivoli Provisioning Manager for Software troubleshooting overview

We describe the Tivoli Provisioning Manager/SDI infrastructure in detail in 1.1, "Introduction to the covered material" on page 2. This section covers:

- How to set trace levels
- How to set debug levels
- How to find information about the environment for all of the various components that make up Tivoli Provisioning Manager for Software.
- For all of the components, we address the log files and contents as well as increasing debug levels.

11.1.1 Log file summary

Table 11-1, shows where to find Tivoli Provisioning Manager for Software log files. For further details on components log files and its contents, you can consult each section in this document or review *Tivoli Provisioning Manager for Software Problem Determination and Troubleshooting Guide, Version 5.1*, SC32-2256, "Chapter 5, Log File Directories".

Table 11-1 Tivoli Provisioning Manager for Software log files summary

Component	Log root path
Tivoli Provisioning Manager server	*$TIO_LOGS* *$TIO_HOMES*/tioprofile/logs/server1
Deployment Engine (DE)	*$TIO_LOGS*/deploymentengine
Activity Plan Engine	*$TIO_LOGS*/activityplan
Policy Engine	*$TIO_LOGS*/policyengine
Agent Shell server	*$TIO_LOGS*/agentshellserver
Device management service server	*$TIO_HOME*/tioprofile/logs/server1
Remote Execution Access (RXA)	*$TIO_LOGS*/deploymentengine grep RXA contents in console.log
Dynamic content delivery services (CDS)	/opt/ibm/tivoli/common/ctgde/logs
User interfaces	*$TIO_LOGS*/j2ee
WebSphere	*$TIO_HOME*/tioprofile/logs/server1

Component	Log root path
Agent manager	Installation logs: *$AM_HOME*/logs Runtime logs: $TIO_HOME/tioprofile/logs/server1 On lightweight infrastructure (Fast Start) the installation logs are found in %LWI_HOME%\logs\rcp.log.0 and runtime logs in %LWI_HOME%\runtime\agentmanager\logs. Log collector script: Run $AM_HOME/toolkit/bin/logcollector.bat and a zip file is created in $AM_HOME Additional logs: $TIO_HOME/tioprofile/logs/ffdc
tcdrivermanager (workflows and automation packages)	$TIO_LOGS/tcdrivermanager
Conventions (default directories): Tivoli Provisioning Manager for Software log files ▶ UNIX systems: *$TIO_LOGS*, /usr/ibm/tivoli/common/COP/logs ▶ Windows systems: *%TIO_LOGS%*, C:\Program Files\ibm\tivoli\common\COP\logs Tivoli Provisioning Manager for Software installation directory ▶ UNIX systems: *$TIO_HOME*, /opt/IBM/tivoli/tmpfsw ▶ Windows systems: *%TIO_HOME%*, C:\Program Files\ibm\tivoli\tpmfsw Agent Manager installation directory ▶ UNIX systems: $AM_HOME, /opt/IBM/AgentManager ▶ Windows systems: %AM_HOME%, C:\Program Files\IBM\AgentManager WebSphere Application Server home directory ▶ UNIX Systems: *$WAS_HOME*, /usr/IBM/WebSphere/AppServer ▶ Windows Systems: *%WAS_HOME%*, C:\Program Files\IBM\WebSphere\AppServer	

Table 11-2 summarizes the health check summary information.

Table 11-2 Tivoli Provisioning Manager components health check summary

Component	**How to check health**
Tivoli Provisioning Manager server	$TIO_HOME/tools/tioStatus.sh/cmd <wasadmin_user> <wasadmin_passwd>

Tivoli Common Agent (TCA)	The common agent has a heartbeat function that sends periodic status and configuration reports to the Agent Manager, so that you can run a report to check the last status or run the TCA_PingAgent workflow. Another option is to locally run a script to collect diagnostic information, which creates the CASservice.zip file.
	on Windows:
	`C:\Progra~1\tivoli\ep\runtime\agent\service`
	on UNIX:
	`/usr/tivoli/ep/runtime/agent/service.sh`

Agent manager	Run TCA_PingAgentManager workflow. Another option is the following: Go to the $AM_HOME/toolkit/bin directory, and run the script Health_Check.sh/cmd. For further details about its execution, consult the $AM_HOME/toolkit/HealthCheck.readme file.
Dynamic content delivery service	Using the GUI, go to **Inventory** → **Infrastructure Management** → **Depots**. Click the Depot server link you want to check. The Depot's page appears, where you can click the **Check Status** button.
Device management service (DMS)	All operations to the device manager subagents through the JBS_COMMAND flow only through Job Execution Service (JES) and common agent services interfaces. Real-time control or queued control are possible through common agent services interfaces.

Component	How to check health
Deployment Engine	The Deployment Engine manages workflows. You can verify the deployment request status checking workflow status. You can also use the SOAP client interface: `$TIO_HOME/soapclient/tpmlteSoap/soapcli.sh/cmd` *`<username> <password> <wsdl_location>`* `findDeploymentStatus` *`<DeploymentRequestID>`*. The workflow status are the following: ► 0: unknown ► 1: in-progress ► 2: cancelled ► 3: success ► 4: failed ► 5: created

11.1.2 Environment summary

When interacting with Tivoli Provisioning Manager for Software, it is best to login as the tioadmin user or source the environment using the following command:

```
. /opt/ibm/tivoli/tpmfsw/.TCprofile
```

In that command, .TCprofile is located in the default installation location for Tivoli Provisioning Manager for Software. Example 11-1 shows the environment variables that you can set with .TCprofile. We use these variables throughout this section when we refer to directory locations.

Example 11-1 Tivoli Provisioning Manager for Software environment variables

```
TIO_HOME=/opt/ibm/tivoli/tpmfsw
TIO_HOSTNAME=jiga.itsc.austin.ibm.com
TIO_LOGS=/usr/ibm/tivoli/common/COP/logs
TIO_SERVER=server1
TIO_WAS_USER_SCRIPT=/opt/ibm/tivoli/tpmfsw/tioprofile/bin/setupCmdLine.
sh
WAS_HOME=/opt/IBM/WebSphere/AppServer
WAS_PROFILE_HOME=/opt/ibm/tivoli/tpmfsw/tioprofile
WAS_PROFILE_NAME=tioprofile
```

11.2 Tivoli Provisioning Manager for Software server

The Tivoli Provisioning Manager for Software (or Tivoli Provisioning Manager) server consists of several different engines that run within it, where each engine manages different functionality. Figure 11-1 shows an overview of the Tivoli Provisioning Manager server infrastructure, which is useful in understanding how the components fit together for troubleshooting. This section discusses each of these components and tells you how to collect and interpret their log files.

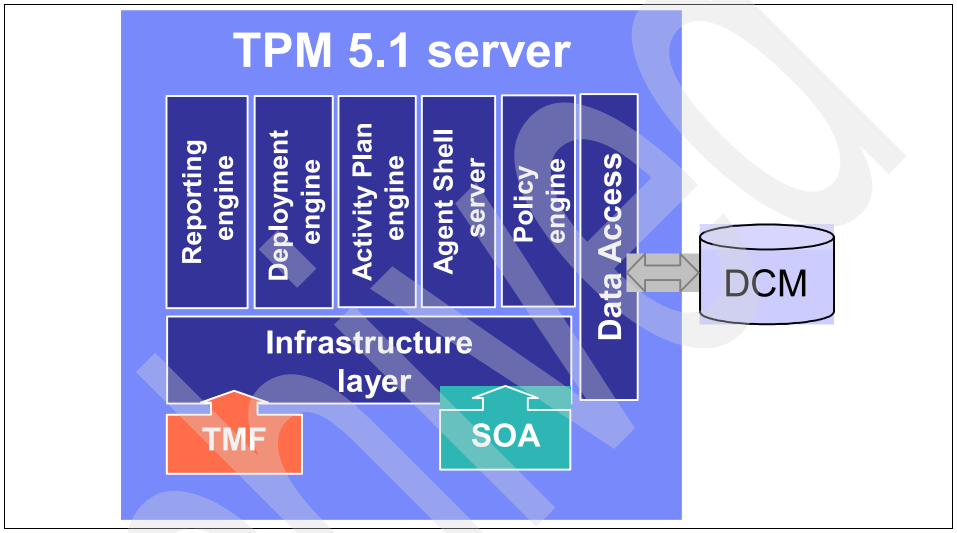


Figure 11-1 Tivoli Provisioning Manager architecture

11.2.1 Tivoli Provisioning Manager server status

The Tivoli Provisioning Manager server comes with a shell/cmd script, called **tioStatus**, which you can use to validate that the server is functional. This script is located in the $TIO_HOME/tools directory and takes the WebSphere username and password as arguments. Example 11-2 shows how you can check the status of the Tivoli Provisioning Manager server.

Example 11-2 Checking the status of the Tivoli Provisioning Manager server

```
[tioadmin@jiga tools]$ ./tioStatus.sh wasadmin wasadmin
ADMU0116I: Tool information is being logged in file

/opt/ibm/tivoli/tpmfsw/tioprofile/logs/server1/serverStatus.log
ADMU0128I: Starting tool with the tioprofile profile
ADMU0500I: Retrieving server status for server1
```

```
ADMU0508I: The Application Server "server1" is STARTED
2007-04-27 14:03:20,213 INFO  log4j configureAndWatch is started with
configuration file: /opt/ibm/tivoli/tpmfsw/config/log4j-util.prop
2007-04-27 14:03:22,544 INFO  COPCOM421I The deployment engine is
started.
2007-04-27 14:03:30,603 INFO  COPCOM423I The policy engine is started.
2007-04-27 14:03:34,550 INFO  COPCOM484I The agent shell server is
started.
2007-04-27 14:03:39,569 INFO  COPCOM560I The activity plan engine is
started.
```

In Example 11-2 on page 330, the command **`./tioStatus.sh wasadmin wasadmin`** is executed with the WAS administrator user ID (**`wasadmin)`** and the WAS password **`(wasadmin).`** The results indicate that the Application Server is running and that the deployment engine, the policy engine, the agent shell server, and the activity plan engine are all started.

The location of the logs for Tivoli Provisioning Manager for Software follow the Tivoli Common Directory standard. For Tivoli Provisioning Manager for Software, all logs are written to a central location that starts from $TIO_LOGS, on Unix systems, or %TIO_LOGS%, on Windows systems. We show the log locations for each Tivoli Provisioning Manager for Software engine component in the following sections.

11.2.2 Log files

Each Tivoli Provisioning Manager for Software JVM™ directory includes five log files, which we describe in Table 11-3.

Table 11-3 Common log names and contents

Log file name	Default Level	Description
console.log	info	Stores all event logs including messages, traces, and debugging information that shows up in trace.log and msg.log.
trace.log	error	Stores errors that are reviewed by IBM Tivoli Support.

Log file name	Default Level	Description
msg.log	MSG_INFO#com.thinkdynamics.kanaha.util.logging.MessageLevel	Stores the globalized event messages so that the user can understand a problem and take action to resolve the problem. The non-globalized event messages are also stored in console.log.
cbe.log	error	Stores all error messages in common the base event format.
heartbeat.log	info	Stores all heartbeat messages

11.2.3 Changing logging levels

Log data, in Tivoli Provisioning Manager for Software, is managed by log4j, which is an established open source logging tool. This section details the customized Tivoli Provisioning Manager for Software settings and tells you how you can modify settings dynamically. For complete log4j documentation, access the following Web site:

http://logging.apache.org/log4j/docs/documentation.html

The data from console.log, msg.log, trace.log and cbe.log are recorded based on the default logging levels and configuration parameters set in the log4j.prop file or in the log4j-util.prop file. Use the log4j-util.prop file to configure the logging for all scripts located in the $TIO_HOME/tools directory. Table 11-4 shows the possible debug levels and how they relate to each other.

Table 11-4 Debug levels and their relationships

Log level	Description
ALL or DEBUG	Highest level of debugging, and includes messages from all levels below.
INFO	Default level of debugging for console.log, and includes all levels below.
WARN	*Unused*. Warning level, includes all levels below.
ERROR	Default for trace.log and cbe.log and only logs errors and fatal messages.
FATAL	*Unused.*
OFF	Provides the ability to disable ALL logging.

To change the log level, you must alter the log4j.prop, which is located in $TIO_HOME/config. Practically speaking, you will only alter the log level for the 'console.log file since it includes all messages sent to the other logs. To achieve your desired log level, you have to make several changes to the log4j.prop file:

- **`log4j.rootCategory`**

  The default entry is:

  ```
  log4j.rootCategory=INFO, consolefile, errorfile, messagefile
  ```

  Change the default entry to:

  ```
  log4j.rootCategory=DEBUG, consolefile, errorfile, messagefile
  ```

  You have now configured the master logging level for the console, trace, and msg log files to be DEBUG; however, each appended entry in the file has its own filter. The only one of interest for us is the **`consolefile`** entry.

- **`log4j.appender.consolefile.threshold`**

  The default entry is:

  ```
  log4j.appender.consolefile.threshold=info
  ```

  Change the default entry to:

  ```
  log4j.appender.consolefile.threshold=debug
  ```

11.2.4 Deployment engine

Tivoli Provisioning Manager server uses the deployment engine to manage the workflows that are executed for any task that occurs in the Tivoli Provisioning Manager server, which includes Software Management tasks as well as Discovery tasks. You can find logs related to this component in:

```
$TIO_LOGS/deploymentengine
```

You can filter the deployment engine console.log file based on Deployment Request IDs. The simplest method is to look for the following line:

```
2007-04-30 22:20:44,160 INFO  [RequestDispatcher Thread]
(RequestDispatcher.java:183) engine.RequestDispatcher: Received
deployment request id=11221
```

From this point, to determine all of the activities associated with this request, you can use the following syntax:

```
grep 'request id' console.log
```

In `grep 'request id' console.log`, '*request id*' comes from the previously mentioned log file line. Note that when other operations execute deployment requests, such as tasks and activity plans, they also record a request ID that can you can use to search the deployment engine log files for relevant information.

11.2.5 Activity plan engine

The activity plan engine is the gateway to the underlying infrastructure that the Tivoli Provisioning Manager server actually uses to manage endpoints. It currently supports activity tasks through the Tivoli Management Framework, the SDI, or the DE. All tasks that show up under the Track Tasks interface, in the Tivoli Provisioning Manager user interface, may log information using the activity plan engine component. Logs related to this component can be found in:

```
$TIO_LOGS/activityplan
```

The activity plan engine includes many threads that you can use to isolate sections of the console.log file and to help in problem isolation. All log lines contain the name of the thread used. You can use the following types of threads to search in the console.log file:

- Activity Plan Engine Scheduler
- Status Updater
- DiscoveryStatusUpdater
- Cancel Request Processor
- Final Activity Processor
- Suspend/Resume Request Processor
- Activity Execution Threads (**`activity name(activity id)`**)

You can use **`grep`** to focus on a specific plan, activity, or type of action:

- grep "*name of the thread*" console.log

  "name of thread" is from the thread list

- grep "*name of activity*" console.log

  "name of activity" is your task name, as it shows up in the "Track Tasks" page.

- grep "*activity id*" console.log

11.2.6 Policy engine

The policy engine performs the orchestrated provisioning functions in Tivoli Provisioning Manager, such as capacity-on-demand and also the resource reservation functions in Tivoli Provisioning Manager. This process runs in both Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager products.

You can find the logs related to the policy engine in:

```
$TIO_LOGS/policyengine
```

You look in this log file if you experience problems in the orchestrated provisioning or the reservation functions. However neither of these functions are integral to the Tivoli Provisioning Manager for Software functionality.

11.2.7 Agent shell server

The agent shell server executes scriptlets on common agents. When executing workflows, scriptlets perform local script-type actions on a common agent. The agent shell server provides the means for workflows to perform the calls to the common agent.

You can find logs related to scriptlet execution in:

```
$TIO_LOGS/agentshellserver
```

11.2.8 Remote Execution and Access

Remote Execution and Access (RXA) is a Java toolkit for agent-less remote management that Tivoli Provisioning Manager for Software uses. Using existing industry protocols, such as SSH and Service Message Block (SMB), RXA provides basic remote management functionality that Tivoli Provisioning Manager for Software uses for the following items:

- Target discovery and identification
- Retrieval of target information
- Remote file and directory operations, including file transfer
- Remote command execution

You can find the RXA-specific logs in:

```
$TIO_LOGS/deploymentengine
```

General RXA activity is noted in the console.log, which you can observe by searching the log for ‘RXA’, for example:

```
grep ‘RXA’ $TIO_LOGS/deploymentengine/console.log
```

You can find more detailed RXA activity in the msg_rxa.log and the trace_rxa.log, which is located in the same directory. You can modify the logging level for these logs in $TIO_HOME/config/jlog.properties, according to the comments in that file.

Example 11-3 Changing the logging levels

```
# To enable trace logging for RXA comment the following two lines
# and uncomment next section
remoteAccess.logger.listenerNames=remoteAccess.handler.message
remoteAccess.logger.level=INFO

# To enable trace logging for RXA uncomment the following two lines
#remoteAccess.logger.level=DEBUG_MAX
#remoteAccess.logger.listenerNames=remoteAccess.handler.trace
remoteAccess.filter.messages
```

Note: The values for the following attribute are on one line: `remoteAccess.logger.listenerNames`

11.2.9 User interfaces

The components of the user interfaces for Tivoli Provisioning Manager for Software provide functional interaction through the Web User Interface and the Web Services command line interface, which are implemented in J2EE™. Logs that are related to problems with these interfaces are stored in $TIO_LOGS/j2ee.

11.3 SDI infrastructure

In this section, we discuss how to troubleshoot SDI.

11.3.1 Device management service

The device management service (DMS) manages work that is delegated to endpoints, such as discoveries, tasks, or distributions, either through caching servers (eDMS federated agents) or through subagents. Agents contact the caching servers periodically to get queued jobs and to process them. The agents then process the jobs and report the results to the federating server, which allows the device management service infrastructure to support intermittent connections between the caching servers and endpoints.

Figure 11-2 illustrates the device management service Infrastructure.

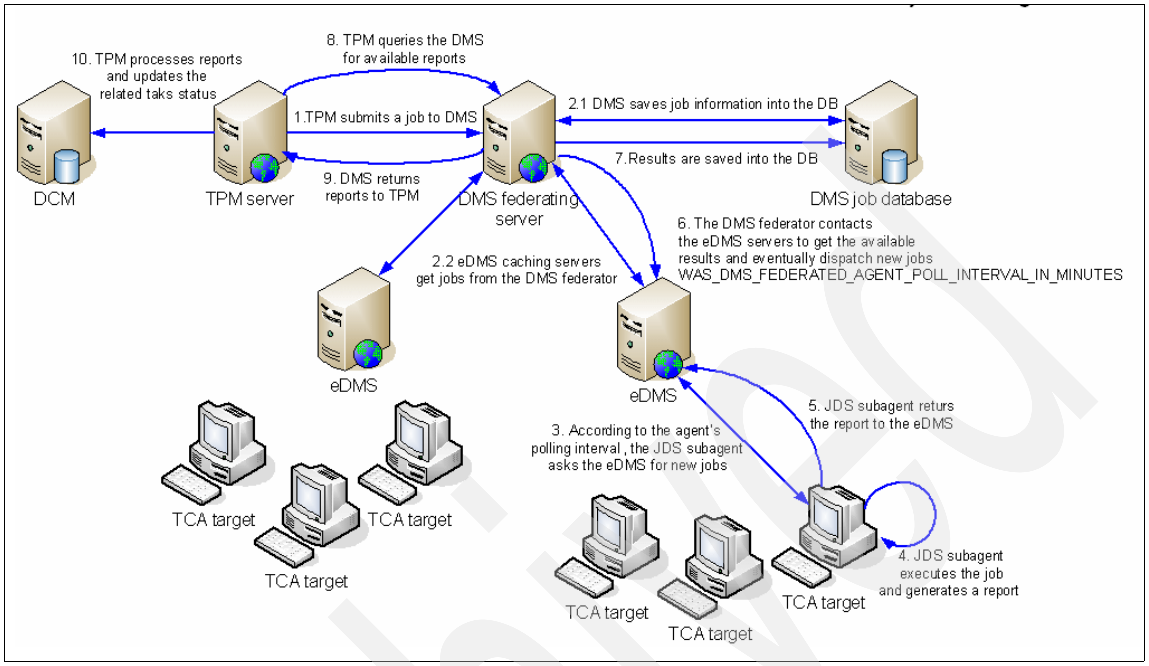

10. TPM processes reports and updates the related taks status
8. TPM queries the DMS for available reports
1.TPM submits a job to DMS
9. DMS returns reports to TPM
2.1 DMS saves job information into the DB
7.Results are saved into the DB
DCM
TPM server
DMS federating server
DMS job database
2.2 eDMS caching servers get jobs from the DMS federator
6. The DMS federator contacts the eDMS servers to get the available results and eventually dispatch new jobs WAS_DMS_FEDERATED_AGENT_POLL_INTERVAL_IN_MINUTES
eDMS
eDMS
5. JDS subagent returs the report to the eDMS
3. According to the agent's polling interval, the JDS subagent asks the eDMS for new jobs
4. JDS subagent executes the job and generates a report
TCA target
TCA target
TCA target
TCA target
TCA target
TCA target

Figure 11-2 Device management service architecture

The device management service is described in detail in 3.2.3, “Device management service” on page 51. The device management service specific logs for troubleshooting are located in:

```
$TIO_HOME/tioprofile/logs/server1/
```

There are many logs located in this directory; however, when you troubleshoot device management service related issues, use the following logs:

- TraceDMS<n>.log
- DMSMsg<n>.log
- SystemOut.log
- SystemErr.log

To increase the debug level for the device management service server, edit the file traceConfig.properties. Depending on the type of device management service server you deployed, you need to change the file in one of the following directories:

- For device management service only, go to: $WAS_PROFILE_HOME/installedApps/<hostname>Node01Cell/DMS_WebApp.ear/dmserver.war/WEB-INF/classes

- For Federated DMS (all Device Manager Servers in a cluster) go to: $TIO_HOME/DeviceManager/config/dmserver.war/WEB-INF/classes

In the traceConfig.properties file, make the following changes to increase the debug level.

1. Set the TraceLevel. The default TraceLevel is 1. There are four possible TraceLevels: 0, 1, 2, 3. 3 is the lowest level with the most detail shown. 0 means no trace messages will be shown.

   ```
   TraceLevel=3
   ```

2. Increase the trace information. This determines what trace information is included on the trace record.

   ```
   DISPLAY_ENTRY_EXIT=true
   ```

3. Set the maximum file size of each file.

   ```
   MaxFileSize=55555
   ```

4. Set the maximum number of files (when the max is reached the oldest one gets deleted).

   ```
   MaxTraceFiles=10
   ```

5. Ensure that the trace is enabled for components.

   ```
   component.console=true
   component.dmserver=true
   component.event=true
   component.notification=true
   component.plugins=true
   component.resultscollector=true
   component.twgapi=true
   component.database=true
   component.enrollserver=true
   component.datconverter=true
   component.mcollect=true
   component.notificationhandler=true
   component.api=true
   component.userservices=true
   component.federator=true
   ```

6. Reload the trace properties for new logging levels to take affect.

   ```
   https://<fullyQualifiedHostNameOfServer>:9045/dmserver/TraceServlet?
   trace=set
   ```

   This should return SUCCESS!

At any time, you can run the command in step number six to just check the status of the device management service server.

The TraceDMS.out file at TraceLevel 3 shows all of the interaction of the device management service with its databases when it receives jobs and dispatches jobs. The start of the job is determined by the string *createJob*, and if there are multiple entries when you search, you have to match the time you think the job starts. Example 11-4 shows what the start of a job submission looks like.

Example 11-4 TraceDMS1.log job start

```
05/15/2007 4:56:06.178 com.tivoli.dms.federator.FederatedJobManager
createJob Migrated.Servlet.Engine.Transports : 284
  Job Object len in 9450

05/15/2007 4:56:06.179 com.tivoli.dms.federator.FederatedJobManager
createJob Migrated.Servlet.Engine.Transports : 284
  Create Federator job -
```

Another important piece of information that you can find in the TraceDMS<n>.log is when, or if, a common agent client connects to the device management service to receive jobs. Example 11-5 shows the agent, nice.itsc.austin.ibm.com, checking into the device management service server.

Example 11-5 TraceDMS1.log TCA checking in

```
05/15/2007 4:50:08.273 com.tivoli.dms.plugin.syncmldm.OMADMServlet
doPost Migrated.Servlet.Engine.Transports : 278
  deviceObj=
SyncML DM Device Info ------------------
   SyncML <Hdr> <Source>  :9FDD7F2E0F9A3718A078DB894BD8E6C3
   SyncML <Hdr> <Target>
:https://9.3.4.163:9045/dmserver/SyncMLDMServlet
   Hostname    :nice.itsc.austin.ibm.com
   POST Counter:1
   Client Pwd Fail Counter:0
   Challenge Server Counter:0
   ContentType :1 (application/vnd.syncml.dm+wbxml)
   SecureTransport:true
   Client:
      ServerReqClientAuth:none
      ActualClientAuth   :null
      ClientDeviceAuth  :true
      ClientCredentialStatus  :212
   Server:
      ClientReqServerAuth:basic
      ActualServerAuth   :null
      ServerAuth  :false
      ServerCredentialStatus  :200
   MaxMsgSize  :20000
   MaxObjSize  :100000
   Locale      :en_US
   ./DevInfo/Man         :IBM
   ./DevInfo/Model       :tpmOSGi
   ./DevInfo/DevId       :9FDD7F2E0F9A3718A078DB894BD8E6C3
   ./DevInfo/Lang        :en
   ./DevInfo/DmV         :2.0.0
```

11.3.2 Dynamic content delivery service

Tivoli Provisioning Manager uses the dynamic content delivery service infrastructure to distribute files to common agents. This process is described in detail in 3.2.3, “Device management service” on page 51, but in general, the distribution consists of two parts:

1. Tivoli Provisioning Manager publishes a file to the dynamic content delivery service infrastructure. Figure 11-3 shows the flow of publishing a file to the depots so that a common agent can download them. When problems occur with the distribution, the first thing to confirm is that the files were published correctly.

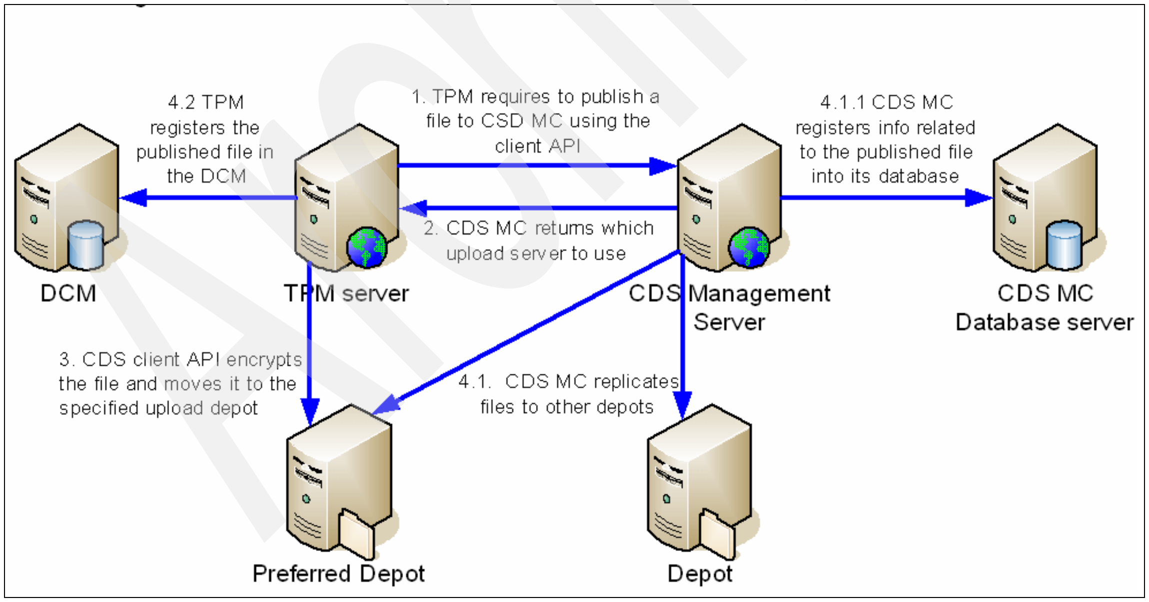

4.2 TPM registers the published file in the DCM
1. TPM requires to publish a file to CSD MC using the client API
4.1.1 CDS MC registers info related to the published file into its database
2. CDS MC returns which upload server to use
DCM
TPM server
CDS Management Server
CDS MC Database server
3. CDS client API encrypts the file and moves it to the specified upload depot
4.1. CDS MC replicates files to other depots
Preferred Depot
Depot

Figure 11-3 Publishing a file to a dynamic content delivery service

2. Common agents connect to the infrastructure and download the files. After the files are published, a job is scheduled with the device management service that informs the common agents that they need to connect to the dynamic content delivery service management server to receive a download plan. If a distribution is failing, but the file was published, then check the common agents to determine the cause. Figure 11-4 shows how common agents connect and download files.

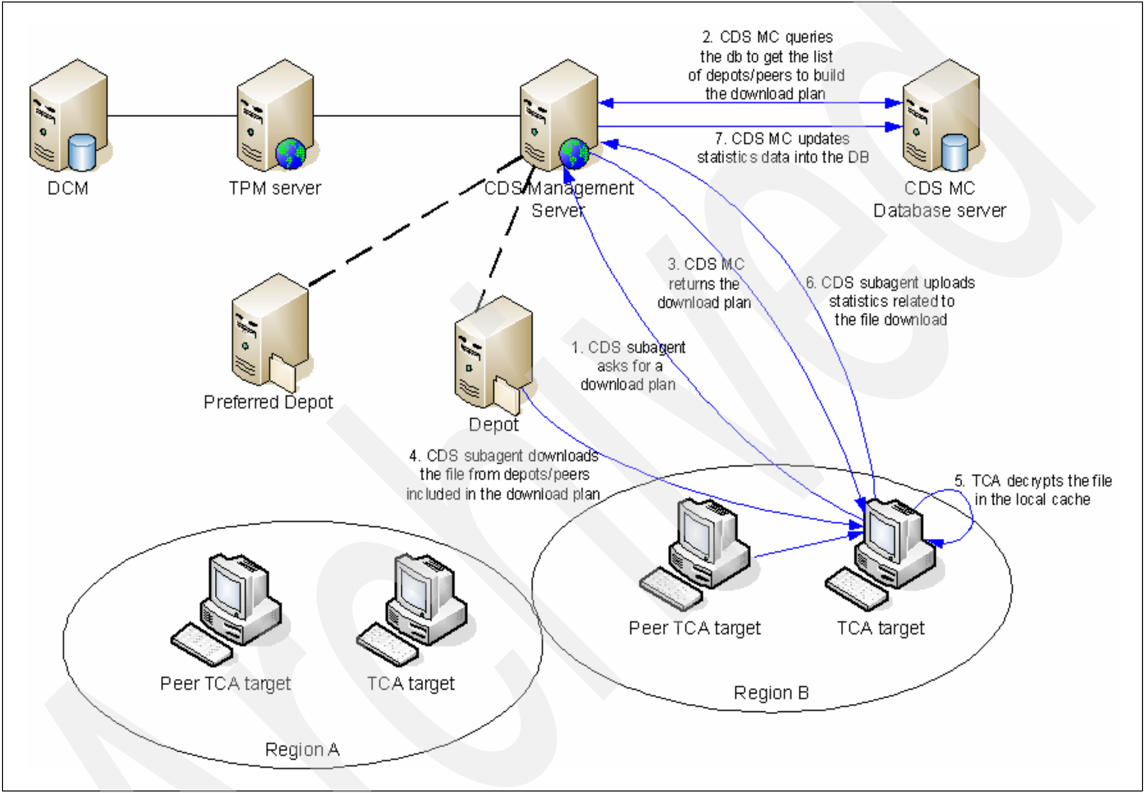

DCM
TPM server
CDS Management Server
CDS MC Database server
2. CDS MC queries the db to get the list of depots/peers to build the download plan
7. CDS MC updates statistics data into the DB
3. CDS MC returns the download plan
6. CDS subagent uploads statistics related to the file download
1. CDS subagent asks for a download plan
Preferred Depot
Depot
4. CDS subagent downloads the file from depots/peers included in the download plan
5. TCA decrypts the file in the local cache
Peer TCA target
TCA target
Region B
Peer TCA target
TCA target
Region A

Figure 11-4 Downloading a file from a dynamic content delivery service

Dynamic content delivery service management center

This section describes how to configure logging and tracing for the dynamic content delivery service management center. The logging and tracing properties for the management center are stored locally in the cdslog.properties file. To configure the logging and tracing properties for a component, edit the cdslog.properties file for that component.

Table 11-5 lists the components associated with the dynamic content delivery service management center.

Table 11-5 Dynamic content delivery service management center components

Component name	Description
downloadGrid	Receives and processes download plans.
downloadGridSecurity	Handles Web services authentication and authorization.
distribution	Handles distribution of files to targets.
monitoring	Keeps track that services are running.
portal	Runs the Admin GUI.
remoteAccess	Handles RXA access for dynamic content delivery service.

Enabling and disabling logging and tracing

By default, logging and tracing are enabled for all components of the content delivery service. To disable logging and tracing for a component, set the component_name.logger.message.logging property, in the cdslog.properties file of that component, to false. For example, to disable logging and tracing for the dynamic content delivery service Administrator Portal, set the portal.logger.message.logging property in the cdslog.properties file to false as shown in the following example:

```
portal.logger.message.logging=false
```

To enable logging and tracing for a component, set the component_name.logger.message.logging property in the cdslog.properties file of that component to true. For example, to enable logging and tracing for the dynamic content delivery service Administrator Portal, set the portal.logger.message.logging property in the cdslog.properties file to true, as shown in the following example:

```
portal.logger.message.logging=true
```

Setting logging and tracing levels

Table 11-6 provides the levels of logging and tracing for the content delivery service.

Table 11-6 Log and trace level descriptions

Level	Description
ERROR	Only error messages are logged. No tracing is logged.
WARN	Only warning and error messages are logged. No tracing is logged.
INFO	All messages (informational, warning, and error) are logged. No tracing is logged.
DEBUG_MIN	All messages are logged. Minimal tracing information is logged.
DEBUG_MID	All messages are logged. Moderate tracing information is logged.
DEBUG_MAX	All messages are logged. All tracing information is logged.

By default, the logging and tracing level of all components is set to DEBUG_MIN for the trace file and INFO for the msg file. To configure the logging and tracing level of a component, set the component_name.logger.level property in the cdslog.properties file of that component to the desired level. For example, to set the logging and tracing level for a distribution to the maximum setting, set the distribution.logger.level property in the cdslog.properties file to DEBUG_MAX, as shown in the following example:

```
distribution.logger.level=DEBUG_MAX
```

Location of files

By default, the management center log and trace files and the logging properties files are stored in the following locations:

Log and trace files	$TIO_HOME/../common/ctgde/logs
Log properties file	$TIO_HOME/CDS/logprop

Description of files

Table 11-7 contains log and trace files, and their descriptions, that in are stored in the management center.

Table 11-7 Dynamic content delivery service management center log and trace files

Log file name	Description
msg_admingui.log trace_admingui.log	Message and trace information for the administration GUI.
msg_cdsmgr.log trace_cdsmgr.log	Message and trace information for download plans as well as general message and trace information.
msg_cdssec.log trace_cdssec.log	Message and trace information for security.
msg_distribution.log trace_distribution.log	Message and tracing information for the distribution agent.
msg_monitoring.log trace_monitoring.log	Message and tracing information for the monitoring agent.
msg_client.log trace_client.log	Message and tracing information for the download applet.
msg_EPM_Install.log trace_EPM_Install.log	Message and tracing information about the installation of the management center.
trace_rxa.log msg_rxa.log	Logging information about the Remote Execution and Access component.
CDS_DB2_install_schema .log	Logging information for the DB2 installation.

The log and trace files are in ASCII format. Log and trace files are limited to 8 KB. When a file reaches that size, a new file is created and a “1” is appended to the file name. When *file_name*1 reaches 8 KB, a new file is created, and a “2” is appended to the file name. When *file_name*2 reaches 8KB, the cycle starts over, and the original file is overwritten.

Dynamic content delivery service status and configuration

You can examine the details of the dynamic content delivery service management center server by using the dynamic content delivery service Administration Portal, which you can access from the following URL:

```
https://tpmhostname:9045/admin
```

When prompted, use the **tioappadmin** account that you use to access the Tivoli Provisioning Manager server, by default. You can also specifically alter configuration items using the following URL:

```
https://tpmhostname:9045/admin/configuration
```

Warning: Only use the configuration URL when directed by IBM Support. Making changes to the configuration this way can damage the dynamic content delivery service server.

Dynamic content delivery service depots

This section describes the log and trace files that are stored on the depot server. In the context of dynamic content delivery service, depots are installed to common agents and they use the same files to configure logging. For this section, we use $INSTALL_DIR to indicate the directory where the common agent is installed.

Changing log and trace levels

By default, the logging level of the cds_depot component is set to INFO and the trace level is set to DEBUG_MIN. The maximum tracing level for the cds_depot component is DEBUG_MAX or ALL. To change this setting, set **tracer.cds_depot.level** to ALL, as shown in the following example:

```
tracer.cds_depot.level=ALL
```

To change the log level for cds_depot, change the parameter **logger.cds_depot.level**in the file logging.properties, as shown in the following example:

```
logger.cds_depot.level=ALL
```

Note: By default the log file is set much lower than the trace file and contains much less information. However, the trace file does contain the same information as the log file, so it is not necessary to change both logging levels.

Location of files

By default, the depot server log files are stored in the following locations:

Log and trace files	$INSTALL_DIR/ep/runtime/base/logs
Log properties file	$INSTALL_DIR/ep/runtime/agent/config
Storage	$INSTALL_DIR/ep/runtime/agent/subagents/cds/data/files/1

Description of files

The log and trace files Table 11-8 are stored on the depot server.

Table 11-8 Depot Server log files

Log file names	Description
cds_msg_depot.log	Information about the depot server.
cds_trace_depot.log	Trace records for the depot server.

Depot status

To check if the depot is active, type the following command:

```
telnet depot-server 2100
```

From the telnet session, type the command **syst**. The response is:
`DS:<server_name>=1.3.1.0`

Common agent log and trace files

This section describes the specific configuration parameters, as well as the log and trace files, for the dynamic content delivery service client. The dynamic content delivery service client is a subagent of the common agent. For this section, we use $INSTALL_DIR to indicate the directory where the common agent is installed.

Changing log and trace levels

By default, the logging level of the cds_client component is set to INFO and the trace level is set to DEBUG_MIN. The maximum tracing level for the cds_client component is DEBUG_MAX or ALL. To change this setting, set the **tracer.cds_client.level** to ALL as in the following example:

```
tracer.cds_client.level=ALL
```

To change the log level for cds_client, change the parameter **logger.cds_client.level** in the file logging.properties, as shown in the following example:

```
logger.cds_client.level=ALL
```

File Locations

By default, the client log and trace files are stored in the following locations:

Log files	common agent$INSTALL_DIR/ep/runtime/base/logs
Log file properties	$INSTALL_DIR/ep/runtime/agent/config

Description of files

Table 11-9 shows the og and trace files, which are stored on the client.

Table 11-9 Log and trace files that are stored on the client

Log file names	Description
cds_msg_client.log	Information about the client manager
cds_trace_client.log	Client trace records

11.3.3 Common Agent Services

This section contains information to help you resolve problems with Common Agent Services (CAS), which includes the agent manager and common agent.

Agent manager

To verify that agent manager is installed correctly, run the following TCA_PingAgentManager workflow.

To validate the agent manager installation, you can access the following URL, which displays information about the agent manager configuration:

```
http://localhost:9513/AgentMgr/Info
```

When you troubleshoot installation errors, you can find installation log files in $AM_HOME/logs.

You can find the agent manager runtime logging in the standard WAS SystemOut.log and SystemErr.log files, on an Enterprise installation, located on the $TIO_HOME/tioprofile/logs/server1, by default.

In the SOA infrastructure, the agent manager logs into the LWI_HOME/logs/rcp.log.0 file, by default C:/Program Files/IBM/tivoli/tpm/logs/rcp.log.0.

You can find installation logs in LWI_HOME\runtime\agentmanager\logs.

Determining the version of agent manager

To determine the version of the agent manager, use the **`GetAMInfo`** command, which is located in the Agent_Manager_install_dir/bin directory.

- On Windows systems, it has the .cmd extension (**`GetAMInfo.cmd`**).
- On AIX, Linux, and Solaris systems, is has the .sh extension (**`GetAMInfo.sh`**).

To display the version of the agent manager and agent recovery service applications, run the following command for an AIX, Linux, or Solaris system:

```
Agent_Manager_install_dir/GetAMInfo.sh
```

To display the version of the agent manager application, run the following command for a Windows system:

```
Agent_Manager_install_dir\GetAMInfo.bat AgentManager
```

Note: The application name, AgentManager, is not related to the name of the application server in which you install agent manager. For example, if you install the agent manager into the application server named server1, then the application name remains AgentManager. To display the version of the agent recovery service on an AIX, Linux, or Solaris system, run the following command:

```
Agent_Manager_install_dir/GetAMInfo.sh AgentRecoveryService
```

Health Check

You can verify the correct function of the agent manager with the **`HealthCheck.sh/bat`** script in the $AM_HOME/toolkit/bin directory. To see the available arguments run:

```
C:\PROGRA~1\ibm\AgentManager\toolkit\bin>healthcheck -?
```

See Example 11-6.

Example 11-6 Running the Agent Manager Health Check report script

```
C:\PROGRA~1\ibm\AgentManager\toolkit\bin>cmd /c HealthCheck.bat
-RegistrationPW <tioadmin_password>
        1 file(s) copied.
Tool Launcher is trying to instantiate Command line tool
com.tivoli.cas.manager.
tools.HealthCheck ...
Command Line Tool com.tivoli.cas.manager.tools.HealthCheck succesfully
instantiatied.
May 3, 2007 10:13:54 AM CDT  Arguments passed to Command Line Tool:
-HOST localhost -RegistrationPW itso05
May 3, 2007 10:13:55 AM CDT  Initializing configuration with
file:C:\PROGRA~1\ibm\AgentManager\toolkit\bin\config\endpoint.propertie
s
May 3, 2007 10:13:58 AM com.tivoli.agentmgr.credentialmgr.ARSPKIClient
getProxyConfiguration
SEVERE: CTGEM0016E The ARS.port.public parameter cannot be NULL.
```

```
May 3, 2007 10:13:58 AM
com.tivoli.agentmgr.util.security.CRLTrustManager getCRL SEVERE:
BTC1048E
May 3, 2007 10:13:58 AM com.tivoli.agentmgr.credentialmgr.ARSPKIClient
getProxy Configuration SEVERE: CTGEM0016E The ARS.port.public parameter
cannot be NULL.
May 3, 2007 10:13:58 AM
com.tivoli.agentmgr.util.security.CRLTrustManager getCRL SEVERE:
BTC1048E
May 3, 2007 10:13:58 AM
com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCache
Item tryConnect
INFO: NOTE  ==>Connected to host=localhost on port=9511
May 3, 2007 10:13:58 AM
com.tivoli.agentmgr.client.proxy.WSDLClient$AddressCache
Item directConnect
INFO: Directly connected
Agent Manager Name:
ibm-cdm:///CDM-ManagementSoftwareSystem/TivoliGUID=4CDEB4F1E95411DBA35B
00164139637C,InstallPath=file%3A%2F%2F%2FC%3A%2FProgram%20Files%2FIBM
%2FAgentManager,Feature=CTGEM
Registration.domain                           = itsc.austin.ibm.com
CA.keyRing.name                               = certs/CARootKeyRing.jks
CA.Certificate.Root.Alias                     = rootcert
CA.Key.Root.Alias                             = rootkey
CA.CRL.TimeToLive                             = 24
CA.CRL.filename                               =
certs/CertificateRevocationList
Registration.Agent.Reregistration.Policy      = Any
Registration.Agent.Certificate.Duration       = 365
Registration.Manager.Certificate.Duration     = 3600
CA.Certificate.graceTime                      = 1380
Config.Server.Host                            = 9.3.5.99
Config.Server.Port                            = 9512
Config.URI                                    =
/AgentMgr/ConfigurationUpdate
CertManagement.Host                           = 9.3.5.99
CertManagement.Renewal.Port                   = 9512
CertManagement.Renewal.URI                    =
/AgentMgr/CertificateRenewal
CertManagement.CRL.Port                       = 9513
CertManagement.CRL.URI                        = /AgentMgr/CRLRequest
CertManagement.Revoke.Port                    = 9512
CertManagement.Revoke.URI                     =
/AgentMgr/CertificateRevocation
```

```
AgentQuery.Host                            = 9.3.5.99
AgentQuery.Port                            = 9512
AgentQuery.URI                             = /AgentMgr/AgentQuery
CommonAgentQuery.URI                       =
/AgentMgr/CommonAgentQuery
AgentConfiguration.Host                    = 9.3.5.99
AgentConfiguration.Port                    = 9512
AgentConfiguration.URI                     =
/AgentMgr/AgentConfiguration
AgentManagerQuery.Host                     = 9.3.5.99
AgentManagerQuery.Port                     = 9511
AgentManagerQuery.URI                      =
/AgentMgr/AgentManagerQuery
Registration.Host                          = 9.3.5.99
Registration.Port                          = 9511
Registration.URI                           = /AgentMgr/Registration
Status.timeToLive                          = 0
ARS.directory                              = C:/Program
Files/IBM/AgentManager
ARS.port.base                              = 9511
ARS.port.secure                            = 9512
ARS.port.public                            = 9513
ARS.URI.root                               = /AgentMgr
ARS.security.enabled                       = true
Status.Authorization.Required              = true
Access.restriction.revocation              = true
Access.restriction.Configuration           = true
Query.Agent.Max.Return                     = -1
Query.Database.Type                        = db2
ARS.version                                = 1.3.2.12
Key.Algorithm.Name                         = RSA
Config.Listener.Manager                    =
com.tivoli.agentmgr.spi.providers.makeAgentRegistryUpdate,
com.tivoli.agentmgr.cert.AgentStatusChangeListener
Config.Listener.Agent                      =
com.tivoli.agentmgr.spi.providers.makeAgentRegistryUpdate
Registration.Listeners.Manager.Request     =
com.tivoli.agentmgr.registration.AuthorizationValidator,
com.tivoli.agentmgr.registration.AuthorizationTestOnly,
com.tivoli.agentmgr.registration.AuthorizationTestOnly,
com.tivoli.agentmgr.registration.AgentReregistrationTest
Registration.Listeners.Manager.Issue       =
com.tivoli.agentmgr.registration.StoreCertificateListener
Registration.Listeners.Agent.Request       =
com.tivoli.agentmgr.registration.SimplePWRequestValidator,
com.tivoli.agentmgr.registration.AuthorizationTestOnly,
com.tivoli.agentmgr.registration.AgentReregistrationTest
Registration.Listeners.Agent.Issue          =
com.tivoli.agentmgr.registration.StoreCertificateListener
May 3, 2007 10:13:58 AM CDT  Health Check passed.
May 3, 2007 10:13:58 AM CDT  Command Line Tool execution successful.
```

Packaging the log files

If you encounter a problem that you cannot resolve immediately, collect a copy of the log files and system configuration information, which preserves the information you need to perform a detailed problem determination and prevents you from having to scan back through messages and trace events that were recorded after the problem occurred.

If you contact IBM Customer Support, you must provide log files package. To package the logs and configuration information into a compressed, or zipped, file for IBM Customer Support, use the log collection tool that is available in the toolkit subdirectory $AM_HOME/toolkit/bin to collect and zip log files from the agent manager to send to IBM support.

You must call the LogCollector script from the $AM_HOME/toolkit/bin directory. The zip file is created in the $AM_HOME directory.

Example 11-7   LogCollector script

```
C:\PROGRA~1\ibm\AgentManager\toolkit\bin>cmd /c logcollector.bat
Tool Launcher is trying to instantiate Command line tool
com.tivoli.cas.manager.
tools.was.WASLogCollector ...
Command Line Tool com.tivoli.cas.manager.tools.was.WASLogCollector
succesfully i
nstantiatied.
May 3, 2007 11:31:41 AM CDT  Arguments passed to Command Line Tool:
May 3, 2007 11:31:43 AM CDT  Command Line Tool execution successful.
```

Locating the installation log files

The log files generated during the installation and initial configuration of the agent manager are located in the Agent_Manager_install_dir\logs directory.

The default directory for the agent manager installation is:

- %TIO_HOME%\AgentManager on Windows
- $TIO_HOME/AgentManager on UNIX or Linux

Table 11-10 lists the installation logs.

Table 11-10 Agent Manager installation log files

Log file name	Description
jacl/appServer_out.log jacl/appServer_err.log	Standard output and standard error logs generated while installing the application server for the agent manager. The EPMAppServer.jacl script generates these logs.
jacl/checkcell_out.log jacl/checkcell_err.log	Standard output and standard error logs for verifying the cell for the WebSphere configuration. The EPMValidate.jacl script generates these logs.
jacl/checknode_out.log jacl/checknode_err.log	Standard output and standard error logs for verifying the node for the WebSphere configuration. TheEPMValidate.jacl script generates these logs.
jacl/jdbc_out.log jacl/jdbc_err.log	Standard output and standard error logs for configuring the WebSphere JDBC provider, data source, and J2C Authentication Data Entry. The EPMJdbcProvider.jacl script generates these logs.
jacl/ssl_out.log jacl/ssl_err.log	Standard output and standard error logs for SSL configuration. The EPMSSLConfig.jacl script generates these logs.
jacl/virHost_out.log jacl/virHost_err.log	Standard output and standard error logs for creating the WebSphere virtual host. The EPMVirtualHost.jacl script generates these logs.

Locating the uninstalled log files

The log files that are generated when you uninstall the agent manager are located in the Agent_Manager_install_dir\logs directory. Table 11-11 lists the agent manager uninstallation logs.

Table 11-11 Agent Manager uninstallation log files

Log file name	Description
uninstall.log	InstallShield MultiPlatform (ISMP) log for uninstalling the agent manager.
AMUninstallReturnValues.log	Summary of return values of the steps of the agent manager uninstallation.
msg_EPM_Install.log trace_EPM_Install.log	Messages and trace information generated when you uninstall the agent manager applications in WebSphere Application Server.

Locating the remote registry logs

If the registry is on a system other than the agent manager server, install a subset of the agent manager files on the database server to create and initialize the registry database. Table 11-12 lists the logs that are created when you create the remote registry database.

Table 11-12 Remote registry database installation log files

Log file	Description
datastore.out	A log of the SQL statements that are run to create the registry database and its tables.
ds_install.log	An ISMP log for installing the files necessary to create the registry database.
db_stdout.log db_stderr.log	Standard output and standard error logs for creating the registry database. Check db_stdout.log first to see if the registry was created successfully.

Locating the runtime log files

The runtime logs for the agent manager are located in the was_install_dir\logs\app_server_name directory, where app_server_name is the name of the application server. By default, this is Agent Manager.

The runtime logs for WebSphere are located in the WAS_install_dir\logs\server1 directory. Additional logs are in the was_install_dir/logs/ffdc directory.

Locating the DB2 Universal Database runtime logs

DB2 Universal Database provides several First Failure Data Capture (FFDC) facilities that log information as errors occur. The DB2 Universal Database FFDC facilities are used with CLI and DB2 traces to diagnose problems.

The following list contains information that DB2 Universal Database for FFDC captures:

- db2diag.log

  When an error occurs, the db2diag.log is updated with information about the error. This is the primary log to use when debugging DB2 Universal Database problems.

- db2alert.log

  If an error is determined to be an alert, then an entry is made in the db2alert.log file and to the operating system or native logging facility.

- dump files

  For some error conditions, additional information is logged in external binary dump files, which are named after the failing process ID. These files are intended for DB2 Universal Database Customer Support.

- trap files

  The database manager generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception. Trap files contain a function flow of the last steps that were executed before a problem occurred.

Other useful DB2 Universal Database commands include:

- db2trc

  This command gives you the ability to control tracing.

- db2support

  This command collects environment information and log files and places them into a compressed archive file.

Installation and uninstallation verifications

In this section, we give you commands and processes that can help you verify installation or uninstallation.

- Verifying that the agent manager service is running

  If you suspect that the agent manager server is not running, then use the health check tools that are available in the toolkit subdirectory of the agent manager installation directory (Agent_Manager_install_dir).

  The WebSphere Application Server **`serverStatus`** command tells us if the application server is started; however, the command does not indicate if the application server is fully functional.

  For example, if a remote registry database is inaccessible because of a network or authorization problem, the **`serverStatus`** command reports that the application server is operational. However, the agent manager server cannot register agents or resource managers without a connection to the registry.

- Verifying that the WebSphere Application Server is installed

  To verify that the WebSphere Application Server is installed, use the firststeps tool, which is located in the WAS_install_dir/firststeps directory. Run the program for your operating system:

  – Windows: **`firststeps.bat`**
  – AIX, Linux, or Solaris: **`firststeps.sh`**

- Verifying that DB2 is installed

  If the registry is in a DB2 database and the agent manager server does not install or start properly, or if you receive errors about problems accessing the registry, then verify that DB2 Universal Database ESE installed properly and that the DB2 Universal Database server, that controls the registry database, is running.

  To verify that DB2 Universal Database ESE installed properly, refer to the DB2 Universal Database documentation for your operating system.
  The installation documentation contains information about verifying the installation using either the command line processor (CLP) or the First Steps GUI.
  The general procedure is the same for both methods:

  a. Create the SAMPLE database.
  b. Connect to the SAMPLE database.
  c. Execute a query against the SAMPLE database.
  d. Drop the SAMPLE database.

- Errors starting the agent manager application server

  If you cannot start the agent manager application server:

  – If you get a NoServerDefinedException when you use the WebSphere **`startServer`** command to start the agent manager application server, ensure that you specified the correct server name. Although the default name of the agent manager application server is AgentManager, the agent manager might be installed to a different application server, such as server1.

  – If you reconfigured the WebSphere Application Server on UNIX or Linux to run as a user other than root, and you get a FileNotFoundException exception or a “Permission denied” error, then make sure that the user ID that runs the WebSphere processes has permission to read the agent manager files, particularly the files in the Agent_Manager_install_dir/certs directory.

- Errors installing the agent manager on UNIX

  If an installation on a UNIX operating system fails while copying a file, or doing a put file, then the /tmp directory or device may be full on the UNIX computer.

  The install agent copies the file to the /tmp/tcatemp file.
  Run the df -k /tmp command to determine if the disk is full. If the disk is full, create the necessary space required in order for the installation to succeed.

- Errors starting the agent manager application

  If the logs indicate a problem starting the certificate authority, a common cause is a problem with the certificates. To correct this, you can replace your certificates.

- Security certificate error with the Firefox Web browser

  The following security error is generated by the Firefox Web browser when a user tries to log on to Tivoli Provisioning Manager for Software:

  ```
  “Your certificate contains the same serial number as another
  certificate issued by the certificate authority. Please get a new
  certificate containing a unique serial number.”
  ```

  The certificate that an agent manager instance generated has been trusted before and the certificate is stored locally on the customer side. When the user tries to log on to another instance of Tivoli Provisioning Manager for Software with another instance of the agent manager, the agent manager will generate another certificate but with the same serial number.

  To resolve this error, remove the certificates:

  a. In the Firefox Web browser, navigate to **Tools** → **Options** → **Advanced** → **Security** → **View Certificates**, and click **Web Sites**.

  b. Delete the certificates that appear in that section.

- Verifying the connection to the registry

  If any errors indicate that there is a problem accessing the registry, then verify that the registry is correctly configured and available by performing the following steps:

  a. Open the WebSphere Administrative Console, expand **Resources**, and click **JDBC Providers**.

  b. In the **Server** field, which is located on the **JDBC Providers** panel, type the name of the application server where the agent manager is installed, or click **Browse** to select it from a list. By default, this is AgentManager.

  c. Click **Apply** to list the JDBC providers for the agent manager application server.

  d. In the list of providers, click **AgentJDBCProvider**.

e. In the **Additional Properties**, which is located in the area at the bottom of the Configuration page of the JDBC provider, click **Data Sources**.

f. In the **Data Sources** page, check the box next to **AgentRegistry**, and click **Test Connection**.

g. Review the status message at the top of the Data Sources panel. A message similar to the following indicates that the JDBC connection is correctly configured: `Test connection for datasource AgentRegistry on server server1 at node WAShostname was successful.`

Enabling and disabling tracing

You can control whether trace information is captured. You can also control the amount of details in the agent manager trace levels. By default, tracing is disabled.

- Enabling tracing

  To enable tracing for the agent manager:

  a. Start the WebSphere Administrative Console at the following location: `http://{servername}:9044/admin`

  b. Click **Servers** → **Application Servers** → **app_server_name** → **Logging and Tracing** → **Diagnostic Trace Service**. Replace app_server_name with the name of the application server where the agent manager applications are installed, for example, AgentManager.

  c. In the **General Properties** area of the **Configuration** page, check the **Enable Log** option.

  d. In **Additional Properties**, click **Change Log Detail Levels**.

  e. Go to the **Runtime** page. In the **General Properties**, **Change Log Detail Levels** field, type one of the values specified in Table 11-13:

Table 11-13 Setting trace levels

Task	Value in Trace Specification text box
Turn on all tracing	mgr.*=all=enabled
Turn on the entry and exit tracing	mgr.*=entryexit=enabled
Turn on the highest level of tracing	mgr.*=debug=enabled
Turn on tracing for warnings	mgr.*=event=enabled

The Trace Output area shows the directory and file name where the trace information is written. A typical location is ${SERVER_LOG_ROOT}/trace.log, where the WebSphere environment variable ${SERVER_LOG_ROOT} represents the runtime log directory, WAS_install_dir/logs/app_server_name.

f. Click **OK**.

g. In the Task bar, click **Save** to save the change to the master configuration.

h. In the **Save to Master** Configuration area, click **Save** again to confirm the change.

- Disabling tracing

  To disable tracing for the agent manager:

  a. Start the WebSphere Administrative Console.

  b. Click **Servers** → **Application Servers** → **app_server_name** → **Logging and Tracing** → **Diagnostic Trace**. Replace app_server_name with the name of the application server, where the agent manager applications are installed, for example, AgentManager.

  c. Ensure that **Enable Trace** is not checked.

  d. Click **OK**.

Recovering from an incomplete uninstallation of the agent manager

This section describes how to manually remove the agent manager from your environment if the uninstallation wizard does not complete successfully. Use this procedure only after you run the uninstall wizard. Some of these steps might not be necessary, depending on how much of the uninstallation wizard was completed.

- Stop the agent manager server if it is running. Run the following command:
  - On Windows systems: **`WAS_install_dir\bin\stopServer.bat AgentManager`**
  - On AIX, Linux, or Solaris systems: **`WAS_install_dir/bin/stopServer.sh AgentManager`**
- If the agent manager server is configured to start automatically after a Windows system restarts, delete the Windows service for the agent manager using the following command:
  `WAS_install_dir\bin\WASService.exe -remove "Tivoli Agent Manager"`
- If the agent manager server is configured to start automatically after an AIX, Linux, or Solaris system restarts, then remove the entries in the /etc/inittab file that start the server:
  - Remove the entry that starts the agent manager server. This entry is added on all systems.
    am:2345:once:/opt/IBM/WebSphere/AppServer/bin/rc.am >/dev/console

2>&1 In this example, WebSphere Application Server is installed in the /opt/IBM/WebSphere/AppServer directory.

– If the registry is in a local DB2 database, then remove the entry that starts the DB2 server. The entry is not created if the registry is in a remote DB2 database or a local or remote Oracle database:
amdb:2345:once:su - db2inst1 -c db2start >/dev/console 2>&1.

In this example, db2inst1 is the DB2 user name for accessing the registry.

- Remove the agent manager from the WebSphere Application Server:

  a. Open the WebSphere Administrative Console.

  b. In the navigation tree on the left side of the console, expand **Environment**, click **Virtual Hosts**, and then delete **AgentManagerHost**.

  c. Expand **Security**, click **SSL**, and then delete **AgentManagerSSL** and **AgentManagerClientAuthSSL**.

  d. Expand **Resources**, and click **JDBC Providers**. Using the filter table, set the scope to the **server AgentManager**, and then delete **AgentJDBCProvider**.

  e. Expand **Applications**, click **Enterprise Applications**, and then uninstall **AgentManager** and **AgentRecoveryService**.

  f. Expand **Servers**, click **Application Servers**, and then delete **AgentManager**.

  g. In the **Message(s)** area of the console, click **Save** to save the changes to the configuration.

  h. In the **Save to Master Configuration** area, click **Save**, again, to exit the Master Configuration window.

- Optionally, clean up the agent manager objects in the database.

Important: Remove the agent manager tables from the database, or drop the registry database only if all products that use the registry are uninstalled.

a. Remove the agent manager tables from the registry database by running the following command:

   - On Windows systems:

     `Agent_Manager_install_dir\db\database_type\RemoveCASTables.bat`

   - On AIX, Linux, and Solaris systems:

     `Agent_Manager_install_dir/db/database_type\RemoveCASTables.sh`
     Replace variable database_type with one of the following values to identify the database type: – db2 – oracle

b. If the database is used only by the agent manager and is not shared with another program, optionally drop the registry database using the database administration tools for your type of database.

c. Delete the agent manager installation directory, which is by default the following directories:

   - On Windows systems: C:\Program Files\IBM\AgentManager
   - On AIX, Linux, and Solaris systems: /opt/IBM/AgentManager

   Note: On a Windows system, you might have to restart the system before you can delete the agent manager installation directory.

Logging level

The new CAS logging for the agent is based on ICL and uses the standard Java logging facility (JSR-47). To enable the right logging level, edit the following file:

```
...\ep\runtime\base\rcp\plugin_customization.ini
```

Add a line for the package or class that you want to do logging for and the right logging level.

If you want all messages for all packages or classes, then enter the line `.level=ALL`. Alternately, if you want warning messages, enter the line `.level=WARNING`.

Modify <install_dir>/runtime/base/rcp/plugin_customization.ini, to stop and start the agent.

Example 11-8 shows how to set the logging level to the highest level.

Example 11-8 Setting the logging level to the highest level

```
handlers=java.util.logging.ConsoleHandler
com.ibm.rcp.core.logger.boot.RCPFileHandler.level=ALL
com.ibm.rcp.core.logger.boot.RCPFileHandler.level=FINEST
com.ibm.pvc.wct.internal.logredirector.level=FINEST
```

```
java.util.logging.ConsoleHandler.level=FINEST
org.eclipse.update.core/updatePolicyURL=file:../../update.images/update
_policy.xml
```

Common agent

In the following section, we discuss how to troubleshoot the common agent.

The common agent default install directory depends on the operational system as follows:

- Windows: C:\Program Files\tivoli\ep
- AIX, HPUX: /usr/tivoli/ep
- Solaris, Linux: /opt/tivoli/ep

Determining the version of the currently installed TCA

To determine the version of the currently installed common agent, run the following command:

- On Windows:

  ```
  C:\Program Files\tivoli\ep\runtime\agent\endpoint version
  ```

- On AIX operating systems:

  ```
  /usr/tivoli/ep/runtime/agent/endpoint.sh version
  ```

- On other operating systems:

  ```
  /opt/tivoli/ep/runtime/agent/endpoint.sh version
  ```

The output of the command indicates the version, major release, minor release (modification level), and fix pack level of the common agent. It also includes a build identifier.

Example 11-9   Output of the command that determines the version

```
[root@elpaso][/usr/tivoli/ep/runtime/agent]-> ./endpoint.sh version
Common Agent signature file information
Version           : 1
Release           : 3
Modification      : 1
FixPack           : 6
Build Level        : 200609140622
```

Use the method in Example 11-9 to determine information about the agent itself. You can also determine the version of the agents from the inventory data.

Collecting diagnostic information about the common agent

The service command takes a snapshot of the diagnostic information about a common agent and the computer system where the common agent is running. It compresses the information into an archive that you can use for troubleshooting or to send to IBM Customer Support for analysis. The archive includes an HTML file that displays a compilation of the information gathered.

- Running the service command
  - On Windows:

    ```
    C:\Program Files\tivoli\ep\runtime\agent\service
    ```

  - On AIX operating systems:

    ```
    /usr/tivoli/ep/runtime/agent/service.sh
    ```

  - On other operating systems:

    ```
    /opt/tivoli/ep/runtime/agent/service.sh version
    ```

  The script creates an archive file named `CASservice.zip` file that resides in the same directory.

  If you have more than one common agent on a managed system, run the tool in the CA_HOME directory of the common agent for which you want to capture diagnostic information. If you plan to send multiple CASservice.zip files to IBM Customer Support, rename the files to give them unique names.

- Contents of the CASservice.zip file

  The CASservice.zip file contains a summary of key system information in the CASserviceInfo.html file, which you can view in a Web browser. The summary displays:

  - The configuration of the lightweight runtime where the common agent runs.
  - The common agent configuration.
  - Operating system and network configuration.
  - Java configuration.
  - A list of the files and directories in the TCA directory.
  - On Windows, the Microsoft system information that the Msinfo32.exe command displays. This information is in the msInfo.txt file.
  - The files and subdirectories in the cert, config, and logs directories.
  - The common agent software signature file, BTCversion.sys.
  - The runtime logs (rcp.log.n) in the LWI_HOME/logs or LWI_HOME/runtime/base/logs directory.

Common Agent health check

The common agent contacts the agent manager and reports its status and any configuration changes at these times:

- When a common agent starts or stops.
- After a configurable period of time. The default is 24 hours.
- Any time a bundle is installed, upgraded, or removed.

You can set the heartbeat frequency as desired. If the common agent status service does not find the property (status.heartbeat.frequency) in the configuration file, then it defaults to 1440 minutes (one day). Adding this property to the $AGENT_HOME/runtime/agent/config/endpoint.properties file and setting it to the new heartbeat changes it. You need to restart the agent for this change to take effect.

- Running the common agent discovery and health status report

  You can run a common agent discovery scan on the agent manager database to discover information about the common agent health on the target computers. You can output the common agent status information to a health status report.

  To run the common agent discovery scan:

  a. In the navigation pane, click **Inventory** → **Manage Discovery** → **Discovery Configurations**. The Manage Discovery Configurations page is displayed.

  b. In the list, find the IBM Tivoli Common Agent Discovery Configuration and, in that row, click **Run**. The Initiate Discovery page is displayed.

  c. Specify the Task Name or accept the default name.

  d. Under **Schedule**, select **Now** to run the discovery task immediately, or schedule it to run at a specified date and time. Select the appropriate notification options.

  e. Click **Submit**. The Track Tasks page is displayed.

  f. On the Track Tasks page, you can click the task name to see details about its status and to monitor its progress until the task has completed. Click **Refresh** to see an update on its status.

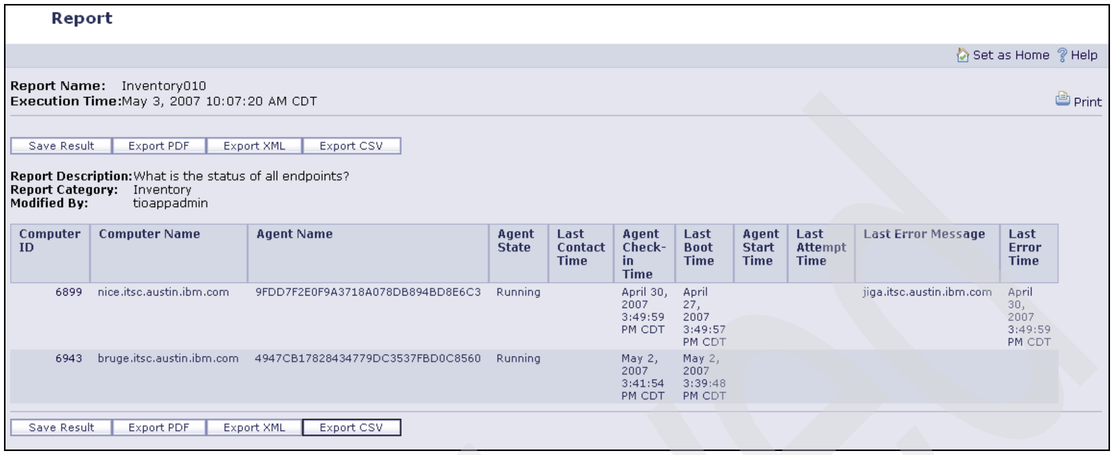
- Running the common agent health status report

  You can now run a health status report, to display the common agent status information that was generated during the common agent discovery scan.

  To generate a common agent health status report:

  a. In the navigation pane, click **Reports** → **Inventory**.

b. In the **Inventory010** row, click **Run**. The Run Report page is displayed, as shown in Figure 11-5.



Report Name: Inventory010
Execution Time: May 3, 2007 10:07:20 AM CDT

Report Description: What is the status of all endpoints?
Report Category: Inventory
Modified By: tioappadmin

Computer ID	Computer Name	Agent Name	Agent State	Last Contact Time	Agent Check-in Time	Last Boot Time	Agent Start Time	Last Attempt Time	Last Error Message	Last Error Time
6899	nice.itsc.austin.ibm.com	9FDD7F2E0F9A3718A078DB894BD8E6C3	Running		April 30, 2007 3:49:59 PM CDT	April 27, 2007 3:49:57 PM CDT			jiga.itsc.austin.ibm.com	April 30, 2007 3:49:59 PM CDT
6943	bruge.itsc.austin.ibm.com	4947CB17828434779DC3537FBD0C8560	Running		May 2, 2007 3:41:54 PM CDT	May 2, 2007 3:39:48 PM CDT				

Figure 11-5 Common Agent Health status report page

The generated report displays agent status parameters, such as the status, last contact time, last boot time, agent start time, last attempt time, and the last error message for each of the target computers, which enable you to assess the current health status of the common agent that is installed on the target computers. All target computers must be in good health. You can save the result, export pdf, export XML, and export CSV as well.

Return codes for the common agent installation

The installation program sets the return value to reflect the success or failure of the installation procedure and writes the return value to a log file located on the Agent Manager server, as follows:

- For installation, the return value is in the <install_dir>/runtime/agent/logs/install/epInstallStatus.log file.
- For uninstallation, return values are written to the <install_dir>/runtime/agent/logs/install/epUnInstallStatus.log file.

Table 11-14 on page 365 lists the logs created during the installation and uninstallation of the common agent.

Table 11-15 on page 365 lists the runtime logs for the common agent and subagents.

Table 11-14 Installation logs for the common agent

Log file located in LWI_HOME/runtime	Description
agent/logs/install/ epInstallStatus.log	Contains the return code for the installation of the common agent. For information about the contents of this log, see "Return codes for the common agent installation" on page 364".
agent/logs/install/ epUnInstallStatus.log	Contains the return code for the uninstallation of the common agent.
agent/logs/agentInstall.log	The complete installation log. The contents of other installation logs are duplicated in this file.
agent/logs/guid_install.log	Standard output and standard error from the installation of the Tivoli GUID.
agent/logs/lwiSetJavaHome_stdout.log	Standard output for setting the JAVA_HOME environment variables in the lwiSetJavaHome.bat and lwiSetJavaHome.sh commands.
agent/logs/lwiSetJavaHome_stderr.log	Standard error for setting the JAVA_HOME environment variables in the lwiSetJavaHome.bat and lwiSetJavaHome.sh commands.
agent/logs/nonstop.log	The log of the nonstop process.
agent/logs/nonstopbundle.log	On Windows, the log for the nonstop bundle.
base/logs/nonstopbundle.log	On operating systems other than Windows, the log for the nonstop bundle. Note: This file is in a different directory than other installation logs.
agent/logs/preinstall.log	The preconfiguration log for the installation.

Table 11-15 Runtime logs

Log file	Description
LWI_HOME/logs/rcp.log.0	The most current runtime log for the common agent. When the size of this log reaches 1.9 MB, it is renamed to rcp.log.1, and a new rcp.log.0 file is started.
LWI_HOME/runtime/base/logs/rcp.log.0	The runtime logs for subagents.

- Return values are in the following format for an installation:

```
#date_and_time InstallStatus=number
```

- For an uninstallation, the file has the following format:

```
#date_and_time UnInstallStatus=number
```

- The variable number is a signed integer value. For example, the following lines in the epInstallStatus.log file indicate a successful installation:

```
#Tue Oct 25 10:38:08 EDT 2005 InstallStatus=0
```

The following lines indicate that the installation failed:

```
#Tue Oct 25 10:38:08 EDT 2005 InstallStatus=-101
```

Table 11-16 Installation and uninstallation program return values

Value of install or unInstall status	**Meaning**
0	Successful installation or uninstallation.
-101	Installation failed. Read other log files for more information. For example, epInstall.log in the common agent installation directory and other log files in the logs/install directory.
-102	A newer version of the common agent exists in the location that you are trying to upgrade.
-103	User canceled the upgrade to the existing common agent v. The upgrade is canceled in silent mode if the parameter -W CASInstall.InstallType is set to install the value.
-104	User entered an incorrect password for the Windows service account that is associated with the common agent.
-108	Error during the Tivoli GUID installation.
-111	An incorrect JVM is used to invoke the installer.
-120	A file operation failure occurred. Possible reasons include: ► Cannot change file or directory permissions ► Cannot copy the certificate or truststore files
-121	A Windows or UNIX service cannot be created, started, or stopped.
-123	One or more of the following property files cannot be updated: ► nonstop.properties ► endpoint.properties

Value of install or unInstall status	Meaning
-124	One or more common agent response file options are not valid. This error applies only to a silent installation. In the wizard, errors are detected either when the user specifies the value or when they click Next to go to the next panel. In either case, the wizard gives an error message and the user gets to try again.
-125	The OEMInstall flag is set to true, but a common agent is already installed on the system. The OEMInstall flag can only be used to install a single common agent on a system.
-201	The uninstallation failed. Look at other log files for more information. For example, epUnInstall.log in the common agent root directory and other log files in the logs/install directory.
-202	You cannot uninstall the common agent because there are one or more product subagents on this common agent.

For example:

- When installing an agent on UNIX, if it fails while copying the file, or doing a put file, check if your /tmp dir (and device) is full on UNIX. The install agent copies the file to /tmp/tcatemp. A df -k /tmp can determine if the disk is full.
- The agent manager workflow TCA_PingAgent hangs on Windows computers, testing communication with the TCA_PingAgent workflow. A non-stop process may still be running because the previous uninstall was not completely clean. Check the `SystemErr.out` file on the Windows computer for an error message similar to the following error:

  ```
  com.tivoli.agent.system.SystemStreamMgr$LogPrintStream
  println(String) SEVERE: NOTE ==><4> Server socket failed to accept
  incoming connections.[java.io.IOException: SSL session object is
  null. Socket resource may not be available.]
  ```

  The message may indicate that the agent cannot open the port, which means that the port is already open by something else. Determine which type of error is causing the problem and follow the appropriate steps to resolve the error:

  – Access privileges are missing for the local administrator (`error -121`): The `-121` error is caused by lack of "log on as service" privilege for the local administrator account on the machine. If you are using an existing account for the Windows service, make sure that you add the following privileges, and ensure that the following policies are assigned to the administrator:

    • Act as a part of the operating system
    • Log on as service

On the Windows computer, navigate to **Control Panel** → **Administrative Tools** → **Local Security Policy** → **Local Policies** → **Local User Rights Assignment.**

– Ports specified for install are already in use (`error -124`):
To successfully install the common agent on target endpoints, you must ensure that ports 80 (WebContainerPort) and 443 (WebContainerSSLPort) are free on the endpoints. If you attempt the common agent installation on an endpoint that already uses ports 80 or 443, the common agent installation will fail.

  The -124 error is caused by two ports, which we are not setting; therefore, the ports are defaulted to WebContainerPort (defaults to 80) and WebContainerSSLPort (defaults to 443). If you install in a system that already has a process with a socket bound to these, or any of the other ports that we specify, the installation will fail with code -124.
  You can obtain information about this error by looking at the preinstall log file (C:\Program Files\tivoli\ep\runtime\agent\logs\preinstall.log). Check also if you have IIS running in the target.

- The agent installation error can also be "cannot ping agent", and on the target there is only one entry in the epinstallstatus.log InstallStatus=-104.
- If you are not on Tivoli Provisioning Manager for Software 5.1 Fix Pack 2, then check DNS properties.

  In order to deploy the agent, the target machine has to be registered with the DNS server.

  The Agent registration requires DNS on the Tivoli Provisioning Manager server because the endpoint.properties file uses the fully qualified domain name of the Tivoli Provisioning Manager server. You can also add to the hosts file the IP and host name for the Agent Manager server on the target to get around this, and when installing Tivoli Provisioning Manager there is a section that asks if you want to use the IP address for this by default.
  If you find the -101 error in the preinstall.log, an unsuccessful user action could be the cause. The error can occur when executing the InstallUtil tool, which is the tool that manipulates the system user. Launch this tool manually in the same way to reproduce the error. For example,
  `“C:\tcatemp\utility\InstallUtil.exe" -userexists -user rlshah returned 0`.

11.4 Tivoli Configuration Manager and Tivoli Provisioning Manager linking and resource exchange

Before you execute the tcmLink and tcmreplicate scripts, make sure your environment is Tivoli Provisioning Manager for Software 5.1 Enterprise installation. When you perform Tivoli Provisioning Manager for Software 5.1 FastStart installation, you cannot execute the replication since FastStart uses Cloudscape™ as its database.

11.4.1 Troubleshooting tcmLink

Every time a change that is related to Tivoli Configuration Manager's relational databases occurs, you should execute the `%TIO_HOME%\tools\`tcmLink script to keep the Tivoli Provisioning Manager database (DCM) up-to-date.

Make sure you follow all the requirements that we described in 5.3, "Replicating the Tivoli Management Framework data" on page 111, before you run the tcmLink script.

Note: If the script ends with errors, the database part of the bridge is not created. In this case you need to analyze the log files before attempting the script execution again.

To troubleshoot the tcmLink script execution, consult the `tcmLink_timestamp.log` file, which is created in the `%TIO_LOGS%` directory. The default value for `%TIO_LOGS%` is:

- On Windows, `C:\Program Files\ibm\tivoli\common\COP\logs`
- On UNIX machines, `/usr/ibm/tivoli/common/COP/logs`

The following section contains some common mistakes when the requirements are not matched.

Database applications still active error message

```
SQL1025N  The database manager was not stopped because databases are
still active. ERROR: Failed to create the inventory links, exiting. TCM
link failed.
```

This error message means that you are running the tcmLink script while the database applications toward the DCM (Data Center Model) are still active; therefore, Tivoli Provisioning Manager might still be running or you need to wait for the complete shutdown operation.

- Consult the active db2 database applications:
  - On Windows go to **Start → IBM DB2 Command Line Tools → Command Start → Command Window**. Perform the following command:

    ```
    db2 list applications
    ```

  - On a UNIX environment, use the database user ID and perform the following command to check active db2 applications:

    ```
    db2 list applications
    ```

  If some of the db2 applications are present, even after the Tivoli Provisioning Manager shutdown and all other attempts, then you might want to kill the db2 applications using the following command:

  ```
  db2 “force application (<Appl. Handle number>)”
  ```

  You should see the following message:

  ```
  DB20000I The FORCE APPLICATION command completed successfully.
  ```

  Double check the active database applications, where you should see no applications. Then, try the tcmLink execution again.

Wrong user or password set for database error message

If you see the following error message in the end of the tcmLink execution script, then you probably made a mistake in mapping Tivoli Management Framework on Tivoli Provisioning Manager environment:

```
SQL30082N  Attempt to establish connection failed with security reason
"24". ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001. ERROR:
Failed to create the inventory links, exiting. TCM link failed.
```

Make sure you entered the correct user and password when you mapped the Tivoli Management Framework and added databases. Both the user and password are case sensitive.

You can double-check the information about the inventory database using the combination of the following arguments:

- To check the database server, run the following command:
  `%TIO_HOME%\tools\tcmConfig.cmd dbsvr`
- To check the port information, run the following command:
  `%TIO_HOME%\tools\tcmConfig.cmd port`
- Locate the database detailed information by running the following command:
  `%TIO_HOME%\tools\tcmConfig.cmd db`

Different database vendors error

The tcmLink script federates all of the Tivoli Configuration Manager databases that are configured into the tcm.xml file. The script also creates a unified view of all those databases.

The supported links are: DB2 to DB2 and Oracle to Oracle. See the following example.

- A Tivoli Management Framework on AIX with Oracle RDBMS for Tivoli Configuration Manager 4.2.3 RIMs.
- Tivoli Provisioning Manager for Software 5.1 on AIX with DB2 database for Data Center Model (DCM).

If you try to link unlike database vendor types between Tivoli Configuration Manager and Tivoli Provisioning Manager environments, the operation will fail because it is not supported.

11.4.2 Troubleshooting tcmReplicate

Every time Tivoli Provisioning Manager database (DCM) need refreshed data from Tivoli Configuration Manager, you should execute the `%TIO_HOME%\tools\`tcmReplicate script.

To troubleshoot the tcmReplicate script execution, consult the `tcmReplicate_`timestamp`.log` file, which is created in the *%TIO_LOGS%* directory.

The default value for `%TIO_LOGS%` is:

- On Windows, `C:\Program Files\ibm\tivoli\common\COP\logs`
- On UNIX machines, `/usr/ibm/tivoli/common/COP/logs`

Credentials

If you are facing replication issues, make sure that your credentials are correct because the user and password are case sensitive from Tivoli Provisioning Manager to Tivoli Configuration Manager or Tivoli Management Region, even on Windows systems. Following is an example of a credentials error message when you run the tcmReplicate script:

```
Tivoli Provisioning Manager for Software COPTMF105E Replication Failed:
The user credentials provided for the IBM Tivoli Configuration Manager
server are not valid or do not exist.
```

Restart a new replica after the previous one crashed for any reason

When a replica ends abnormally, for some reason it could remain locked. At this point, if you try to run the replica again, it fails because the previous run of the operation created a lock in the DCM database that no one removed. We tell you how to remove this lock, both through the CLI and through a direct update of the DCM database, after the example.

Example 11-10 shows a tcmReplicate execution that was unexpectedly stopped followed by a new attempt:

Example 11-10 tcmReplicate execution crashed

```
C:\IBM\tivoli\tpmfsw\tools>tcmreplicate
2007-05-07 17:13:42,391 INFO  log4j configureAndWatch is started with
configuration file: C:\ibm\tivoli\tpmfsw/config/log4j-util.prop
2007-05-07 17:13:47,922 INFO  COPTMF002I Start Replication
2007-05-07 17:13:48,328 INFO  Object id1=6220 threadName=2007-05-07
17:13:48.328

2007-05-07 17:13:48,344 INFO  Replicating full environment
Terminate batch job (Y/N)? Y

C:\IBM\tivoli\tpmfsw\tools>tcmreplicate
2007-05-07 17:14:02,562 INFO  log4j configureAndWatch is started with
configurat
ion file: C:\ibm\tivoli\tpmfsw/config/log4j-util.prop
2007-05-07 17:14:08,156 INFO  COPTMF002I Start Replication
2007-05-07 17:14:08,562 INFO  Object id1=6220 threadName=2007-05-07
17:14:08.562

2007-05-07 17:14:08,578 INFO  Replication started at 2007-05-07
17:13:48.328 is
already in progress. Skipping this and all new requests untill the
replication i
n progress will be completed.
2007-05-07 17:14:08,578 INFO  Object id1=6220 threadName=Exit  full
environment
2007-05-07 17:14:08,578 INFO  Replication for thread name 2007-05-07
17:13:48.32
8 is not the one in progress. Not resetting
```

In a case like Example 11-10 on page 372, before a new attempt you must clean up the Tivoli Configuration Manager database according to the following procedure:

- Using the command line to clean up the Tivoli Configuration Manager database and unlock the replica:

  ```
  db2 connect to TC user <DCM_user> using <DCM_password>
  ```

  a. Run the following command to un-lock:

     ```
     delete from db2admin.properties where key_name =
     'REPLICATION_IN_PROGRESS'
     ```

     For huge environments, we suggest the following DB2 settings to avoid deadlocks:

     ```
     db2set DB2_SKIPINSERTED=YES
     db2set DB2_SKIPDELETED=YES
     db2set DB2_EVALUNCOMMITTED=YES_DEFERISCANFETCH
     ```

  b. Run the following DB2 command after the replica is performed:

     ```
     db2 reorgchk update statistics on table all
     ```

  When a deadlock occurs, it may be a lock time out or a true deadlock. Tivoli Provisioning Manager exception output marks them both as database deadlock exceptions, so in order to distinguish between these two lock situations you may use the SQLERRMC value.

  – SQLERRMC of 68 indicates a lock timeout

  – SQLERRMC of 2 indicates a deadlock

- Using GUI to clean up the Tivoli Configuration Manager database and unlock the replica:

  If the replication is run from the discovery configuration from the GUI, in the TCM_ROOT group (see Figure 11-6) that is under the variables tab, a new key is created when the replica is in progress, and it is called REPLICATION_IN_PROGRESS.

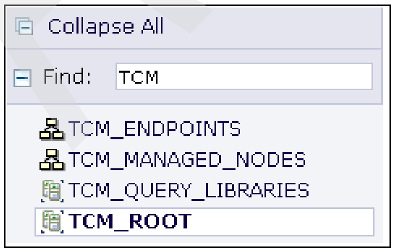
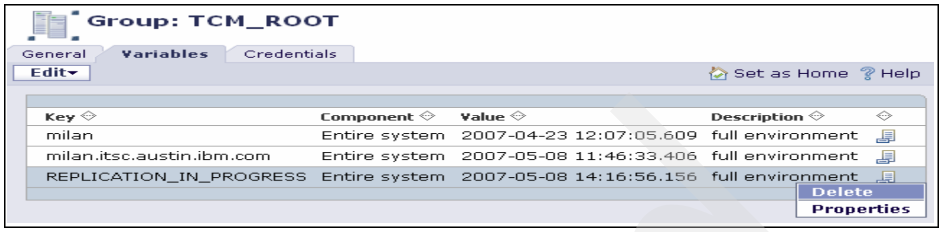


Figure 11-6 TCM_ROOT group under ‘Find’ on the left menu

You can delete the REPLICATION_IN_PROGRESS variable (see Figure 11-7), and then run the tcmReplicate script again.



Key	Component	Value	Description
milan	Entire system	2007-04-23 12:07:05.609	full environment
milan.itsc.austin.ibm.com	Entire system	2007-05-08 11:46:33.406	full environment
REPLICATION_IN_PROGRESS	Entire system	2007-05-08 14:16:56.156	full environment

Figure 11-7 REPLICATION_IN_PROGRESS variable under TCM_ROOT

- Using the tcmConfig script to clean up the Tivoli Configuration Manager database and unlock the replica:

  You can run the **`%TIO_HOME%\tools\tcmConfig.cmd reset`** command to reset the replication locks.

> **Note:** The script `%TIO_HOME%\tools\tcmConfig.cmd` shows information about the tcm.xml, which is used by the tcmLink and tcmReplicate to read the tcm.xml file and to validate the information contained there.

11.4.3 Report Manager

The Report Manager component provides a command-line interface for configuring trace information: **`wrpmgcfg`**.

Follow the usage of the command:

```
wrpmgcfg -s key[=value]
```

Following is an explanation of the valid keys for **`wrpmgcfg -s key[=value]`**:

- trace_level [0..6]
- trace_size [100..2000000000]
- trace_files [>0]

The Report Manager component stores trace files, as well as temporary files, for unprocessed reports in the follow folders:

- *%BINDIR%*\..\ReportManager for Windows

  $BINDIR/../ReportManager for UNIX
  These directories contain trace files.
- *%BINDIR%*\..\ReportManager\work for Windows

  $BINDIR/../ReportManager/work for UNIX
  These directories contain temporary files that the consumer thread uses when processing reports.

Example 11-11 shows some sample traces coming from a standard processing.

Example 11-11 Report Manager trace sample of a standard process

```
When the operation completes the receive_report method is invoked: it
stores the report on a temporary file with .msg extension

[944:18574632] 10/9/2006 19:36:01.078 [I] t_receive_report >>>> ENTRY
>>>>>
[944:18574632] 10/9/2006 19:36:01.078 [I] make_name_file attempting to
create D:\IBM\Tivoli\bin\ReportManager\work\rm1160415361078
[944:18574632] 10/9/2006 19:36:01.078 [I] t_receive_report save the
message
[944:18574632] 10/9/2006 19:36:01.093 [I] t_receive_report rename the
file

The consumer thread handles such a file and stores its contents in the
database:

[944:18590968] 10/9/2006 19:36:01.109 [I] t_consumer_report >>>> ENTRY
>>>>>
[944:18590968] 10/9/2006 19:36:01.109 [I] listFiles >>>> ENTRY >>>>>
[944:18590968] 10/9/2006 19:36:01.109 [I] file::list directory path is
D:\IBM\Tivoli\bin\ReportManager\work\*.msg
[944:18590968] 10/9/2006 19:36:01.109 [I] t_consumer_report analize the
file 'D:\IBM\Tivoli\bin\ReportManager\work\rm1160415361078.msg'
[944:18590968] 10/9/2006 19:36:01.109 [I] t_consumer_report read data
[944:18590968] 10/9/2006 19:36:01.109 [I] update_the_db >>>> ENTRY
>>>>>
[944:18590968] 10/9/2006 19:36:01.109 [I] update_the_db targets num: 1
[944:18590968] 10/9/2006 19:36:01.109 [I] update_the_db SERVER_ID: 2293
 DIST_ID: 1098543693.18
 STATUS: 6
 TARGET oid: 1098543693.8.522+#TMF_Endpoint::Endpoint#
 MESSAGE: DISSE0001I Operation successful.
 INST_ID: 3242
 APP_DATA: SoftwareModule.Install
[944:18590968] 10/9/2006 19:36:01.109 [I] RepDB:update_db >>>> ENTRY
>>>>>
[944:18590968] 10/9/2006 19:36:01.109 [I] RepDB:update_db query:
<insert into REP_MANAGER (SERVER_ID , DIST_ID , STATUS , TARGET ,
RECORD_TIME , MESSAGES , INST_ID , APP_DATA , REGION_ID) values
('2293', '1098543693.18', '6',
'1098543693.8.522+#TMF_Endpoint::Endpoint#', '20061009173601109',
'DISSE0001I Operation successful.
', '3242', 'SoftwareModule.Install', '1098543693') >
```

When the report has been processed, the temporary file is removed from the working directory.

```
[944:18590968] 10/9/2006 19:36:01.109[I] file::remove >>>> ENTRY >>>>>
[944:18590968] 10/9/2006 19:36:01.109[I] file::remove return data = 1
[944:18590968] 10/9/2006 19:36:01.109[I] file::remove <<<<< EXIT <<<<<
[944:18590968] 10/9/2006 19:36:01.109[I] t_consumer_report <<<<< EXIT
<<<<<
```

TPMfS reporting thread invokes the query_report method to get results of pending tasks specifying a start date

```
[944:18569120] 10/9/2006 20:07:43.953 [I] t_query_report >>>> ENTRY
>>>>>
[944:18569120] 10/9/2006 20:07:43.953 [I] t_query_report list=011AFD88
```

When the query_report method starts, it first trigger the consumer thread to see if there is any pending report in the working area that needs to be processed.

```
[944:18569120] 10/9/2006 20:07:43.953[I] run_consumer_thread >>>> ENTRY
>>>>>
[944:18569120] 10/9/2006 20:07:43.953[I] run_consumer_thread <<<<< EXIT
<<<<<
```

According to the specified configuration, the report manager delets obsolete rows in the database:

```
[944:18569120] 10/9/2006 20:07:43.953 [I] t_query_report del_old_rows=1
[944:18569120] 10/9/2006 20:07:43.953 [I] RepDB:del_old_rows >>>> ENTRY
>>>>>
```

```
[944:18569120] 10/9/2006 20:07:43.953 [I] RepDB:del_old_rows query:
<delete from REP_MANAGER where RECORD_TIME < '20061009180018251' AND
SERVER_ID = '2293' AND APP_DATA LIKE 'SoftwareModule%' AND REGION_ID =
'1098543693' >
```

It then executes the required query returning the available reports to the caller:

```
[944:18569120] 10/9/2006 20:07:43.984 [I] t_query_report execute
query_db
[944:18569120] 10/9/2006 20:07:43.984 [I] RepDB:query_db query=<SELECT
* FROM REP_MANAGER WHERE RECORD_TIME >= '20061009180018251' AND
REGION_ID = '1098543693' AND SERVER_ID = '2293' AND APP_DATA LIKE
'SoftwareModule%' AND ( STATUS = 6 OR STATUS = 7) ORDER BY RECORD_TIME>
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db DIST_ID=
1098543693.20
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db STATUS= 6
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db TARGET=
1098543693.8.522+#TMF_Endpoint::Endpoint#
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db RECORD_TIME=
18562904
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db MESSAGES=
DISSE0001I Operation successful.
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db INST_ID= 3242
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db APP_DATA=
SoftwareModule.Install
[944:18569120] 10/9/2006 20:07:44.046 [I] RepDB:query_db <<<<< EXIT
<<<<<
[944:18569120] 10/9/2006 20:07:44.046 [I] t_query_report <<<<< EXIT
<<<<<
```

On the TPMfS side, the activityplan console.log register the activity of the Status Updater thread which, for TMF actions, interacts with the Report Manager.

```
2006-10-09 20:07:43,875 DEBUG [Status Updater]
(TmfStatusUpdater.java:44) tmf.TmfStatusUpdater: Thread[Status
Updater,5,main] thread calling ProcessJobStatus on TMF
```

It starts processing the Software related reports

```
2006-10-09 20:07:43,906 DEBUG [Status Updater] (Connector.java:257)
proxy.Connector: current TMR server OID:1098543693.1.0
2006-10-09 20:07:43,938 DEBUG [Status Updater] (TmfService.java:93)
swd.TmfService: processing software related results
```

```
2006-10-09 20:07:43,938 DEBUG [Status Updater] (TMFOperation.java:112)
tmf.TMFOperation: Invoking the getDistributionStatus with start_time =
20061009180018251
2006-10-09 20:07:44,047 DEBUG [Status Updater] (TMFOperation.java:117)
tmf.TMFOperation: building report
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TMFOperation.java:140)
tmf.TMFOperation: managing work item status
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TmfService.java:551)
swd.TmfService: MDist Status Count=1
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TmfService.java:128)
swd.TmfService: singleDist=false
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TmfService.java:129)
swd.TmfService: distId=null
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TmfService.java:138)
swd.TmfService: loop on targets
2006-10-09 20:07:44,062 DEBUG [Status Updater] (TmfService.java:149)
swd.TmfService: returnDate= '20061009180742312' distId= '1098543693.20'
2006-10-09 20:07:44,109 DEBUG [Status Updater] (TmfService.java:161)
swd.TmfService: status=SUCCEEDED
2006-10-09 20:07:44,203 DEBUG [Status Updater] (TmfService.java:166)
swd.TmfService: gino#gbernard-tp-region
2006-10-09 20:07:44,203 DEBUG [Status Updater] (TmfService.java:170)
swd.TmfService: tm.getTaskId()='16913', tm.getTaskJobItemId()='4504',
ep.getId()='8218'
2006-10-09 20:07:44,281 DEBUG [Status Updater] (TmfService.java:226)
swd.TmfService: loop completed
2006-10-09 20:07:44,297 DEBUG [Status Updater] (TmfService.java:229)
swd.TmfService: softwareTasks.size = 1
2006-10-09 20:07:44,297 DEBUG [Status Updater] (TmfService.java:247)
swd.TmfService: returnDate = '20061009180742312'
2006-10-09 20:07:44,297 DEBUG [Status Updater] (TmfService.java:557)
swd.TmfService: MDist Total Count=1
2006-10-09 20:07:44,297 DEBUG [Status Updater] (TmfService.java:105)
swd.TmfService: done processing software related results

It then processes Discovery related reports spawning a new thread for
this task, since it might require running a selective replica

2006-10-09 20:07:44,312 DEBUG [Status Updater] (TmfService.java:109)
swd.TmfService: discovery thread is not alive... let's start it
2006-10-09 20:07:44,406 DEBUG [DiscoveryStatusUpdater]
(Connector.java:257) proxy.Connector: current TMR server
OID:1098543693.1.0
```

```
2006-10-09 20:07:44,453 DEBUG [DiscoveryStatusUpdater]
(TMFOperation.java:112) tmf.TMFOperation: Invoking the
getDistributionStatus with start_time = 20061006154835704
2006-10-09 20:07:44,734 DEBUG [DiscoveryStatusUpdater]
(DiscoveryResultsProcessor.java:47) swd.TmfService: MDist Status
Count=0
```

Checking Report Manager health

Perform the following actions to verify that the Report Manager is properly working:

- `wlookup ReportManager` searches the Tivoli name registry for object information about the Report Manager.

  ```
  1098543693.1.1151#RMEngine::ReportManager
  ```

- `objcall $TMR.1.1151 contents` lists the attributes of the Report Manager.

  ```
  ATTRIBUTE:_BOA_id
  ATTRIBUTE:class_objid
  ATTRIBUTE:collections
  ATTRIBUTE:consumer_sleep_timeout
  ATTRIBUTE:delete_old_rows
  ATTRIBUTE:label
  ATTRIBUTE:max_trans
  ATTRIBUTE:pres_object
  ATTRIBUTE:pro
  ATTRIBUTE:pro_name
  ATTRIBUTE:resource_host
  ATTRIBUTE:skeleton
  ATTRIBUTE:sort_name
  ATTRIBUTE:state
  ATTRIBUTE:trace_files
  ATTRIBUTE:trace_level
  ATTRIBUTE:trace_size
  ATTRIBUTE:trans_timeout
  ```

- `odstat |grep ReportManager` verifies the behavior of the attributes.

  Consult *Tivoli Management Framework Reference Guide, Version 4.1.1,* SC32-0806, for further details about the **`odstat`** command. If you find errors or an error id status as **"**`e=12`**"** for any of those attributes, then the faster way to get it fixed is to reinstall the Report Manager as follows:

  a. `cd $BINDIR/TME/ReportManager`
  b. `./rm_uninst.sh`
  c. `./rm_install.sh`
  d. `wchkdb -ux`

- Perform the Tivoli Configuration Manager replication procedure, as described in 5.3, “Replicating the Tivoli Management Framework data” on page 111, because Tivoli Provisioning Manager for Software needs to identify the new Report Manager component.
- `wchkdb -ux`
- After reinstalling the Report Manager, consult, one more time, its attributes behavior in odstat. If you see the old BAD_ID still registered, as shown in Example 11-12, we recommend that you restart the oserv using the **`odadmin`** command.

Example 11-12 Report Manager BAD_ID after reinstallation

```
4843 0           done     0     0 12:03:20 BAD_ID
1536006190.1.1117#RMEngine::ReportManager# query_report
4844 O+          done    15     0 12:04:06        0.0.0 get_oserv
```

11.5 Putting it all together

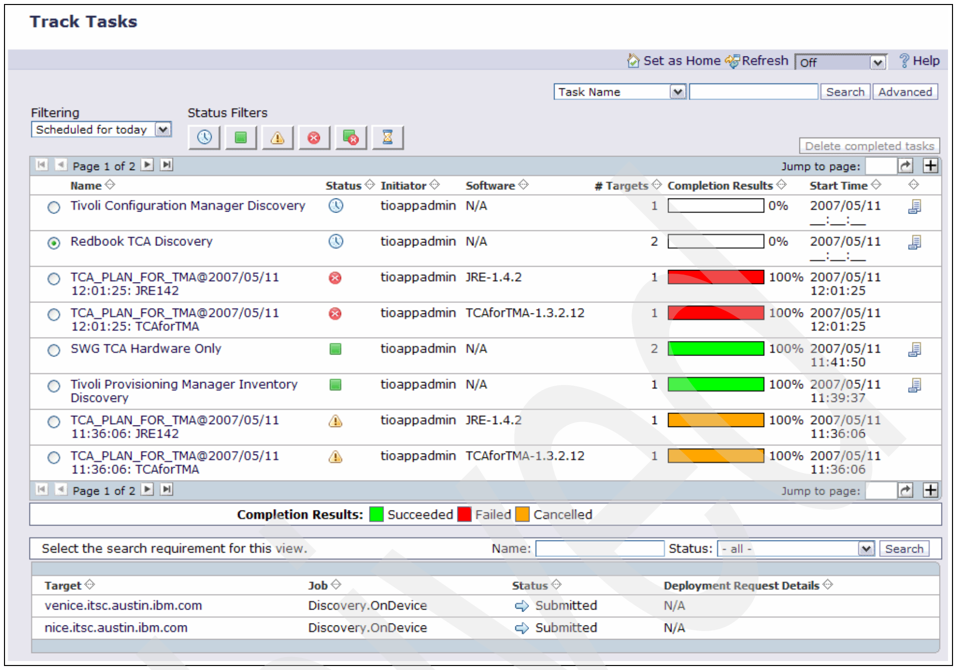
In this section we provide some real-life troubleshooting scenarios.

11.5.1 Discovery

This section discusses how to track a discovery that is initiated to a common agent in the SOA architecture. We go through each step and monitor a discovery.

Starting the discovery

In Figure 11-8 on page 381, we started a discovery called “Redbook TCA Discovery”. This discovery is running a hardware and software scan against the common agents we selected, in this case, nice.itsc.austin.ibm.com and venice.itsc.austin.ibm.com.



Track Tasks

Set as Home Refresh Off Help

Task Name | Search | Advanced

Filtering: Scheduled for today

Status Filters

Delete completed tasks

Page 1 of 2 — Jump to page:

	Name	Status	Initiator	Software	# Targets	Completion Results	Start Time	
○	Tivoli Configuration Manager Discovery	🕒	tioappadmin	N/A	1	0%	2007/05/11 __:__:__	
●	Redbook TCA Discovery	🕒	tioappadmin	N/A	2	0%	2007/05/11 __:__:__	
○	TCA_PLAN_FOR_TMA@2007/05/11 12:01:25: JRE142	⊗	tioappadmin	JRE-1.4.2	1	🟥 100%	2007/05/11 12:01:25	
○	TCA_PLAN_FOR_TMA@2007/05/11 12:01:25: TCAforTMA	⊗	tioappadmin	TCAforTMA-1.3.2.12	1	🟥 100%	2007/05/11 12:01:25	
○	SWG TCA Hardware Only	🟩	tioappadmin	N/A	2	🟩 100%	2007/05/11 11:41:50	
○	Tivoli Provisioning Manager Inventory Discovery	🟩	tioappadmin	N/A	1	🟩 100%	2007/05/11 11:39:37	
○	TCA_PLAN_FOR_TMA@2007/05/11 11:36:06: JRE142	⚠	tioappadmin	JRE-1.4.2	1	🟧 100%	2007/05/11 11:36:06	
○	TCA_PLAN_FOR_TMA@2007/05/11 11:36:06: TCAforTMA	⚠	tioappadmin	TCAforTMA-1.3.2.12	1	🟧 100%	2007/05/11 11:36:06	

Page 1 of 2 — Jump to page:

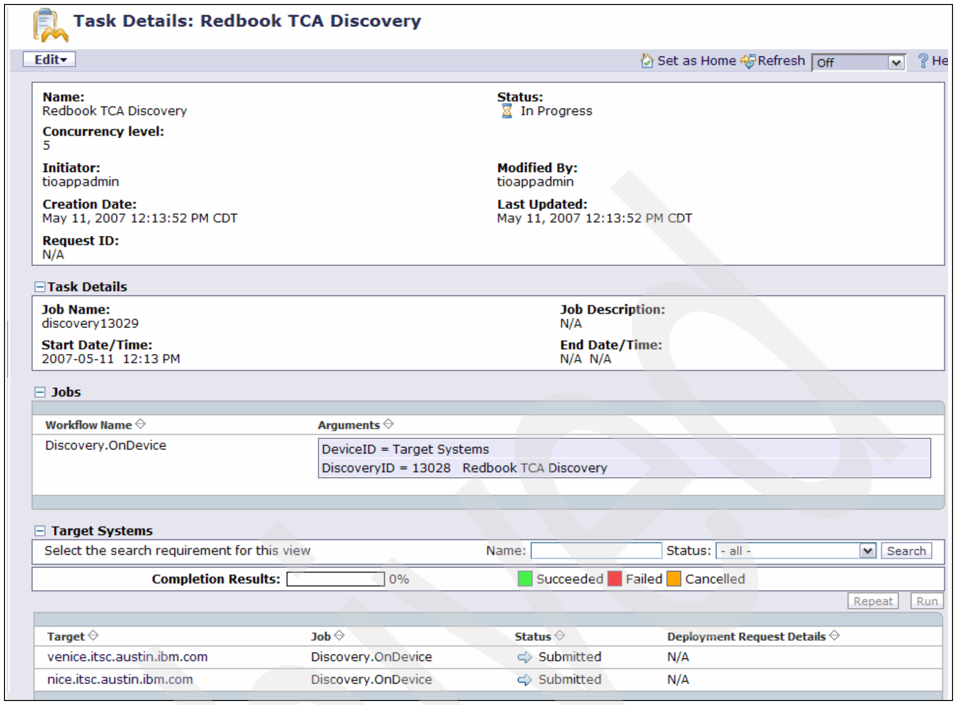
Completion Results: 🟩 Succeeded 🟥 Failed 🟧 Cancelled

Select the search requirement for this view. Name: Status: - all - Search

Target	Job	Status	Deployment Request Details
venice.itsc.austin.ibm.com	Discovery.OnDevice	Submitted	N/A
nice.itsc.austin.ibm.com	Discovery.OnDevice	Submitted	N/A

Figure 11-8 A discovery is started

To see more detailed information, we can click the label **Redbook TCA Discovery**, and we are presented with the page in Figure 11-9 on page 382.


Task Details: Redbook TCA Discovery

Name:
Redbook TCA Discovery
Status:
In Progress
Concurrency level:
5
Initiator:
tioappadmin
Modified By:
tioappadmin
Creation Date:
May 11, 2007 12:13:52 PM CDT
Last Updated:
May 11, 2007 12:13:52 PM CDT
Request ID:
N/A

Task Details

Job Name:
discovery13029
Job Description:
N/A
Start Date/Time:
2007-05-11 12:13 PM
End Date/Time:
N/A N/A

Jobs

Workflow Name	Arguments

Discovery.OnDevice	DeviceID = Target Systems
	DiscoveryID = 13028 Redbook TCA Discovery

Target Systems

Select the search requirement for this view Name: Status: - all -

Completion Results: 0% 🟩 Succeeded 🟥 Failed 🟧 Cancelled

Target	Job	Status	Deployment Request Details
venice.itsc.austin.ibm.com	Discovery.OnDevice	Submitted	N/A
nice.itsc.austin.ibm.com	Discovery.OnDevice	Submitted	N/A

Figure 11-9 Task Details: Redbook TCA Discovery

Now, even though we can see that the task is “In Progress” and that the jobs were “Submitted” for each TCA, we do not know the state of these jobs or have any visibility into what is going on in the background. Using the log files from <SECTION>, we can track what is going on.

Registering the Discovery

Using the task name, example “Redbook TCA Task”, search the $TIO_LOGS/activityplan/console.log using the command:

```
grep "Redbook TCA Discovery" $TIO_LOGS/activityplan/console.log
```

The first thing you will notice in the output is that we must execute the task in question. In the log file, the start of the task execution looks similar to Example 11-13 on page 383.

Example 11-13 activityplan/console.log

```
2007-05-11 12:13:53,482 DEBUG [main] (QueuedActivity.java:71)
activityplan.QueuedActivity: dequeuing Redbook TCA Discovery(3057)
2007-05-11 12:13:53,488 DEBUG [main] ( TpmTask.java:43) manager.TpmTask: Task
execution started: Redbook TCA Discovery
2007-05-11 12:13:53,533 DEBUG [main] ( TpmTask.java:75) manager.TpmTask: Task
execution finised: Redbook TCA Discovery
```

The task is now started, and from this point forward, each line contains [Redbook TCA Discovery(13030)], which matches the pattern discussed in <SECTION> : [taskname(taskid)].

Note: Depending on the type of task, you may want to perform your search on the *taskid* rather than the name. It is often useful to try both ways to make sure you do not miss any important log entries.

The next step, when a discovery starts, is to determine what targets the task will run against. In Example 11-14, we are running against a dynamic group query called *All TCAs*.

Example 11-14 activityplan/console.log

```
2007-05-11 12:13:53,554 DEBUG [Redbook TCA Discovery(13030)] (
Group.java:1457) groups.Group: Querying members of dynamic group named <<All
TCAs>>
2007-05-11 12:13:53,554 DEBUG [Redbook TCA Discovery(13030)] (
Group.java:1458) groups.Group: from clause  = <<endpoint_inv_view ,
server_info_view >>
2007-05-11 12:13:53,555 DEBUG [Redbook TCA Discovery(13030)] (
Group.java:1459) groups.Group: where clause = <<((endpoint_inv_view.SERVER_ID =
server_info_view.SERVER_ID) ) AND DcmObject.ID=endpoint_inv_view.SERVER_ID AND
DcmObject.TYPE_ID=3>>
2007-05-11 12:13:53,555 DEBUG [Redbook TCA Discovery(13030)] (
Group.java:1460) groups.Group: trunc value  = <<-1>>
```

After determining the targets, the TaskInfrastructureManager determines which protocols the targets support, and then the TaskDispatcher executes the tasks on the targets. There are three TaskDispatcher objects created, one for each possible type of target: a DE dispatcher, an SOA dispatcher, and a Tivoli Management Framework dispatcher.

Example 11-15 activityplan/console.log

```
2007-05-11 12:13:53,648 DEBUG [Redbook TCA Discovery(13030)]
(TaskInfrastructureManager.java:49) manager.TaskInfrastructureManager: Creating
task dispatcher for: deploymentengine.DETaskInfrastructure
2007-05-11 12:13:53,649 DEBUG [Redbook TCA Discovery(13030)]
(TaskInfrastructureManager.java:49) manager.TaskInfrastructureManager: Creating
task dispatcher for: com.ibm.tivoli.tpm.infrastructure.SOATaskInfrastructure
2007-05-11 12:13:53,649 DEBUG [Redbook TCA Discovery(13030)]
(TaskInfrastructureManager.java:49) manager.TaskInfrastructureManager: Creating
task dispatcher for:
com.ibm.tivoli.tpm.infrastructure.tmf.TMFTaskInfrastructure
2007-05-11 12:13:53,649 DEBUG [Redbook TCA Discovery(13030)]
(TaskInfrastructureManager.java:52) manager.TaskInfrastructureManager:
returning task dispatchers, size: 3
2007-05-11 12:13:53,650 DEBUG [Redbook TCA Discovery(13030)]
(AggregateTaskDispatcher.java:100) manager.AggregateTaskDispatcher: number of
dispatchers: 3
```

Now we examine the targets to determine which TaskDispatcher to use. In the log snippet in Example 11-16, there are no Tivoli Management Framework targets, but there are two SOA targets.

Example 11-16 activityplan/console.log

```
2007-05-11 12:13:53,659 DEBUG [Redbook TCA Discovery(13030)]
(TaskInstance.java:807) instance.TaskInstance: number of targets found for
protocol TMF: 0
2007-05-11 12:13:53,660 DEBUG [Redbook TCA Discovery(13030)]
(TaskInstance.java:807) instance.TaskInstance: number of targets found for
protocol SOA: 2
2007-05-11 12:13:53,672 DEBUG [Redbook TCA Discovery(13030)]
(TaskInstance.java:838) instance.TaskInstance: number of targets found: 2
2007-05-11 12:13:53,673 DEBUG [Redbook TCA Discovery(13030)]
(AggregateTaskDispatcher.java:56) manager.AggregateTaskDispatcher: Dispatching
2 targets to com.ibm.tivoli.tpm.infrastructure.SOATaskInfrastructure
2007-05-11 12:13:53,704 DEBUG [Redbook TCA Discovery(13030)]
(TaskInstance.java:838) instance.TaskInstance: number of targets found: 0
2007-05-11 12:13:53,712 DEBUG [Redbook TCA Discovery(13030)]
(SoaInfrastructure.java:81) soa.SoaInfrastructure:
SoaInfrastructure.executeTask entry
2007-05-11 12:13:53,713 DEBUG [Redbook TCA Discovery(13030)] (           ?:?)
infrastructure.TaskDispatcher: running task with targets: 2
```

Now that the targets are calculated and associated with a TaskDispatcher, the actual discovery can take place. The first line in the excerpt, in Example 11-17 on page 385, shows that we are starting on two targets.

Example 11-17 activityplan/console.log

```
2007-05-11 12:13:53,716 DEBUG [Redbook TCA Discovery(13030)]
(TaskDispatcher.java:66) infrastructure.TaskDispatcher: start looping through
job items with targets: 2
2007-05-11 12:13:53,719 DEBUG [Redbook TCA Discovery(13030)]
(DiscoveryTaskManager.java:80) discovery.DiscoveryTaskManager:
DiscoveryTaskManager.executeDiscovery entry
```

What the SOA TaskDispatcher actually does for a Discovery is run the Cit_SOA_OnDevice workflow.

Running the Discovery workflow

Example 11-18 shows that deployment engine is going to execute the workflow Cit_SOA_OnDevice.

Example 11-18 activityplan/console.log

```
2007-05-11 12:13:53,722 INFO  [Redbook TCA Discovery(13030)]
(MessageTranslatorProxy.java:289) messagetranslator.MessageTranslatorProxy:
createDeploymentRequest(workflowName=Cit_SOA_OnDevice)
2007-05-11 12:13:53,734 DEBUG [Redbook TCA Discovery(13030)]
(MessageTranslatorProxy.java:302) messagetranslator.MessageTranslatorProxy:
MessageTranslator.createDeploymentRequest() create empty request duration:12
2007-05-11 12:13:53,735 INFO  [Redbook TCA Discovery(13030)]
(MessageTranslatorProxy.java:305) messagetranslator.MessageTranslatorProxy: New
deployment request was created (id=12906)
```

The last line in Example 11-18 indicates that the deployment engine is executing the workflow, and that the request ID is 12906. From this point, all that is logged in the $TIO_LOGS/activityplan/console.log is the status update information for the request.

Note: To continue in our examination of the process through the log files, we must switch to the $TIO_LOGS/deploymentengine/console.log file.

Using the request ID from the activityplan/console.log file, 12906 in this example, search the $TIO_LOGS/deploymentengine/console.log using the following command:

```
grep 12906 $TIO_LOGS/deploymentengine/console.log
```

After the deployment engine receives the request, it executes the requested workflow. In Example 11-19 on page 386, the last line in the log excerpt shows that the workflow is being executed and has an ID of 3765.

Example 11-19 deploymentengine/console.log

```
007-05-11 12:13:53,773 INFO  [RequestDispatcher Thread]
(RequestDispatcher.java:183) engine.RequestDispatcher: Received deployment
request id=12906
2007-05-11 12:13:53,785 DEBUG [RequestDispatcher Thread]
(RequestDispatcher.java:271) engine.RequestDispatcher: Worker='Deployment
Request 12906', IsAlive=false
2007-05-11 12:13:53,786 DEBUG [Deployment Request 12906]
(DeploymentWorker.java:524) engine.DeploymentWorker: run() Begin
2007-05-11 12:13:53,787 INFO  [Deployment Request 12906]
(DeploymentWorker.java:548) engine.DeploymentWorker: Deployment request
started: Id=12906, at Fri May 11 12:13:53 CDT 2007
2007-05-11 12:13:53,793 INFO  [Deployment Request 12906]
(DeploymentWorker.java:254) engine.DeploymentWorker: Executing workflow:
Name='Cit_SOA_OnDevice', id=3765
```

When the workflow executes, it logs to the deploymentengine/console.log file, and we can continue to track the progress of the discovery with this file. The next major step for the workflow is to determine what Software Signature files are necessary for the targets and upload them to a depot. Example 11-20 shows the workflow making this step.

Example 11-20 deploymentengine/console.log

```
2007-05-11 12:13:57,633 DEBUG [Deployment Request 12906]
(ShellCommandHelper.java:347) util.ShellCommandHelper:
Command='/opt/ibm/tivoli/tpmfsw/bin/getPath.sh
"/opt/ibm/tivoli/tpmfsw/tmp/cit_subagent/13028-SoftwareSignatureForSOA"'

... <skipped lines> ...

2007-05-11 12:15:16,601 DEBUG [Deployment Request 12906] (FileManager.java:448)
cds.FileManager: File Name is 13028-SoftwareSignatureForSOA
2007-05-11 12:15:16,606 DEBUG [Deployment Request 12906] (FileManager.java:248)
cds.FileManager: HostName is venice.itsc.austin.ibm.com IP Address is 9.3.5.26
for device 6938
2007-05-11 12:15:16,606 DEBUG [Deployment Request 12906] (FileManager.java:248)
cds.FileManager: HostName is nice.itsc.austin.ibm.com IP Address is 9.3.5.88
for device 6899
2007-05-11 12:15:16,609 DEBUG [Deployment Request 12906] (FileManager.java:160)
cds.FileManager: URL is https://9.3.4.163:9045 username tioappadmin
2007-05-11 12:15:21,151 DEBUG [Deployment Request 12906] (FileManager.java:356)
cds.FileManager: PackageId for new file published is 1178903687125
2007-05-11 12:15:22,442 DEBUG [Deployment Request 12906] (FileManager.java:378)
cds.FileManager: Published file
/opt/ibm/tivoli/tpmfsw/tmp/cit_subagent/13028-SoftwareSignatureForSOA with
taskId 13030 to CDS depots: nice.itsc.austin.ibm.com
```

Example 11-20 on page 386 shows that the file 13028-SoftwareSignatureForSOA is published to the depot nice.itsc.austin.ibm.com with the available targets of nice.itsc.austin.ibm.com and venice.itsc.austin.ibm.com. You can also see, from the dynamic content delivery service server side, that the file was uploaded successfully. From the file /opt/ibm/tivoli/common/ctgde/logs/trace_cdsmgr.log, you can see where the file was uploaded.

Example 11-21   deploymentengine/console.log

```
2007-05-11 12:15:19.880-06:00 com.ibm.tivoli.cds.gui.servlets.AddPackage doPost
jiga.itsc.austin.ibm.com IP CTGDED044I The file 13028-SoftwareSignatureForSOA
was successfully registered with the management center. The file's unique id is
1178903687125 .
2007-05-11 12:15:19.887-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP CTGDED046I The operation to choose potential upload
servers so that client 9.3.4.163 can upload file with unique id 1178903687125
is starting.
2007-05-11 12:15:19.888-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP CTGDED047W No region was found for client
9.3.4.163. The operation to choose potential upload servers will try to
continue. Please update the ip ranges/domains to ensure optimal service.
2007-05-11 12:15:19.890-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP not enough upload servers were found in the
package's target lists, considering all depot servers
2007-05-11 12:15:21.121-06:00
com.ibm.webahead.downloadgrid.server.service.admin.implement.DownloadGridAdminI
mpl addDataUnit2Server      jiga.itsc.austin.ibm.com IP CTGDED007I File
1178903687125-1 was added to server nice.itsc.austin.ibm.com:2100 successfully.
```

Based on the way that the SOA architecture works, a job must be submitted to the device management service, so that when the targets check in they will need to perform the scan on themselves and upload their data. The last thing the workflow does is submit the device management service discovery job.

Example 11-22   deploymentengine/console.log

```
2007-05-11 12:15:23,903 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:307) client.JobManagementServiceClient: Using
endpoint URL:
https://jiga.itsc.austin.ibm.com:9045/dmserver/services/EDMSTPMManagementService
```

```
2007-05-11 12:15:23,910 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:562) client.JobManagementServiceClient:
Submitting JMS job
2007-05-11 12:15:23,915 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:563) client.JobManagementServiceClient:
descriptor =
2007-05-11 12:15:23,915 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:564) client.JobManagementServiceClient:
activationDate = 1178903723903
2007-05-11 12:15:23,915 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:565) client.JobManagementServiceClient:
expirationDate = 1180113323903
2007-05-11 12:15:23,915 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:566) client.JobManagementServiceClient:
priority = 3
2007-05-11 12:15:23,916 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:567) client.JobManagementServiceClient:
interval = 0
2007-05-11 12:15:23,916 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:568) client.JobManagementServiceClient:
interval unit = null
2007-05-11 12:15:23,916 DEBUG [Deployment Request 12906]
(JobManagementServiceClient.java:569) client.JobManagementServiceClient:
target[0] = 6938
```

At this point, you can also see where the job was submitted into the device management service server. The log file $TIO_HOME/tioprofile/logs/server1/TraceDMS1.log contains the job submission information. The few lines in Example 11-23, indicate the start of the job submission in the device management service.

Example 11-23 $TIO_HOME/tioprofile/logs/server1/TraceDMS1.log

```
05/11/2007 12:15:25.999 com.tivoli.dms.federator.FederatedJobManager createJob
Migrated.Servlet.Engine.Transports : 1331
  Job Object len in 9498

05/11/2007 12:15:25.999 com.tivoli.dms.federator.FederatedJobManager createJob
Migrated.Servlet.Engine.Transports : 1331
  Create Federator job -

05/11/2007 12:15:26.000 com.tivoli.dms.common.DBOperation executeUpdate
Migrated.Servlet.Engine.Transports : 1331
  INSERT INTO SUBMITTED_JOB (TARGET_DEVCLASS_ID, TARGET_DEVICE_SCOPE,
INSERTING, SEND_NOTIFICATION, JOB_ID, EXPIRATION_TIME, JOB_PRIORITY,
ACTIVATION_TIME, ALL_DEVICES, JOB_TYPE, ENROLLMENT_JOB) VALUES
(117736335400876302, 'BOTH', 'T', 'F', 117890372600017789, {ts '2007-05-25
17:15:23'}, 3, {ts '2007-05-11 17:15:25'}, 'T', 'FEDERATOR', 'T')
```

Note: It can be very hard to find the relevant job information when you are searching through the TraceDMS*.log. The best technique is to do the following:

1. Determine the time that the job was submitted from the deploymentengine/console.log line:

   ```
   2007-05-11 12:15:23,910 DEBUG [Deployment Request 12906]
   (JobManagementServiceClient.java:562) client.JobManagementServiceClient:
   Submitting JMS job
   ```

2. Search the TraceDMS*.log files for the string “createJob”:

   ```
   grep createJob TraceDMS*.log
   ```

3. Choose the createJob line that has the closest time to the line in step 1:

   ```
   05/11/2007 12:15:25.999 com.tivoli.dms.federator.FederatedJobManager
   createJob Migrated.Servlet.Engine.Transports : 1331
   ```

4. You can use the last number on the line in step 3, `1331,` to find all related information in the TraceDMS*.log:

   ```
   grep 1331 TraceDMS*.log
   ```

   However this will lose much of the activity content so it is best to search in your preferred editor to track job activity for device management service.

In Example 11-24, we can return to the deploymentengine/console.log to see that the deployment request, 12906, finished successfully.

Example 11-24 deploymentengine/console.log

```
2007-05-11 12:15:26,135 INFO  [Deployment Request 12906]
(DeploymentWorker.java:679) engine.DeploymentWorker: Deployment request
succeded: Id=12906, at Fri May 11 12:15:26 CDT 2007
```

It is also possible to see the workflow status using the user interface. Using the link **Automation** → **Workflow Status**, you can see all of the workflows that recently executed and their current state, as Figure 11-10 shows.

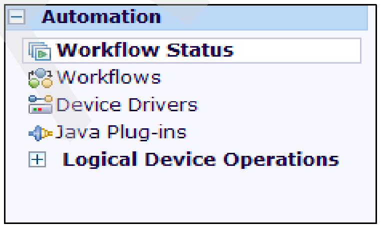
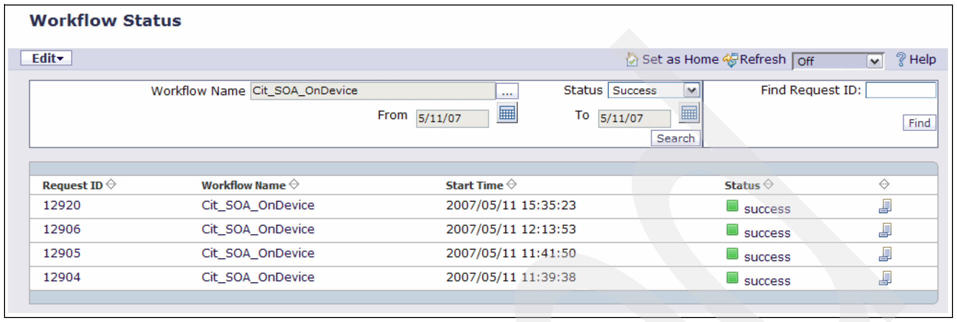


Figure 11-10 Automation Menu

Accessing this status page during a discovery can show you if the workflow completed successfully or not. It also gives you access to see the actual workflow code in case there are any problems. Figure 11-11 shows that the workflow Cit_SOA_Device completed successfully.



Workflow Name: Cit_SOA_OnDevice
Status: Success
Find Request ID:
From: 5/11/07
To: 5/11/07

Request ID	Workflow Name	Start Time	Status
12920	Cit_SOA_OnDevice	2007/05/11 15:35:23	🟩 success
12906	Cit_SOA_OnDevice	2007/05/11 12:13:53	🟩 success
12905	Cit_SOA_OnDevice	2007/05/11 11:41:50	🟩 success
12904	Cit_SOA_OnDevice	2007/05/11 11:39:38	🟩 success

Figure 11-11 Workflow status

Updating the status

We must return to the $TIO_LOGS/activityplan/console.log to continue tracking the status of the discovery task. Example 11-25 shows that while the deployment request is running, the activityplan/console.log continues to log that it is waiting for results. It also updates the status of the task to PROGRESS from NOTSTARTED and updates the plan to IN_PROGRESS from SUBMITTED. Example 11-25 also shows that the DeploymentRequest ID, 12096, finished successfully.

Example 11-25 deploymentengine/console.log

```
2007-05-11 12:13:53,755 DEBUG [Redbook TCA Discovery(13030)]
(DeploymentResultWaiter.java:68) engine.DeploymentResultWaiter: Waiting for
request id=12906
2007-05-11 12:15:14,278 DEBUG [Status Updater] (           ?:?)
manager.StatusUpdateProcessor: updating task Redbook TCA Discovery from
NOTSTARTED to PROGRESS
2007-05-11 12:15:15,151 DEBUG [Status Updater] (           ?:?)
manager.StatusUpdateProcessor: Updating plan instance status of: Redbook TCA
Discovery(3052) from SUBMITTED to: IN_PROGRESS
2007-05-11 12:15:26,791 DEBUG [Redbook TCA Discovery(13030)]
(DeploymentResultWaiter.java:78) engine.DeploymentResultWaiter:
DeploymentRequest id=12906 finished (DB check) in 93035 milliseconds.
```

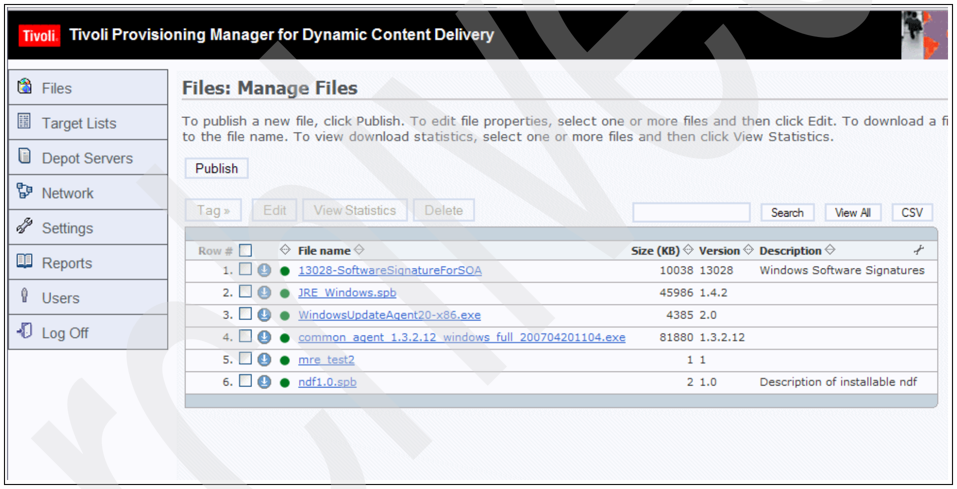
Monitoring the discovery

The user interface now shows that the discovery is in progress. We now have a Signature file that was uploaded to a dynamic content delivery service depot and a job submitted to device management service. However, nothing is going to happen until the common agent connects into the device management service. After the common agent is in progress, you monitor when the agent connects and when the file is downloaded, which you can observe in several locations, but most easily on the dynamic content delivery service server. To do this, go to:

```
https://<TPM Server>:9045/admin
```

From the File Manager window, you can see that the file was published. Figure 11-12 shows that 13028-SoftwareSignatureForSOA is now available on the depot.

> **Note:** If you do not see your signature file here, then most likely the task completed. After the task completes, the file is removed from the dynamic content delivery service depot and the file manager registry.



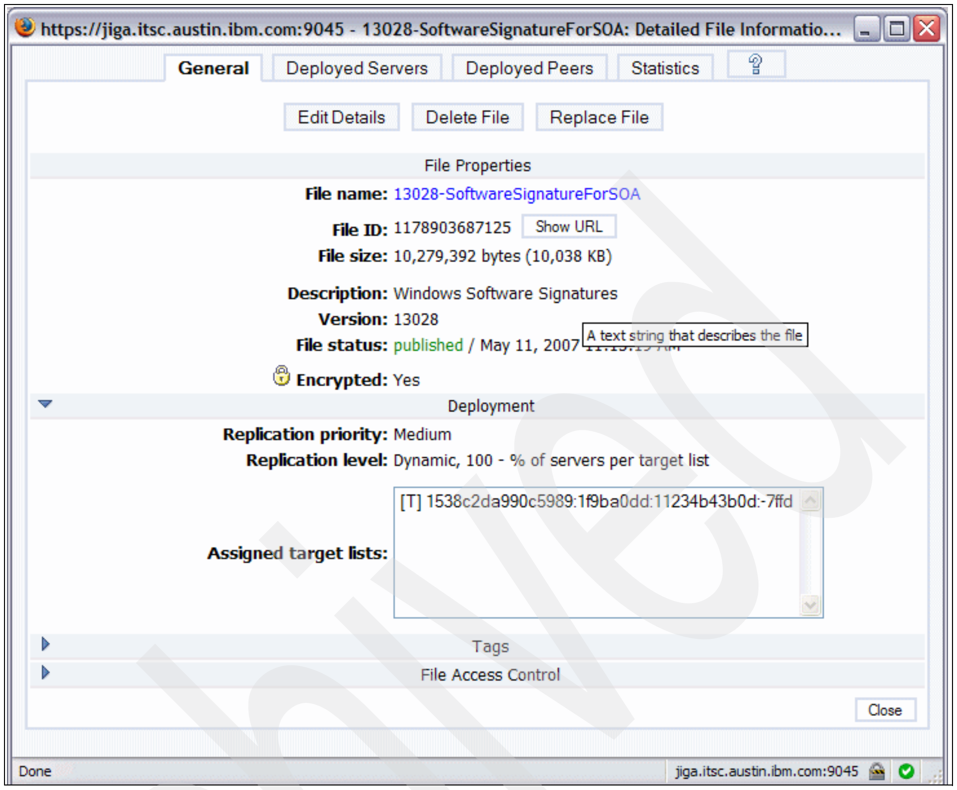
Files: Manage Files

To publish a new file, click Publish. To edit file properties, select one or more files and then click Edit. To download a fi to the file name. To view download statistics, select one or more files and then click View Statistics.

Row #	File name	Size (KB)	Version	Description
1.	13028-SoftwareSignatureForSOA	10038	13028	Windows Software Signatures
2.	JRE_Windows.spb	45986	1.4.2	
3.	WindowsUpdateAgent20-x86.exe	4385	2.0	
4.	common_agent_1.3.2.12_windows_full_200704201104.exe	81880	1.3.2.12	
5.	mre_test2	1	1	
6.	ndf1.0.spb	2	1.0	Description of installable ndf

Figure 11-12 Dynamic content delivery service file manager

From the panel in Figure 11-12, you can click the link for the file 13028-SoftwareSignatureForSOA. This brings up the details panel where you can view detailed information and statistics about the file. Figure 11-13 on page 392 shows the details panel and that the state of the file is PUBLISHED.



https://jiga.itsc.austin.ibm.com:9045 - 13028-SoftwareSignatureForSOA: Detailed File Informatio...

File Properties

File name: 13028-SoftwareSignatureForSOA

File ID: 1178903687125

File size: 10,279,392 bytes (10,038 KB)

Description: Windows Software Signatures

Version: 13028

File status: published / May 11, 2007 11:15:19 AM

A text string that describes the file

Encrypted: Yes

Deployment

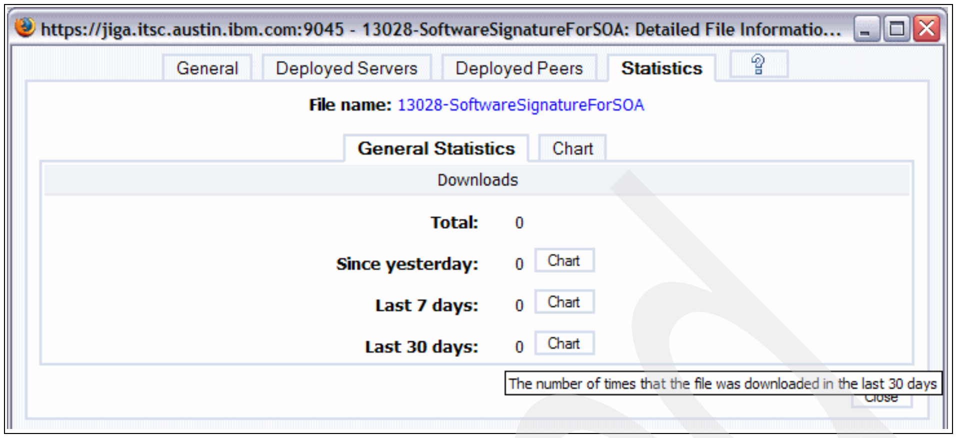
Replication priority: Medium

Replication level: Dynamic, 100 - % of servers per target list

Assigned target lists: [T] 1538c2da990c5989:1f9ba0dd:11234b43b0d:-7ffd

Figure 11-13 File Details panel

From the File Details panel, select the **Statistics** Tab, which shows the detail of how many targets downloaded the file. Until the detail reaches the number of targets you expect, all targets have not connected in to download the software signature file. Figure 11-14 on page 393 shows the contents of the **Statistics** Tab for our example discovery before it is completed.


https://jiga.itsc.austin.ibm.com:9045 - 13028-SoftwareSignatureForSOA: Detailed File Informatio...

General | Deployed Servers | Deployed Peers | Statistics

File name: 13028-SoftwareSignatureForSOA

General Statistics | Chart

Downloads

Total:	0	
Since yesterday:	0	Chart
Last 7 days:	0	Chart
Last 30 days:	0	Chart

The number of times that the file was downloaded in the last 30 days

Close

Figure 11-14 File statistics before completion

You must use the **Reports** → **File Statistics** link to determine whether a distribution was successful after the file is removed from the **File** → **Manage Files** page. Figure 11-15 shows the menu to access to review the file download statistics.

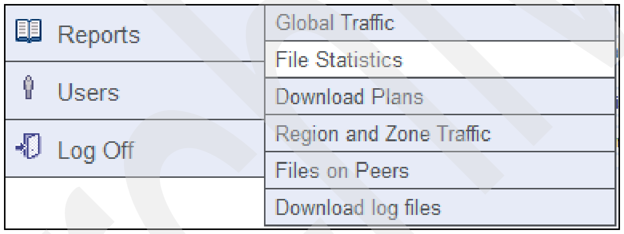
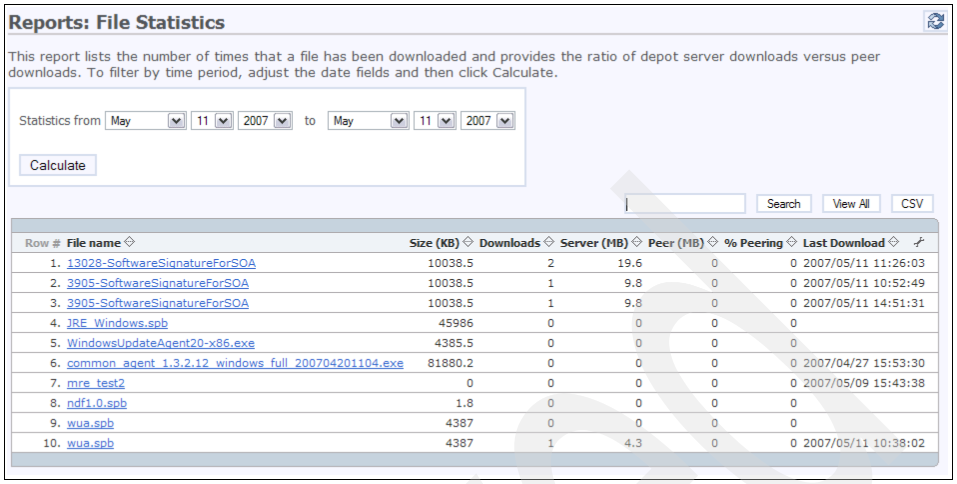


Figure 11-15 Reports menu

Figure 11-16 on page 394, shows the File Statistics page, where there were two downloads for the file 13028-SoftwareSignatureForSOA. Because there were two targets, we now know that the files were downloaded and that the targets should have executed a successful discovery scan.

Note: The common agent cannot contact the dynamic content delivery service manager if it cannot resolve the fully qualified host name of the Manager. The discovery can fail for this reason.



Reports: File Statistics

This report lists the number of times that a file has been downloaded and provides the ratio of depot server downloads versus peer downloads. To filter by time period, adjust the date fields and then click Calculate.

Statistics from May 11 2007 to May 11 2007

Row #	File name	Size (KB)	Downloads	Server (MB)	Peer (MB)	% Peering	Last Download
1.	13028-SoftwareSignatureForSOA	10038.5	2	19.6	0	0	2007/05/11 11:26:03
2.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/11 10:52:49
3.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/11 14:51:31
4.	JRE_Windows.spb	45986	0	0	0	0	
5.	WindowsUpdateAgent20-x86.exe	4385.5	0	0	0	0	
6.	common_agent_1.3.2.12_windows_full_200704201104.exe	81880.2	0	0	0	0	2007/04/27 15:53:30
7.	mre_test2	0	0	0	0	0	2007/05/09 15:43:38
8.	ndf1.0.spb	1.8	0	0	0	0	
9.	wua.spb	4387	0	0	0	0	
10.	wua.spb	4387	1	4.3	0	0	2007/05/11 10:38:02

Figure 11-16 Dynamic content delivery service File Statistics panel

Monitoring the discovery on the common agent

On the common agent, we can now use the files referenced in 11.3.3, "Common Agent Services" on page 347. Look and see what happened when the common agent connected into the device management service and then to the dynamic content delivery service. In Example 11-26, you can see where the job is received from the device management service server by dynamic content delivery service.

Example 11-26 $AgentInstallDir/ep/base/logs/cds_trace_client.log

```
12:21:55-05:00 com.ibm.tivoli.cds.client.protocolhandlers.cds.CDSURLConnection
CDSURLConnection JES Job Processor Entry, parm 1 = name=logger.cds_client,
class=com.ibm.tivoli.cds.cas.CASJLogLoggingAdapter, isLogging=true,
loggedEvents=0, levelName=ALL, levelValue=-2147483648, levelName=ALL,
levelValue=-2147483648

12:21:55-05:00 com.ibm.tivoli.cds.client.protocolhandlers.cds.CDSURLConnection
CDSURLConnection JES Job Processor CDS Client URL Handler object Requested

12:21:55-05:00 com.ibm.tivoli.cds.client.protocolhandlers.cds.CDSURLConnection
CDSURLConnection JES Job Processor Exit
```

Further down in the cds_trace_client.log, you can also find where the common agent references the file it will download. Example 11-27 on page 395, shows the line that indicates that the download was requested.

Example 11-27 $AgentInstallDir/ep/base/logs/cds_trace_client.log

```
12:21:55-05:00 com.ibm.tivoli.cds.client.CDSClient download JES Job Processor
Download of package 1178903687125 to C:\Program
Files\tivoli\ep\runtime\agent\subagents\cds\downloads\1178903687125 requested
```

Note: In the message in Example 11-27, the *1178903687125* is the file ID on the dynamic content delivery service server of the file to be downloaded. You can confirm or reference this by looking at Figure 11-13 on page 392.

Moving further in the file, you can see where the common agent receives the download plan and begins downloading the file from the available depots. Example 11-28 shows that the depot server to be used is ‘nice.itsc.austin.ibm.com’ and that the total size to be downloaded is 10279391 bytes.

Example 11-28 $AgentInstallDir/ep/base/logs/cds_trace_client.log

```
12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 Initiating new download of 1178903687125-1

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 Chunk #0: [0 - 10279391] Completed 0 out of 10279391 bytes

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 There are 1 depot servers in the download plan.

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 [0] nice.itsc.austin.ibm.com

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 There are 0 peer servers in the download plan.

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 Adding depot server nice.itsc.austin.ibm.com to the pool of free
depot server workers.

12:21:55-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 Downloading...
```

Finally, you see the line in Example 11-29, which indicates that the download of the software signature file completed. The next step is that the Common Inventory Technology (CIT) Agent scan starts and returns data.

Example 11-29 $AgentInstallDir/ep/base/logs/cds_trace_client.log

```
12:21:56-05:00 com.ibm.webahead.downloadgrid.client.DownloadManager run
Thread-2433 The download completed. Post download operations are starting.
```

After the CIT Agent starts, you can monitor any errors in the CIT Agent log directory (C:\Program Files\IBM\tivoli\common\CIT\logs). The file traceCIT.log contains errors and information due to the CIT scan itself.

Example 11-30 C:\Program Files\IBM\tivoli\common\CIT\logs\traceCIT.log

```
<Trace Level="MIN">
 <Time Millis="1178904267265">2007-05-11 12:24:27.265-05:00</Time>
 <Server Format="IP">venice</Server>
 <LogText><![CDATA[Unable to register a scan package as a service with the
Windows Update Agent  (err = -2147024893)]]></LogText>
 <Source
FileName="./../../../../../src/plugins/RegPlugin/win/patch/PatchObjectList.cpp"
Method="PatchObjectList::create(bool, PatchObjectList *&amp;)" Line="110"/>
 <Thread>2524</Thread>
 <Process>3284</Process>
</Trace>
```

11.5.2 Common agent installation

To drive Tivoli Configuration Manager operations on Tivoli management agents (TMA) using Tivoli Provisioning Manager for Software, you must install common agents on the systems on which the Tivoli management agents are installed. This section shows you how to troubleshoot the common agent and the Tivoli management agent installation using the TCA for TMA tc driver that Tivoli Provisioning Manager for Software 5.1 Fix Pack 2 provides.

The old version of the TCA for TMA tc driver (provided with Tivoli Provisioning Manager for Software 5.1) contains the Tivoli Common Agent 1.3.1.10 images (only for Windows) together with the JRE (only for Windows). The disadvantages were the following:

- It is only for Windows.
- The tc driver must be rebuilt/reworked every time a new common agent version is ready.
- The tc driver size is very big (it contains Windows JRE and common agent images).

Today, the new version of the TCAFromTMA tc driver is provided with Tivoli Provisioning Manager for Software 5.1 Fix Pack 2. The advantages are the following:

- The tc driver does not contain TCA/JRE images (except for Windows JRE). The images are automatically generated and built at installation time.
- The tc driver is ready for every platform that the common agent supports.
- If a new common agent version is ready, it is no longer needed to rebuild/rework the tc driver.
- The tc driver size is small (it contains only Windows JRE images).

After you install fix pack 2, the CreateTCASPBs post-install workflow is executed. This workflow creates some folders to $TIO_HOME/repository/TCAFromTMA (plan, props and XML) and starts building the SPBs for common agent and JRE. After the workflow execution completes, the XML plan (for example, TCA_PLAN_FOR_TMA-1.3.2.5.xml) is automatically imported as draft into the database.

So if you are in Tivoli Provisioning Manager for Software 5.1 Fix Pack 2, you can use the `%TIO_HOME%`\repository\plan\Plan_TCA_PLAN_FOR_TMA.xml file to install the common agent for the Tivoli management agent.

Before proceeding with the TCAFromTMA installation

There is not a particular way to troubleshoot the TCAFromTMA tc driver; however, you can do the following:

1. Run the CreateTCASPBs workflow, and verify its completion status.
2. Verify the creation of the properties files under $TIO_HOME/repository/TCAFromTMA/props.
3. Verify the creation of the SPB files under $TIO_HOME/repository/TCAFromTMA/spb.
4. Verify the existence of the plan TCA_PLAN_FOR_TMA (saved as draft).
5. Verify the existence of the objects JRE-1.4.2 and TCAForTMA-<version>.

Before installing a common agent in an existent Tivoli management agent (TMA) as an activity plan target, make sure this endpoint was discovered by Tivoli Provisioning Manager for Software as a Tivoli Management Framework object through Tivoli Configuration Manager Discovery or through the tcmReplicate script execution.

To verify that the endpoint is discovered as a Tivoli Management Framework object:

1. Go to **Inventory** → **Manage Inventory** → **Computers**, and search for the desired computer(s). You might be able to find the Tivoli Management Framework discovered computers in the following format:

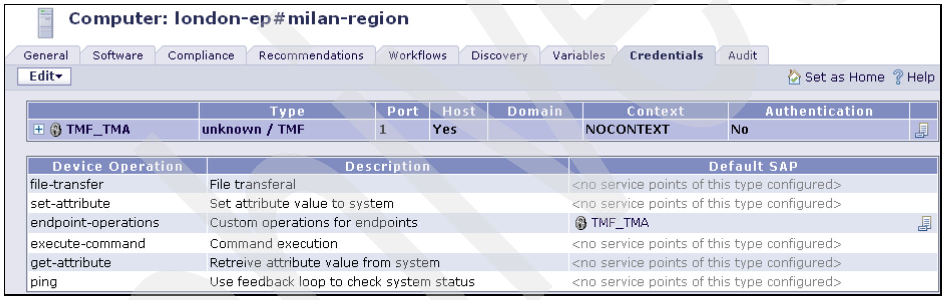
   `<endpoint_name>#<region_name>`

   Following is an explanation of `<endpoint_name>#<region_name>`:

   – *`endpoint_name`* is the name of the endpoint on Tivoli Management Framework.

   – *`region_name`* is the name of the Tivoli Management Region.

2. Click `<endpoint_name>#<region_name>`. You will see the general information of the computer and the Default SAP® column as TMF_TMA instead of RXA, as shown in Figure 11-17.



Computer: london-ep#milan-region

	Type	Port	Host	Domain	Context	Authentication
TMF_TMA	unknown / TMF	1	Yes		NOCONTEXT	No

Device Operation	Description	Default SAP
file-transfer	File transferal	<no service points of this type configured>
set-attribute	Set attribute value to system	<no service points of this type configured>
endpoint-operations	Custom operations for endpoints	TMF_TMA
execute-command	Command execution	<no service points of this type configured>
get-attribute	Retreive attribute value from system	<no service points of this type configured>
ping	Use feedback loop to check system status	<no service points of this type configured>

Figure 11-17 Manage inventory computer credentials

The target(s) used in the TCA for TMA activity plan must also work properly on the Tivoli Management Framework side. You can use the Tivoli Management Framework to try to spawn a method to the endpoint, for example:

```
wadminep <endpoint_name> view_version
```

If the endpoint version was returned, your endpoint is manageable and ready for the Tivoli Provisioning Manager for Software Server activity plan operations to use.

Also, verify the following configurations and requirements:

- Time differences (maximum 2 hours)
- Hard drive disk space needed to install a common agent is about 250 MB

- Settings for the user running installation on the Windows target machine:
  - Local policies
    - Act as part of the operating system
    - Log on as a service
  - TCP IP policies permit all
  - Verify that Internet Information Services (IIS) is stopped and disabled
  - WINS
    - Check the NETBIOS

Monitoring the common agent for the Tivoli management agent activity plan execution

The output of the task is available in the log file of the Activity Planner, which is located in the `$TIO_LOGS`/activityplan directory where all other trace, error, and common base event messages are found. The log file is console.log.

In the following section, we show a sample of the error messages that are displayed through the Tivoli Provisioning Manager for Software GUI Track Task. We also provide some tips on how you can figure out the error messages.

Sample cases of error messages

This section provides sample cases and resolution methods for the following errors:

- `COPTMF085E Software Package TCA Windows Installable FOR TMA.1.3.1.12 does not exist and it cannot be created on server milan`
- `COPTMF055E Operation install failed. See the log files for more information.`
- Activity plan never ends error

COPTMF085E error message

```
COPTMF085E Software Package TCA Windows Installable FOR TMA.1.3.1.12
does not exist and it cannot be created on server milan
```

If you see the error message above after submitting an activity plan that contains a product installation, verify if the SPB file was successfully imported into the Tivoli Management Framework.

See Example 11-31 on page 400. The operation failed after submitting an activity plan to install the common agent for the Tivoli management agent due to issues on importing the package /TCA_Windows.spb on the Tivoli Management Framework Milan Region.

Example 11-31 Activity plan TCAforTMA execution failed

```
2007-05-11 18:53:27,024 DEBUG [Status Updater]
(StatusUpdateProcessor.java:133) manager.StatusUpdateProcessor:
Updating plan instance status of: TCA_PLAN_FOR_TMA@2007/05/11
18:53:07(3092) from SUBMITTED to: IN_PROGRESS
2007-05-11 18:53:47,044 DEBUG [TCAforTMA(13295)] (SPBHandler.java:423)
swd.SPBHandler: putSpb - Sending a null message to close the IOM
channel
2007-05-11 18:53:47,045 DEBUG [TCAforTMA(13295)] (SPBHandler.java:442)
swd.SPBHandler: putSpb - The spb has been moved to
/TCA_Windows.spb_temporary_TPM_5.1_importing_file.spb of the source
host milan#milan-region
2007-05-11 18:53:47,046 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:514) utils.TmfSoftwarePackageUtils: file
moved to /TCA_Windows.spb_temporary_TPM_5.1_importing_file.spb
2007-05-11 18:53:47,048 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:623) utils.TmfSoftwarePackageUtils:
importing the package on TMF
2007-05-11 18:53:47,048 DEBUG [TCAforTMA(13295)] (Utilities.java:143)
common.Utilities: looking up TPM_Region of type PolicyRegion at
hostname milan
2007-05-11 18:53:47,135 DEBUG [TCAforTMA(13295)]
(PolicyRegionUtils.java:188) utils.PolicyRegionUtils: Resource
TaskLibrary already present
2007-05-11 18:53:47,159 DEBUG [TCAforTMA(13295)]
(PolicyRegionUtils.java:188) utils.PolicyRegionUtils: Resource
ProfileManager already present
2007-05-11 18:53:47,180 DEBUG [TCAforTMA(13295)]
(PolicyRegionUtils.java:188) utils.PolicyRegionUtils: Resource
SoftwarePackage already present
2007-05-11 18:53:47,198 DEBUG [TCAforTMA(13295)]
(PolicyRegionUtils.java:188) utils.PolicyRegionUtils: Resource
InventoryConfig already present
2007-05-11 18:53:47,214 DEBUG [TCAforTMA(13295)]
(PolicyRegionUtils.java:188) utils.PolicyRegionUtils: Resource
QueryLibrary already present
2007-05-11 18:53:47,215 DEBUG [TCAforTMA(13295)] (Utilities.java:143)
common.Utilities: looking up TPM_Swd_ProfileManager of type
ProfileManager at hostname milan
2007-05-11 18:53:47,223 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:701) utils.TmfSoftwarePackageUtils:
Creating SoftwarePackage object
```

```
2007-05-11 18:53:48,696 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:727) utils.TmfSoftwarePackageUtils:
setting source host milan#milan-region
2007-05-11 18:53:48,762 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:735) utils.TmfSoftwarePackageUtils:
importing the spb package from
/TCA_Windows.spb_temporary_TPM_5.1_importing_file.spb
2007-05-11 18:53:48,763 DEBUG [TCAforTMA(13295)]
(TmfSoftwarePackageUtils.java:741) utils.TmfSoftwarePackageUtils: using
path = /TCA_Windows.sp
2007-05-11 18:53:53,292 ERROR [TCAforTMA(13295)] (SWDInstall.java:275)
swd.SWDOperation: COPTMF055E Operation IMPORT failed. See the log files
for more information.
2007-05-11 18:53:53,293 DEBUG [TCAforTMA(13295)] (SWDInstall.java:345)
swd.SWDOperation: spo is null: building a failure report for the whole
target list
2007-05-11 18:53:53,294 WARN  [TCAforTMA(13295)] (SWDInstall.java:354)
swd.SWDOperation: Failed target: london-ep#milan-region
2007-05-11 18:53:53,295 DEBUG [TCAforTMA(13295)] (SWDInstall.java:358)
swd.SWDOperation: returning report
2007-05-11 18:53:53,297 DEBUG [TCAforTMA(13295)]
(SoftwareTaskManager.java:280) tmf.SoftwareTaskManager: processing
Software Task completion for task 13295 on server 13063 with status
Failed
```

According to the Tivoli Provisioning Manager server %TIO_LOGS/activityplan/console.log error message, "`COPTMF055E Operation IMPORT failed. See the log files for more information`", you should consult the Tivoli Management Framework and Tivoli Configuration Manager log files because the issue occurred during the import operation, and probably details were registered in log files, if you enabled trace logs. This is the reason TPM returned "`COPTMF085E Software Package TCA Windows Installable FOR TMA.1.3.1.10 does not exist and it cannot be created on server milan`".

You can enable tracing for Tivoli Software Distribution using the command **`wswdcfg`**, as follows:

1. `wswdcfg -s trace_level=5`
2. `wswdcfg -s trace_size=10000000`
3. `wswdcfg -h <source_host> -s trace_level=5`
4. `wswdcfg -h <source_host> -s trace_size=10000000`
5. Verify trace settings: `wswdcfg -s`

6. Delete any existing trace files in the location '$BINDIR/../swdis/' (trace files end in .trc or tpc)
7. Recreate the problem by submitting the activity plan TCAforTMA directly from Tivoli Provisioning Manage Server.

See Example 11-32, which shows the $BINDIR/../swdis/spde1.trc trace file and the import operation error.

Example 11-32 Software distribution trace file

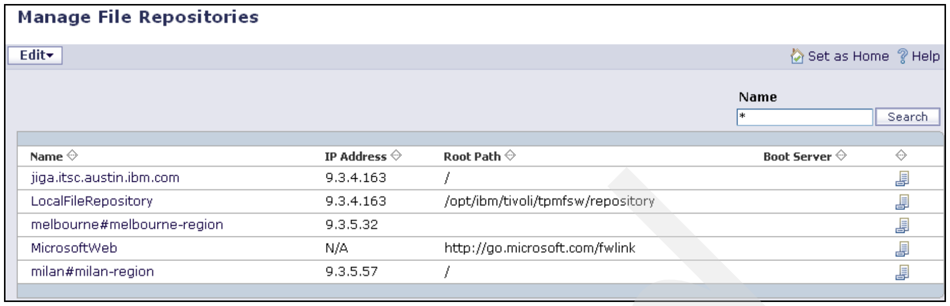
```
[45214:537367944] Fri May 11 19:31:56 CDT 2007.306 [I] get_file_invoke
spinput->path = /TCA_Windows.spb_temporary_TPM_5.1_importing_file.spb
[45214:537367944] Fri May 11 19:31:56 CDT 2007.314 [I] get_file_invoke
invoking method read_from_file on managed node
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I]
SpDistEngine::make_input_channel() looking for exception...
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I]
SpDistEngine::make_input_channel() invalidating env for other thread
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I]
SpDistEngine::make_input_channel() <<<<< EXIT <<<<<
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I] parse_keywlist
>>>> ENTRY >>>>>
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I] parse_keywlist
<<<<< EXIT <<<<<
[45214:536996856] Fri May 11 19:31:56 CDT 2007.318 [I]
SpDistEngine::t_spde_build() start copy of Software Package Block
[45214:536996856] Fri May 11 19:31:56 CDT 2007.319 [I] copy_file >>>>
ENTRY >>>>>
[45214:536996856] Fri May 11 19:31:59 CDT 2007.342 [I]
file_handle::write >>>> ENTRY >>>>>
[45214:536996856] Fri May 11 19:31:59 CDT 2007.343 [E]
file_handle::write failure writing in file /TCA_Windows.sp, errno=28
[45214:536996856] Fri May 11 19:31:59 CDT 2007.343 [I]
file_handle::write message = 'There is not enough space in the file
system.' ...
[45214:536996856] Fri May 11 19:31:59 CDT 2007.343 [I]
file_handle::write return data = 0
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [I]
file_handle::write <<<<< EXIT <<<<<
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [F] copy_file error
writing on output file,rc=5
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [I] copy_file <<<<<
EXIT <<<<<
```

```
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [F]
SpDistEngine::t_spde_build() error writing file '/TCA_Windows.sp',
error=28
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [I]
SpDistEngine::t_spde_build() in catch section
[45214:536996856] Fri May 11 19:31:59 CDT 2007.344 [I]
SpDistEngine::t_spde_build() free input_ch
[45214:536996856] Fri May 11 19:31:59 CDT 2007.345 [I]
SpDistEngine::t_spde_build() free pkg
[45214:536996856] Fri May 11 19:31:59 CDT 2007.345 [I]
SpDistEngine::t_spde_build() free msglist
[45214:536996856] Fri May 11 19:31:59 CDT 2007.345 [I]
SpDistEngine::t_spde_build() free src_before program
[45214:536996856] Fri May 11 19:31:59 CDT 2007.345 [I]
SpDistEngine::t_spde_build() free src_after program
[45214:536996856] Fri May 11 19:31:59 CDT 2007.345 [I]
SpDistEngine::t_spde_build() free buffer for Software Package
data
```

According to the highlighted error message in Example 11-32 on page 402, there is not enough space in the file system to import the spb file. Tivoli Provisioning Manager for Software creates a copy of the original spb file, which is named <spb_file_name>_temporary_TPM_5.1_importing_file.spb, before importing it into Tivoli Management Framework, so it uses a temporary directory for that.

Important: The temporary directory is defined according to the Tivoli Management Region File Repository, Root Path, which is root or "/" by default.

In this sample case, there is not enough space on the root path of `milan#milan-region`. To check or edit properties of File Repositories, go to **Inventory** → **Infrastructure Management** → **File Repositories**. It is not necessary to restart Tivoli Provisioning Manager to validate the root path change.



Manage File Repositories

Name	IP Address	Root Path	Boot Server
jiga.itsc.austin.ibm.com	9.3.4.163	/	
LocalFileRepository	9.3.4.163	/opt/ibm/tivoli/tpmfsw/repository	
melbourne#melbourne-region	9.3.5.32		
MicrosoftWeb	N/A	http://go.microsoft.com/fwlink	
milan#milan-region	9.3.5.57	/	

Figure 11-18 Manage File Repositories window

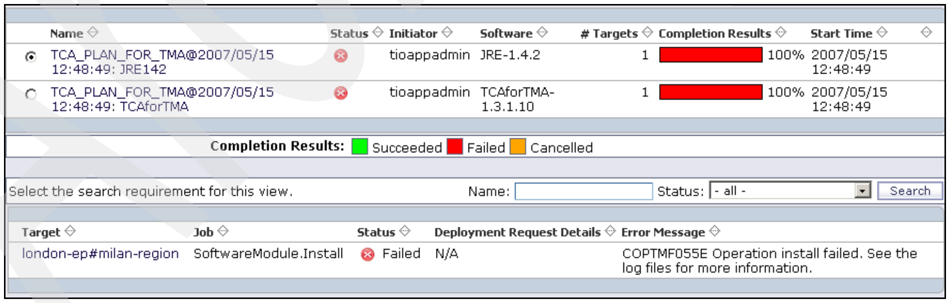
After you submit a software distribution or inventory task from Tivoli Provisioning Manager to Tivoli Management Framework, double-check if the respective distribution was created in Mdist2, and follow its progress using the **`wmdist`** commands.

In case of an unsuccessful software package installation, view the $LCF_DATDIR/lcfd.log file of the TMA or the $BINDIR/../swdis/work/<profile_name>.log file for further details about the error.

COPTMF055E Operation install failed error message

```
COPTMF055E Operation install failed. See the log files for more information.
```

See if the deployment request ID is N/A. The problem is probably that the operation could not be submitted in Tivoli Management Region, as shown in Figure 11-19.



Name	Status	Initiator	Software	# Targets	Completion Results	Start Time
TCA_PLAN_FOR_TMA@2007/05/15 12:48:49: JRE142		tioappadmin	JRE-1.4.2	1	🟥 100%	2007/05/15 12:48:49
TCA_PLAN_FOR_TMA@2007/05/15 12:48:49: TCAforTMA		tioappadmin	TCAforTMA-1.3.1.10	1	🟥 100%	2007/05/15 12:48:49

Completion Results: 🟩 Succeeded 🟥 Failed 🟧 Cancelled

Select the search requirement for this view. Name: Status: - all -

Target	Job	Status	Deployment Request Details	Error Message
london-ep#milan-region	SoftwareModule.Install	Failed	N/A	COPTMF055E Operation install failed. See the log files for more information.

Figure 11-19 Activity plan operation install failed

In this sample case, check $BINDIR/../swdis/work/<swd_profile_name>.log and $LCF_DATDIR/lcfd.log for further details. See Example 11-33 on page 405, which shows a cm_status check validation failure.

Example 11-33   Software package validation failure

```
Software Package: "TCA Windows Installable FOR TMA.1.3.1.10"
Operation:          install
Mode:               not-transactional,not-undoable
Time:               2007-05-14 17:55:16
=================
DISSE0072E List of targets on which the requested operation cannot be
submitted:
london-ep   DISSE0407E Failed cm_status check.
```

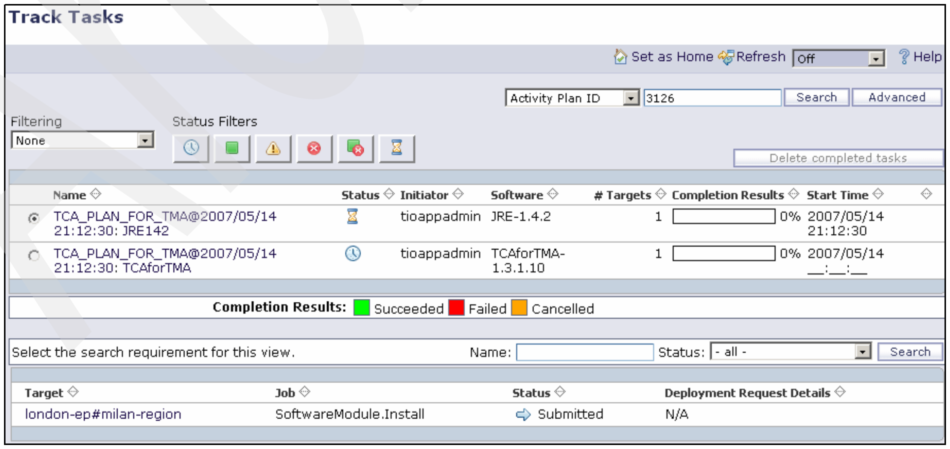
Also, see Example 11-34, which shows the exception related to the nested software package validation issue registered in $LCF_DATDIR/lcfd.log file at the Tivoli Management Framework target.

Example 11-34   lcfd.log error when cm_check fails

```
May 14 18:59:21 Q MethInit ** Exception caught in run_impl: unknown
exception:
Exception:UserException:SysAdminException::ExException:SpDistEngine::Sp
de::ExNestedSpErrorMsg
```

Activity plan never ends error

To track an Activity Task, go to **Task Management** → **Track Activity Plans**, and click the activity plan name. If the activity plan never ends, for example the first task is already submitted but it never completes; therefore, the next ones keep waiting for submission, as shown in Figure 11-20, use the following steps to troubleshoot the issue.



Track Tasks

Set as Home Refresh Off Help

Activity Plan ID 3126 Search Advanced

Filtering: None   Status Filters

Delete completed tasks

	Name	Status	Initiator	Software	# Targets	Completion Results	Start Time
●	TCA_PLAN_FOR_TMA@2007/05/14 21:12:30: JRE142	⌛	tioappadmin	JRE-1.4.2	1	0%	2007/05/14 21:12:30
○	TCA_PLAN_FOR_TMA@2007/05/14 21:12:30: TCAforTMA	🕓	tioappadmin	TCAforTMA-1.3.1.10	1	0%	2007/05/14 __:__:__

Completion Results: 🟩 Succeeded 🟥 Failed 🟧 Cancelled

Select the search requirement for this view. Name: Status: - all - Search

Target	Job	Status	Deployment Request Details
london-ep#milan-region	SoftwareModule.Install	⇨ Submitted	N/A

Figure 11-20   Pending activities

1. Go to $TIO_LOGS/activityplan/console.log for UNIX or %TIO_LOGS%/activityplan/console.log for Windows systems to see if the activity was successfully submitted. If so, you will see the following line:
```
2007-05-15 12:32:10,363 DEBUG [Status Updater]
(StatusUpdateProcessor.java:133) manager.StatusUpdateProcessor:
Updating plan instance status of: TCA_PLAN_FOR_TMA@2007/05/15
12:32:05(3400) from SUBMITTED to: IN_PROGRESS
```
2. Go to the Tivoli Management Region, and check if the distribution related to the first activity was created over Mdist2. You can use the **wmdist** command. If necessary, consult *Tivoli Management Framework Reference Guide, Version 4.1.1,* SC32-0806 for further details.
3. If the distribution is successfully completed but Tivoli Provisioning Manager never submits the next activity, you probably have a Report Manager component issue in Tivoli Management Framework because it is also responsible for giving a feedback to Tivoli Provisioning Manager for Software about invoked methods in Tivoli Management Framework.

   Note: Double check the conditions for the next activity submission in the activity plan.

4. To confirm the Report Manager's health, follow the procedure described in 11.4.3, "Report Manager" on page 374.

11.5.3 Installing through SDI

This section discusses the installation of a software package through the Scalable Distribution Infrastructure (SDI). See 1.3.2, "Scalable Distribution Infrastructure" on page 15 for detailed information about SDI.

In this example, we install the installable, *ndf*, which contains the software package file named **ndf.1.0.spb**, to two common agents [nice.itsc.austin.ibm.com and venice.itsc.austin.ibm.com] through Tivoli Provisioning Manager for Software. The key infrastructure components involved during a software install through SDI are the device management service and the dynamic content delivery service server and depots.

Starting the installation

1. The Install Software Products Task is initiated for the installable *ndf* to nice.itsc.austin.ibm.com and venice.itsc.austin.ibm.com using the **Software Management** → **Install** → **Software Products** menu item, as shown in Figure 11-21 on page 407.

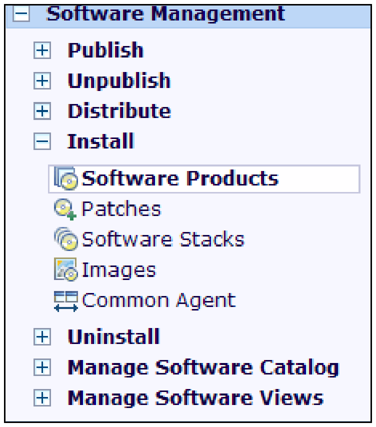
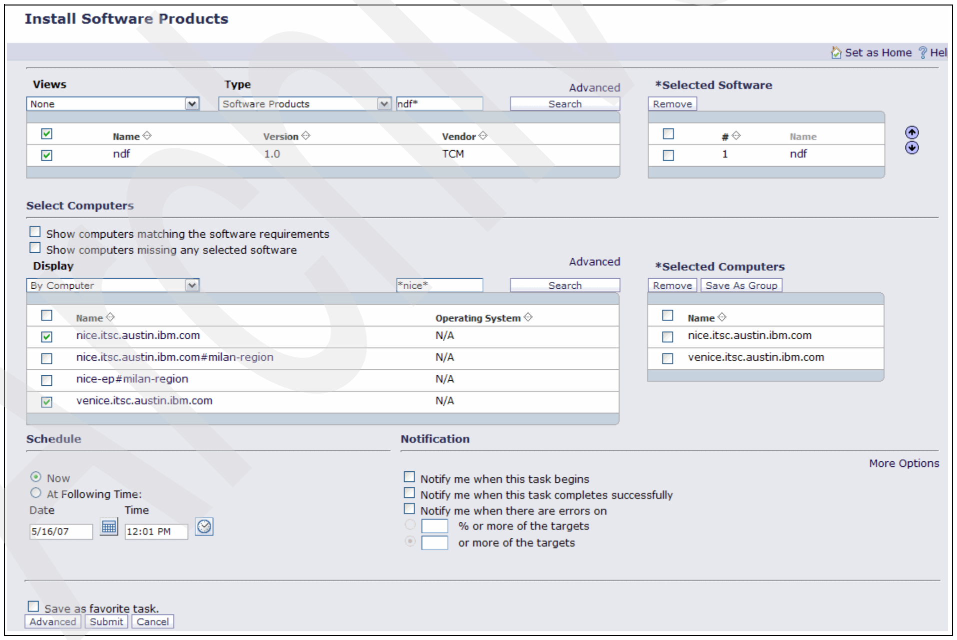


Figure 11-21 Starting an installation

2. Select the installable package (*ndf)* and the targets (nice.itsc.austin.ibm.com and venice.itsc.austin.ibm.com). Figure 11-22 shows the Install Software Products panel where you make these selections.



Install Software Products

Set as Home | Hel

Views: None
Type: Software Products
ndf*
Advanced | Search

☑	Name	Version	Vendor
☑	ndf	1.0	TCM

*Selected Software
Remove

☐	#	Name
☐	1	ndf

Select Computers

☐ Show computers matching the software requirements
☐ Show computers missing any selected software

Display: By Computer
nice
Advanced | Search

☐	Name	Operating System
☑	nice.itsc.austin.ibm.com	N/A
☐	nice.itsc.austin.ibm.com#milan-region	N/A
☐	nice-ep#milan-region	N/A
☑	venice.itsc.austin.ibm.com	N/A

*Selected Computers
Remove | Save As Group

☐	Name
☐	nice.itsc.austin.ibm.com
☐	venice.itsc.austin.ibm.com

Schedule

[● Now]
○ At Following Time:
Date: 5/16/07 Time: 12:01 PM

Notification

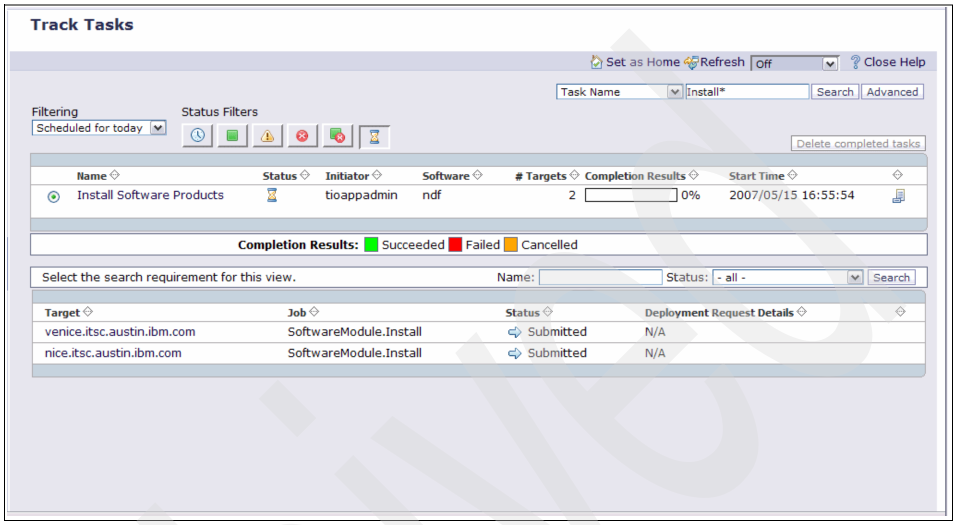
More Options

☐ Notify me when this task begins
☐ Notify me when this task completes successfully
☐ Notify me when there are errors on
○ % or more of the targets
○ or more of the targets

☐ Save as favorite task.
Advanced | Submit | Cancel

Figure 11-22 Install Software Products panel

3. After you make your selection and the task starts, you can monitor the completion of the task using the Track Tasks panel, as shown in Figure 11-23. This panel shows that the task started and is in Submitted status on the two targets, but does not show any further detail.



	Name	Status	Initiator	Software	# Targets	Completion Results	Start Time
●	Install Software Products	⌛	tioappadmin	ndf	2	0%	2007/05/15 16:55:54

Completion Results: 🟩 Succeeded 🟥 Failed 🟧 Cancelled

Select the search requirement for this view. Name: Status: - all - Search

Target	Job	Status	Deployment Request Details
venice.itsc.austin.ibm.com	SoftwareModule.Install	Submitted	N/A
nice.itsc.austin.ibm.com	SoftwareModule.Install	Submitted	N/A

Figure 11-23 Track Tasks

Monitoring the distribution

To determine exactly what is going on, remember what occurs in the process:

1. ndf.1.0.spb is published to dynamic content delivery service.
   a. Dynamic content delivery service calculates the depots to publish to based on the target lists regions.
   b. The file is placed on the depots.
2. A job is submitted to device management service for the targets to install the file.
3. The targets connect into the device management service based on the job execution service polling interval (default is 60 minutes).
4. The device management service gives a job to the target, which informs it to connect to the dynamic content delivery service server to get a download plan.
5. The agent connects to the dynamic content delivery service server and retrieves its download plan that contains which depots or peers that have the file.
6. The agent retrieves the file(s) from the depots based on the download plan.

7. After the job (install) is complete, the agent sends a report to the device management service.
8. The device management service receives the report and updates the Tivoli Provisioning Manager server based on its own polling interval (default 10 minutes).

You can observe these steps in detail through the log files.

Publishing the file

The first step in distributing a file is for the Tivoli Provisioning Manager server to *publish* the file to the dynamic content delivery service infrastructure. We can use the $TIO_LOGS/activityplan/console.log at the DEBUG logging level (described in 11.2.5, “Activity plan engine” on page 334) to confirm if the file was published.

Example 11-35 shows the definition of the file to be published.

Example 11-35 activityplan/console.log file publish

```
2007-05-15 16:55:55,689 DEBUG [Install Software Products(14742)]
(FileManager.java:644) cds.FileManager: DeviceModelId=2403 for FileRepository
LocalFileRepository. LocalFileRepositoryDM Id is 2403
2007-05-15 16:55:55,689 DEBUG [Install Software Products(14742)]
(FileManager.java:773) cds.FileManager: Installable file is
/opt/ibm/tivoli/tpmfsw/repository/ndf1.0.spb, version=1.0,
description=Description of installable ndf
```

Example 11-36 shows the target list for the file, venice.itsc.austin.ibm.com and nice.itsc.austin.com. These targets are used to determine to which depot(s) the dynamic content delivery service should publish the file.

Example 11-36 activityplan/console.log target list

```
2007-05-15 16:55:55,698 DEBUG [Install Software Products(14742)]
(FileManager.java:248) cds.FileManager: HostName is venice.itsc.austin.ibm.com
IP Address is 9.3.5.26 for device 6938
2007-05-15 16:55:55,701 DEBUG [Install Software Products(14742)]
(FileManager.java:248) cds.FileManager: HostName is nice.itsc.austin.ibm.com IP
Address is 9.3.5.88 for device 6899
```

Example 11-37 on page 410 shows that the PackageId for the file is 1179266129194. This is important when searching for activity in the dynamic content delivery service log files or on the common agent. You can also find this information in the dynamic content delivery service admin interface.

Example 11-37 PackageId for the file is 1179266129194

```
2007-05-15 16:55:55,710 DEBUG [Install Software Products(14742)]
(FileManager.java:160) cds.FileManager: URL is https://9.3.4.163:9045 username
tioappadmin
2007-05-15 16:56:03,035 DEBUG [Install Software Products(14742)]
(FileManager.java:356) cds.FileManager: PackageId for new file published is
1179266129194
2007-05-15 16:56:03,811 DEBUG [Install Software Products(14742)]
(FileManager.java:378) cds.FileManager: Published file
/opt/ibm/tivoli/tpmfsw/repository/ndf1.0.spb with taskId 14742 to CDS depots:
nice.itsc.austin.ibm.com,
```

The last line in Example 11-37 indicates that the file `C:/Program Files/ibm/tivoli/tpmfsw/repository/ndf1.0.spb` was successfully published to the depot `nice.itsc.austin.ibm.com`.

Note: As we mentioned in 11.2.5, “Activity plan engine” on page 334, you can follow the flow of a particular activity by using the activity name (Install **`Software Products`** in this case) or the taskId (**`14742`** in this case). To do this for this example, use the command: **`grep ‘22781’ $TIO_LOGS/activityplan/console.log`**

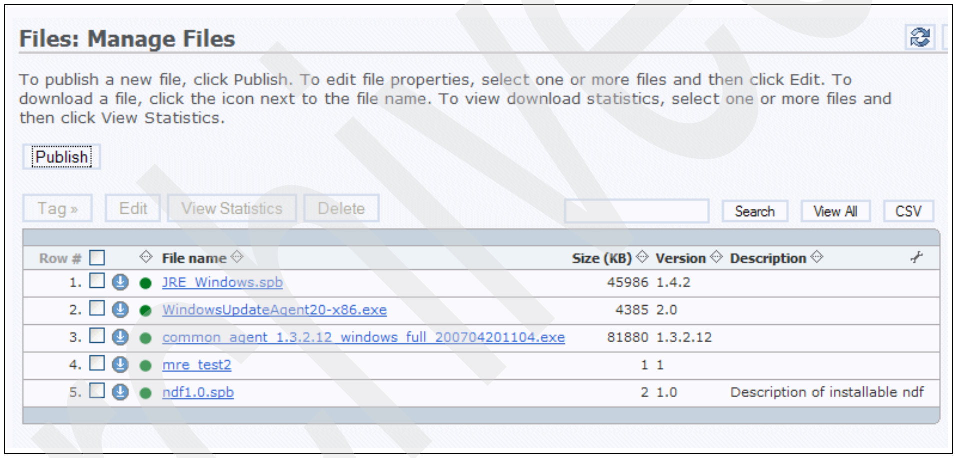
On the dynamic content delivery service side, the file trace_cdsmgr.log shows that the file was successfully published. Example 11-38 shows the relevant content of the trace_cdsmgr.log file.

Example 11-38 trace_cdsmgr.log file is published

```
2007-05-15 16:56:01.994-06:00 com.ibm.tivoli.cds.gui.servlets.AddPackage doPost
jiga.itsc.austin.ibm.com IP CTGDED044I The file ndf1.0.spb was successfully
registered with the management center. The file's unique id is 1179266129194 .

2007-05-15 16:56:02.029-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP CTGDED046I The operation to choose potential upload
servers so that client 9.3.4.163 can upload file with unique id 1179266129194
is starting.

2007-05-15 16:56:02.031-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP CTGDED047W No region was found for client
9.3.4.163. The operation to choose potential upload servers will try to
continue. Please update the ip ranges/domains to ensure optimal service.
```

```
2007-05-15 16:56:02.042-06:00
com.ibm.webahead.downloadgrid.server.optimizedplan.nooptimization.OptimizedPlan
GeneratorNoOptimization choosePotentialUploadServers
jiga.itsc.austin.ibm.com IP not enough upload servers were found in the
package's target lists, considering all depot servers

2007-05-15 16:56:03.012-06:00
com.ibm.webahead.downloadgrid.server.service.admin.implement.DownloadGridAdminI
mpl addDataUnit2Server      jiga.itsc.austin.ibm.com IP CTGDED007I File
1179266129194-1 was added to server nice.itsc.austin.ibm.com:2100 successfully.
```

Figure 11-24 shows the dynamic content delivery service user interface at https://<tpmserver>:9045/admin → **File** → **Manage File**. This interface shows that the file ndf1.0.spb was not uploaded to the dynamic content delivery service server. We can click the file to see more information.

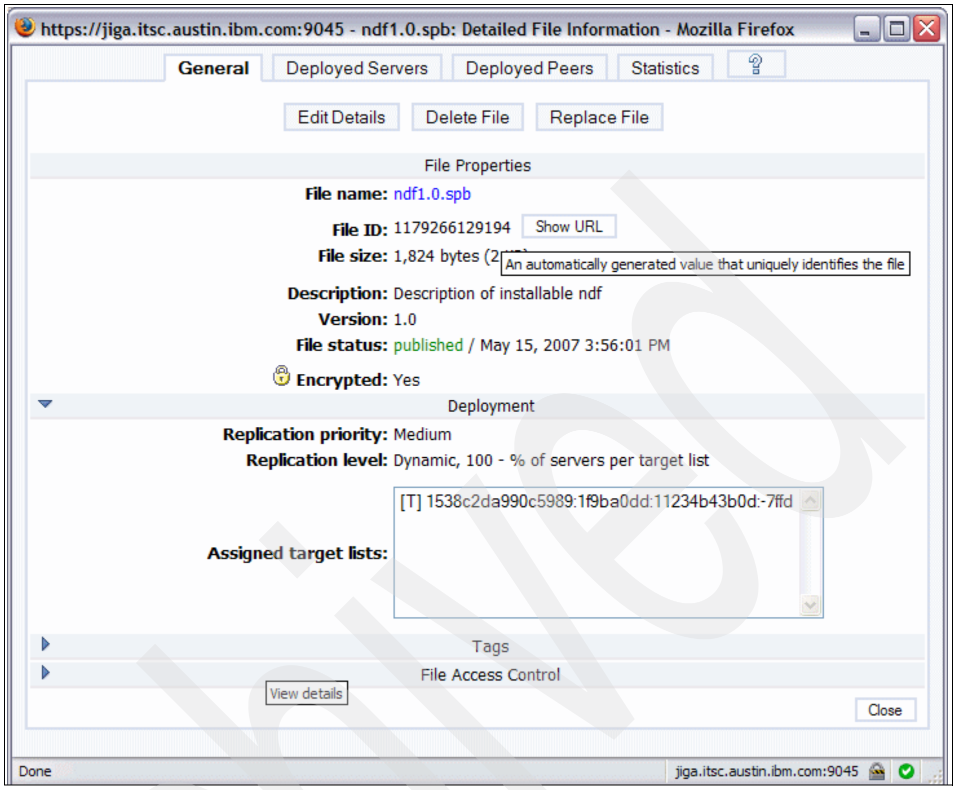


To publish a new file, click Publish. To edit file properties, select one or more files and then click Edit. To download a file, click the icon next to the file name. To view download statistics, select one or more files and then click View Statistics.

Row #	File name	Size (KB)	Version	Description
1.	JRE_Windows.spb	45986	1.4.2	
2.	WindowsUpdateAgent20-x86.exe	4385	2.0	
3.	common_agent_1.3.2.12_windows_full_200704201104.exe	81880	1.3.2.12	
4.	mre_test2	1	1	
5.	ndf1.0.spb	2	1.0	Description of installable ndf

Figure 11-24 Dynamic content delivery service File Manage Files panel

Figure 11-25 on page 412 shows the details of the file, including the File ID of 1179266129194, which you can use to track activity in the log files related to this file.


https://jiga.itsc.austin.ibm.com:9045 - ndf1.0.spb: Detailed File Information - Mozilla Firefox

General | Deployed Servers | Deployed Peers | Statistics

Edit Details | Delete File | Replace File

File Properties

File name: ndf1.0.spb
File ID: 1179266129194 Show URL
File size: 1,824 bytes (2 An automatically generated value that uniquely identifies the file
Description: Description of installable ndf
Version: 1.0
File status: published / May 15, 2007 3:56:01 PM
Encrypted: Yes

Deployment

Replication priority: Medium
Replication level: Dynamic, 100 - % of servers per target list
Assigned target lists: [T] 1538c2da990c5989:1f9ba0dd:11234b43b0d:-7ffd

Tags

File Access Control

View details

Close

Done | jiga.itsc.austin.ibm.com:9045

Figure 11-25 ndf1.0.spb file details

We have now confirmed, using several different methods, that the file, ndf1.0.spb, was published to the dynamic content delivery service server and is now available for download. The next step is that Tivoli Provisioning Manager will submit a job to the device management service.

Tivoli Provisioning Manager submits the job to the device management service

The second step in the distribution of a software package to a common agent is that a job is submitted to the device management service. This job ensures that when the common agent connects to the device management service to check for jobs, the device management service will tell the common agent to connect to the dynamic content delivery service to receive a download plan. Continuing further in the $TIO_LOGS/activityplan/console.log, following the same task Id as before (14742), we can check if the job was successfully submitted.

Example 11-39 shows the successful submission of the job.

Example 11-39 `activityplan/console.log job submission`

```
2007-05-15 16:56:03,925 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:307) client.JobManagementServiceClient: Using
endpoint URL:
https://jiga.itsc.austin.ibm.com:9045/dmserver/services/EDMSTPMManagementServic
e
2007-05-15 16:56:03,955 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:562) client.JobManagementServiceClient:
Submitting JMS job
2007-05-15 16:56:03,983 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:563) client.JobManagementServiceClient:
descriptor =
2007-05-15 16:56:03,984 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:564) client.JobManagementServiceClient:
activationDate = 1179266163926
2007-05-15 16:56:03,984 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:565) client.JobManagementServiceClient:
expirationDate = 1180475763926
2007-05-15 16:56:03,984 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:566) client.JobManagementServiceClient:
priority = 3
2007-05-15 16:56:03,985 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:567) client.JobManagementServiceClient:
interval = 0
2007-05-15 16:56:03,985 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:568) client.JobManagementServiceClient:
interval unit = null
2007-05-15 16:56:03,985 DEBUG [Install Software Products(14742)]
(JobManagementServiceClient.java:569) client.JobManagementServiceClient:
target[0] = 6938
2007-05-15 16:56:06,379 DEBUG [Install Software Products(14742)]
(SoaInfrastructure.java:93) soa.SoaInfrastructure:
SoaInfrastructure.executeTask exit
```

On the device management service side, you can confirm that the job was submitted by looking in the file $TIO_HOME/tioprofile/logs/server1/TraceDMS1.log, if debugging is turned up.

Example 11-40 shows the start of the job.

Note: The start of the job is determined by the string *createJob*, and if there are multiple entries when you search, you have to match the time the job starts. For example the activityplan/console.log shows that the job submitted at `2007-05-15 16:56:03` and the TraceDMS1.log shows a createJob at `05/15/2007 4:56:06`.

Example 11-40   TraceDMS1.log job start

```
05/15/2007 4:56:06.178 com.tivoli.dms.federator.FederatedJobManager createJob
Migrated.Servlet.Engine.Transports : 284
  Job Object len in 9450

05/15/2007 4:56:06.179 com.tivoli.dms.federator.FederatedJobManager createJob
Migrated.Servlet.Engine.Transports : 284
  Create Federator job -
```

Check if job was received for processing by the common agent

The next thing that occurs in the distribution of a software package to a common agent is that the common agent endpoint connects to the device management service server to check for jobs, which you can observe in the <ep_install_dir>logs/rcp.log.# file that is locally on the endpoint.

Example 11-41 shows the portion of the rcp.log [at debug level ALL] where the common agent receives the job.

Example 11-41   rcp.log receiving a job

```
May 15, 2007 5:07:49 PM
com.ibm.tivoli.osgi.service.jes.impl.JobExecutionServiceImpl submitJob
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Job submitted to queue
May 15, 2007 5:07:49 PM
com.ibm.tivoli.osgi.service.jes.impl.JobExecutionServiceImpl submitJob
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Job added to queue
May 15, 2007 5:07:49 PM com.ibm.pvc.wct.internal.logredirector.Tstream println
INFO: Waiting for JDS Completion
May 15, 2007 5:07:49 PM com.ibm.tivoli.osgi.service.jes.impl.JobQueue getNext
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Deleted job from the job queue: C:\Program
Files\tivoli\ep\conf\org.eclipse.osgi\bundles\64\data\jobQueue\job_117926686957
8_b8121aa1033011dc855b000d6049368e.xml
```

From the log snippet in Example 11-41 on page 414, you can see that the agent retrieves the job and places the job in a queue to be processed. It is then deleted off of the queue when the job is going to be processed. A job consists of *WorkItems* that are executed by the JES.

Note: A job has a general format that consists of a header and WorkItems. The header looks similar to the following example:

```
<Job priority="1" name="SPBInstall" failurePolicy="2" executionMode="1"
rebootAfter="null">
```

A WorkItem looks similar to the following example:

```
<WorkItem priority="1" binding="service:osgi" name="LocalFileExist_10720" >
<Parameter
   name="service"
   type="string"
   value="<Value
   type="primitive"
   isLiteral="true"
   value="com.ibm.tivoli.tpm.osgi.service.FileManagementService"/>"/>
```

There can be many parameters within a WorkItem.

```
<Parameter ... />
<Parameter ... />
</WorkItem> <-- End of WorkItem
```

There can be many WorkItems within a Job.

```
<WorkItem ... >
</WorkItem>
```

There must be an end for the Job.

```
</Job> <-- End of the Job
```

The next portion of the log shows what a job looks like. Example 11-42 shows the job in XML format.

Example 11-42 rcp.log job content

```
May 15, 2007 5:07:49 PM com.ibm.tivoli.osgi.service.jes.impl.JobProcessor run
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Retrieved job from job queue:
<Job priority="1" name="SPBInstall" failurePolicy="2" executionMode="1"
rebootAfter="null"><WorkItem priority="1" binding="service:osgi"
name="LocalFileExist_10720" >
```

```
...
<Parameters Deleted>
...
</WorkItem><WorkItem priority="2" binding="service:osgi"
name="GetInstallable_10720" >
...
<Parameters Deleted>
...
</WorkItem><WorkItem priority="3" binding="service:osgi"
name="InstallInstallable_10720" >
...
<Parameters Deleted>
...

</WorkItem></Job>
```

The last line in Example 11-43 indicates that the common agent successfully received a job for processing and will execute the job “SPBInstall”. This only indicates that we have a job named SPBInstall. We still need to actually download an SPB that can be installed.

Example 11-43   rcp.log receiving job

```
May 15, 2007 5:07:49 PM com.ibm.tivoli.osgi.service.jes.impl.JobProcessor
process
INFO: JES023I Processing job: name = [SPBInstall], requestId =
[b8121aa1033011dc855b000d6049368e]
May 15, 2007 5:07:49 PM com.ibm.tivoli.osgi.service.jes.impl.JobProcessor
process
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Executing Job:
[SPBInstall]
```

The common agent executes each WorkItem in the job

As described in Example 11-42 on page 415, the common agent executes each WorkItem in the job. Three WorkItems are executed:

- LocalFileExist_10720
- GetInstallable_10720
- and InstallInstallable_10720

The first WorkItem to execute is *LocalFileExist_1072*. This WorkItem checks if the file that is going to be downloaded already exists. Example 11-44 on page 417 shows this WorkItem executing.

Example 11-44 rcp.log LocalFileExist_10720 WorkItem

```
May 15, 2007 5:07:49 PM
com.ibm.tivoli.tpm.osgi.service.impl.FileManagementServiceImpl isFileExist
FINEST: NOTE  ==>com.ibm.tivoli.tpm.osgi.service.resource.TpmMessages TPM
(INFO): File C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\ndf1.0.spb does not exist
```

The second WorkItem to execute is *GetInstallable_10720*. This WorkItem actually downloads the appropriate file from the appropriate depots or peers. Example 11-45 shows a call to the FileManagementService, copyFile, which shows what file [cdss://9.3.4.163:9046/1179266129194] will be copied and to where it will be copied
[file://C:/DOCUME~1/ADMINI~1/LOCALS~1/Temp/ndf1.0.spb].

Example 11-45 rcp.log GetInstallable_10720 WorkItem

```
May 15, 2007 5:07:50 PM
com.ibm.tivoli.tpm.osgi.service.impl.FileManagementServiceImpl copyFile
INFO: TPMFMS001I File copy:
   source file:
cdss://9.3.4.163:9046/1179266129194?retryWindowSec=7200&maxRetryIntervalSec=30
   target file:
file://C:/DOCUME~1/ADMINI~1/LOCALS~1/Temp/ndf1.0.spb
```

Note: The number *1179266129194*, in Example 11-45, is the File ID for the file stored in the dynamic content delivery service server, which you can confirm in exactly the same manner that we discussed in the 11.5.1, “Discovery” on page 380 using the dynamic content delivery service server at the following location:

```
URL: https://<tpmserver>:9045/admin
```

The third WorkItem to execute is *InstallInstallable_10720*. This WorkItem actually executes the same disconnected commands that we used to install SPB files in Tivoli Configuration Manager 4.2 to install the file ndf1.0.spb on the common agent. Example 11-46 shows the execution of the **wdinstsp** command to install the SPB.

Example 11-46 rcp.log InstallInstallable_10720 WorkItem

```
May 15, 2007 5:07:53 PM
com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.impl.SPBHandler
getDeploymentDirectory
FINE: NOTE
==>com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.resource.SpbMessages
looking for swd_env script directory swd_env.bat
```

```
May 15, 2007 5:07:53 PM
com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.impl.SPBHandler
getDeploymentDirectory
FINE: NOTE
==>com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.resource.SpbMessages
found at /C:/Program
Files/tivoli/ep/runtime/base/../../conf/org.eclipse.osgi/bundles/68/1/.cp/swd_e
nv.bat
May 15, 2007 5:07:53 PM
com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.impl.SPBHandler
executeCommand
FINE: NOTE
==>com.ibm.tivoli.orchestrator.endpoint.tca.handlers.spb.resource.SpbMessages
Executing: wdinstsp -n ndf.1.0 -f -R y
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\ndf1.0.spb
```

Completing the distribution

After the common agent retrieves the distribution and the distribution is installed locally, the common agent must update the device management service with its results. The device management service must then update the Tivoli Provisioning Manager server with the results as well so that the Tivoli Provisioning Manager server can update the Task status and cleanup after itself in the dynamic content delivery service. If the file that was distributed was not specifically published first, then it must be removed from the depots after all the targets have received the files.

Confirm that the job was processed with results

Example 11-47 shows that the job is “Done executing”, and that the results are ‘Job success’. After the job is complete, a set of XML content is generated that contains the detailed job results, which are sent back to the device management service server for processing.

Example 11-47 rcp.log job status

```
May 15, 2007 5:07:57 PM com.ibm.tivoli.osgi.service.jes.impl.JobProcessor
process
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Done executing Job: [SPBInstall]
May 15, 2007 5:07:57 PM com.ibm.tivoli.osgi.service.jes.impl.JobProcessor run
FINEST: NOTE  ==>com.ibm.tivoli.osgi.service.jes.resources.JesMessages JES
(INFO): Job success: <?xml version="1.0" encoding="UTF-8"?>
```

The content of the result of the XML, which starts on the last line in Example 11-47, contains a section for the overall Job Status shown in Example 11-48 on page 419. This job status shows that the requestID for the job is **`b8121aa1033011dc855b000d6049368e`**, and the status of the job is **6**.

Example 11-48 rcp.log XML results

```
<java version="1.4.2" class="java.beans.XMLDecoder">
 <object class="com.ibm.tivoli.tpm.infrastructure.soa.jdml.JobStatus">
  <void property="applicationData">
   <string>SoftwareModule.Install</string>
  </void>
  <void property="messageCode">
   <string>NO_CODE</string>
  </void>
  <void property="requestId">
   <string>b8121aa1033011dc855b000d6049368e</string>
  </void>
  <void property="status">
   <int>6</int>
  </void>
<void property="workItemStatus">
   <array class="com.ibm.tivoli.tpm.infrastructure.soa.jdml.WorkItemStatus" 
length="3">
```

The status of **6** correlates to JOB_STATE_COMPLETE_ALL in Table 11-17. Use this table to interpret the status results of any job recorded in the rcp.log file on the common agent.

Table 11-17 JobStatus reference codes

JobStatus	Code
JOB_STATE_QUEUED	1
JOB_STATE_SUBMITTED	2
JOB_STATE_CANCELLED_ALL	4
JOB_STATE_CANCELLED_PARTIAL	5
JOB_STATE_COMPLETE_ALL	6
JOB_STATE_COMPLETE_PARTIAL	7
JOB_STATE_FAIL	8
JOB_STATE_REJECTED	9

After the overall job status is reported in the XML results, each workItemStatus is reported as well. The last line of Example 11-48 shows that the workItemStatus is an array of `length=3`. This is as expected because Example 11-42 on page 415 showed that there were three WorkItems, so we can expect to have results returned for three WorkItems in the job results.

Example 11-49 shows the results for the first WorkItem LocalFileExist_10720 existence of the installable into the local cache.

Example 11-49 workItemStatus for LocalFileExist_10720

```
   <void index="0">
        <object
   class="com.ibm.tivoli.tpm.infrastructure.soa.jdml.WorkItemStatus">
         <void property="messageCode">
          <string>NO_CODE</string>
         </void>
         <void property="name">
          <string>LocalFileExist_10720</string>
         </void>
         <void property="result">
          <string>false</string>
         </void>
         <void property="status">
          <int>2</int>
         </void>
        </object>
       </void>
   <void index="1">
```

The status of the WorkItem LocalFileExist_10720 is **2**, which translates to WORK_ITEM_STATUS_COMPLETED from Table 11-18 on page 421. The next WorkItem status returned is for GetInstallable_10720 and is shown in Example 11-50.

Example 11-50 workItemStatus for GetInstallable_10720

```
<void index="1">
    <object class="com.ibm.tivoli.tpm.infrastructure.soa.jdml.WorkItemStatus">
      <void property="messageCode">
       <string>NO_CODE</string>
      </void>
      <void property="name">
       <string>GetInstallable_10720</string>
      </void>
      <void property="result">
       <string>1809</string>
      </void>
      <void property="status">
       <int>2</int>
      </void>
     </object>
    </void>
```

The status of the WorkItem GetInstallable_10720 is **2**, which translates to WORK_ITEM_STATUS_COMPLETED from Table 11-18. The final WorkItem status returned is for the actual install of the installable InstallInstallable_10720. Example 11-51 shows that this WorkItem also has a status of **2** or WORK_ITEM_STATUS_COMPLETED.

Example 11-51 workItemStatus for InstallInstallable_10720

```
<void index="2">
     <object
class="com.ibm.tivoli.tpm.infrastructure.soa.jdml.WorkItemStatus">
      <void property="messageCode">
       <string>NO_CODE</string>
      </void>
      <void property="name">
       <string>InstallInstallable_10720</string>
      </void>
      <void property="status">
       <int>2</int>
      </void>
     </object>
    </void>
```

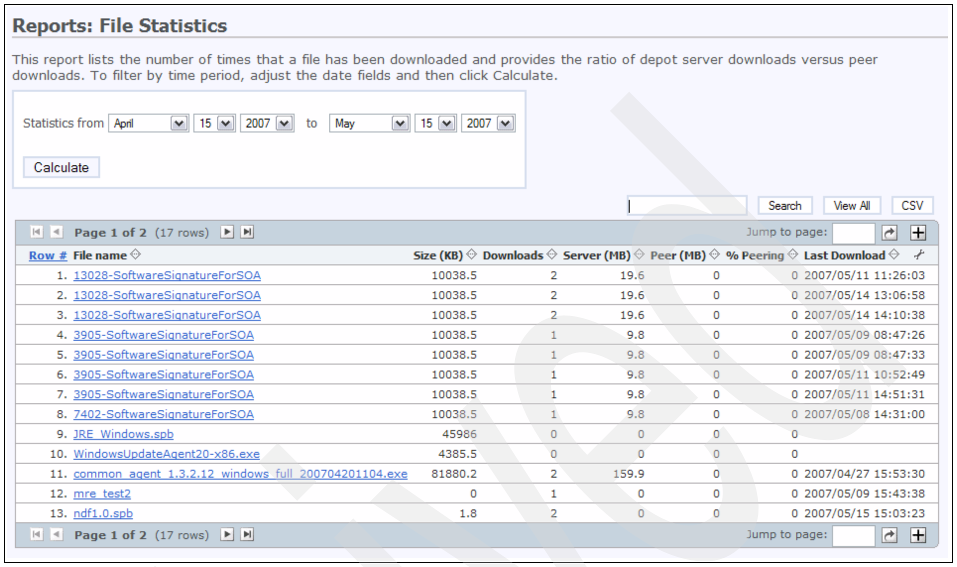
Use Table 11-18 to interpret the status results for WorkItems in the rcp.log file.

Table 11-18 WorkItemStatus reference codes

WorkItemStatus	Code
WORK_ITEM_STATUS_EXECUTING	1
WORK_ITEM_STATUS_COMPLETED	2
WORK_ITEM_STATUS_FAILED	3
WORK_ITEM_STATUS_CANCELLED	4
WORK_ITEM_STATUS_PENDING	5

We have downloaded and installed ndf1.0.spb on the common agent, which you can verify in the dynamic content delivery service user interface using the **Reports** → **File Statistics** panel.

Figure 11-26 shows the File Statistics panel and that the file ndf1.0.spb was downloaded two times and that the last download was at 2007/05/15 15:03:23.


Reports: File Statistics

This report lists the number of times that a file has been downloaded and provides the ratio of depot server downloads versus peer downloads. To filter by time period, adjust the date fields and then click Calculate.

Statistics from April 15 2007 to May 15 2007

Row #	File name	Size (KB)	Downloads	Server (MB)	Peer (MB)	% Peering	Last Download
1.	13028-SoftwareSignatureForSOA	10038.5	2	19.6	0	0	2007/05/11 11:26:03
2.	13028-SoftwareSignatureForSOA	10038.5	2	19.6	0	0	2007/05/14 13:06:58
3.	13028-SoftwareSignatureForSOA	10038.5	2	19.6	0	0	2007/05/14 14:10:38
4.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/09 08:47:26
5.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/09 08:47:33
6.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/11 10:52:49
7.	3905-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/11 14:51:31
8.	7402-SoftwareSignatureForSOA	10038.5	1	9.8	0	0	2007/05/08 14:31:00
9.	JRE_Windows.spb	45986	0	0	0	0	
10.	WindowsUpdateAgent20-x86.exe	4385.5	0	0	0	0	
11.	common_agent_1.3.2.12_windows_full_200704201104.exe	81880.2	2	159.9	0	0	2007/04/27 15:53:30
12.	mre_test2	0	1	0	0	0	2007/05/09 15:43:38
13.	ndf1.0.spb	1.8	2	0	0	0	2007/05/15 15:03:23

Figure 11-26 Report File Statistics panel

Device management service informs Tivoli Provisioning Manager server Job is complete

The final step to occur in the distribution of a software package to a common agent is for the device management service server to inform Tivoli Provisioning Manager that the job is complete. When this occurs, Tivoli Provisioning Manager also informs the dynamic content delivery service that it can remove the file from the depots unless it was previously published.

Note: Typically when we search the activityplan/console.log, we use the activity or thread name or the ID; however, occasionally there is other information in the logs that is relevant too. To find where the device management service updates Tivoli Provisioning Manager, perform the following:

```
cat console.log | awk -F\= '/Dms Status Count/ {if ($2>0) print $1,$2}'
```

Using the previous command returns the following line:

```
2007-05-15 17:26:27,164 DEBUG [Status Updater] (DmsService.java:256)
dms.DmsService: Dms Status Count=2
```

At this point, you can search for all ‘Status Updater’ lines in this time frame to find the complete set of information using a command such as:

```
cat console.log | grep “2007-05-15 17:2” | grep “Status Updater”
```

This command just narrows you down to the appropriate section and time frame. You could also find this section using an editor and by searching for the Package ID, which finds the following line:

```
2007-05-15 17:26:27,286 DEBUG [Status Updater] (FileManager.java:943)
cds.FileManager: Set file transfer status to DOWNLOADED for file 9139 with
packageId 1179266129194
```

This line also gets you to the correct section of the activityplan/console.log to find the lines that indicate that the device management service has updated Tivoli Provisioning Manager with the status.

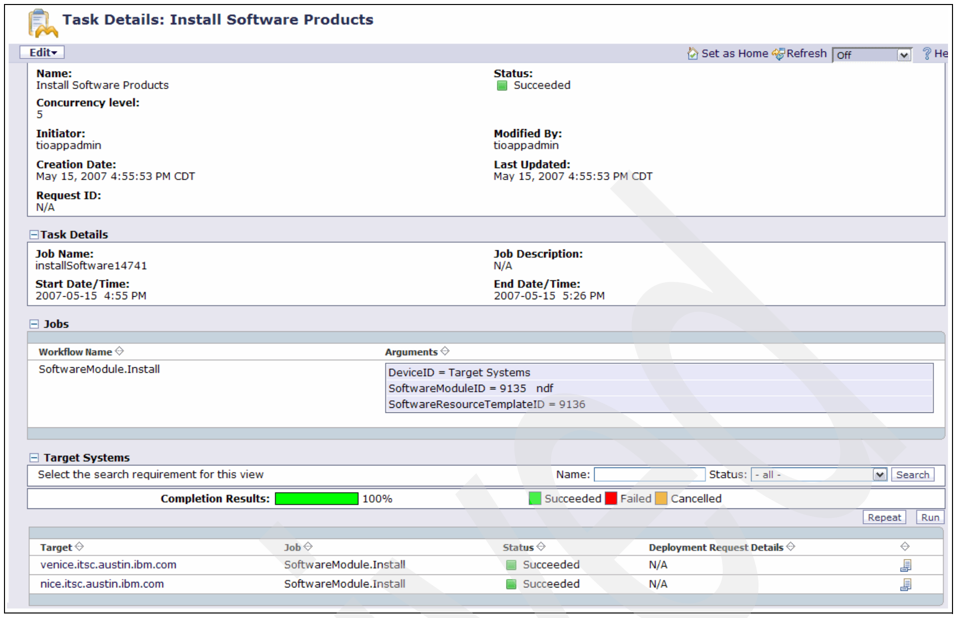
After you find the section of the log where the SOA Status Updater thread has a Dms Status Count that is more than zero, you can verify that the status was updated, and also see that the dynamic content delivery service file transfer status was set to DOWNLOADED. After this is complete, notice that the job was deleted. Example 11-52 shows this activity.

Example 11-52 activityplan/console.log

```
2007-05-15 17:26:26,843 DEBUG [Status Updater] (SoaStatusUpdater.java:59)
soa.SoaStatusUpdater: Thread[Status Updater,5,main] thread calling
ProcessJobStatus
2007-05-15 17:26:26,851 DEBUG [Status Updater]
(JobManagementServiceClient.java:307) client.JobManagementServiceClient: Using
endpoint URL: https://jiga.itsc.austin.ibm.com:9045/dmserver/
services/EDMSTPMManagementService
```

```
2007-05-15 17:26:27,164 DEBUG [Status Updater] (DmsService.java:256)
dms.DmsService: Dms Status Count=2
2007-05-15 17:26:27,278 DEBUG [Status Updater] (CdsManagementCenter.java:104)
cds.CdsManagementCenter: Found CDS Management Center with file repository id =
5400
2007-05-15 17:26:27,286 DEBUG [Status Updater] (FileManager.java:943)
cds.FileManager: Set file transfer status to DOWNLOADED for file 9139 with
packageId 1179266129194
2007-05-15 17:26:27,384 DEBUG [Status Updater] (FileManager.java:160)
cds.FileManager: URL is https://9.3.4.163:9045 username tioappadmin
2007-05-15 17:26:29,688 DEBUG [Status Updater] (FileManager.java:843)
cds.FileManager: Deleted file from CDS with packageId 1179266129194
2007-05-15 17:26:29,876 DEBUG [Status Updater] (DmsService.java:380)
dms.DmsService: Calling cleanup
2007-05-15 17:26:29,883 DEBUG [Status Updater]
(JobManagementServiceClient.java:307) client.JobManagementServiceClient: Using
endpoint URL:
https://jiga.itsc.austin.ibm.com:9045/dmserver/services/EDMSTPMManagementServic
e
2007-05-15 17:26:29,883 DEBUG [Status Updater] (DmsService.java:801)
dms.DmsService: calling deleteJobStatusByDateRange
2007-05-15 17:26:30,790 DEBUG [Status Updater] (DmsService.java:803)
dms.DmsService: cleanup successful
```

After Tivoli Provisioning Manager is updated, the user interface is updated as well. Figure 11-27 on page 425 shows that both venice.itsc.austin.ibm.com and nice.itsc.austin.ibm.com were successful.


Task Details: Install Software Products

Edit▾ | Set as Home | Refresh | Off | He

Name: Install Software Products
Status: 🟩 Succeeded
Concurrency level: 5
Initiator: tioappadmin
Modified By: tioappadmin
Creation Date: May 15, 2007 4:55:53 PM CDT
Last Updated: May 15, 2007 4:55:53 PM CDT
Request ID: N/A

Task Details

Job Name: installSoftware14741
Job Description: N/A
Start Date/Time: 2007-05-15 4:55 PM
End Date/Time: 2007-05-15 5:26 PM

Jobs

Workflow Name	Arguments

SoftwareModule.Install	DeviceID = Target Systems
	SoftwareModuleID = 9135 ndf
	SoftwareResourceTemplateID = 9136

Target Systems

Select the search requirement for this view   Name: Status: - all -   Search

Completion Results: 100%   🟩 Succeeded 🟥 Failed 🟧 Cancelled

Repeat   Run

Target	Job	Status	Deployment Request Details	
venice.itsc.austin.ibm.com	SoftwareModule.Install	🟩 Succeeded	N/A	
nice.itsc.austin.ibm.com	SoftwareModule.Install	🟩 Succeeded	N/A	

Figure 11-27 Task Details panel: final task status

11.6 Patch management

Tivoli Provisioning Manager for Software 5.1 Patch Management is based on the compliance and remediation mechanism. No activity plans are built, and it does not require a WSUS server as well.

Patches are discovered using the wsusscn2.cab file, and the executables are directly downloaded from the Microsoft site. For Microsoft update discovery problems, check the following logs:

- $TIO_LOGS/wsusscan.log
- $TIO_LOGS/deploymentengine/console.log

For errors related to receiving and updating inventory information related to patches, consult the following logs:

- $TIO_LOGS/dmsresultserver/console.log (for SDI)
- $TIO_LOGS/activityplanengine/console.log (for Tivoli Management Framework)

For errors related to the submission of installation/discovery operations, use the following steps:

1. Set trace and log file: open $TIO_HOME/config/log4j.prop file and double-check the following parameters:
   - section console.log:

     log4j.appender.consolefile.threshold=debug
     log4j.appender.consolefile.append=true
   - section error and trace log:

     log4j.appender.errorfile.threshold=error
     log4jappender.errorfile.append=true
2. Recreate the problem.
3. Export the workflow (for example, through the CLI):

   ```
   cd $TIO_HOME/tools
   ```

   ```
   workflowLogExport.cmd -nGroup_Status_Updater -fmyWorkflowExport.xml
   ```

   Hint: There is no space after the -n and no space after -f.
4. Collect the following documentation for problem determination:
   - $TIO_LOGS/deploymentengine/*.log
   - myWorkflowExport.xml (exported in step 3)

Abbreviations and acronyms

ACL	access control list
ADS	Advanced Deployment Services
APDE	Automation Package Development Environment
APE	Activity Plan Editor
APM	Activity Plan Manager
CAS	Common Agent Services
CCMDB	Change and Configuration Management Database
CDS	dynamic content delivery services
CIT	Common Inventory Technology
CSV	comma separated value
CVT	Component Verification and Test
DCM	data center model
DE	Deployment Engine
DHCP	Dynamic Host Configuration Protocol
DMS	device management service
DMZ	dematerialized zone
DNS	Domain Name System
ETL	Extract, Transform and Load
FFDC	First Failure Data Capture
FR	File Repository
IBM	International Business Machines Corporation
ISMP	InstallShield MultiPlatform
ITSO	International Technical Support Organization
JCF	Java Client Framework
JES	Job Execution Service
JVM	Java Virtual Machine
KDE	K Desktop Environment
LCF	Lightweight Client Framework
LDO	logical device operation
LWI	Lightweight Infrastructure
MDist2	Multiplex Distribution 2
MC	Management Center
MIF	Management Information Formats
MSAD	Microsoft Active Directory
NIS	Network Information Service
OAMP	Operations, Administration, Maintenance and Provisioning
OAMPi	Operations, Administration, Maintenance and Provisioning infrastructure
OID	object identifier
OPAL	Open Process Automation Library
OSGi	Open Services Gateway Initiative
PXE	Pre-boot eXecution Environment
RDBMS	Relational Data Base Management Systems
RDM	Remote Deployment Manager
RIM	RDBMS Interface Module
RXA	Remote Execution Access
SAP	service access point
SCM	Security Compliance Manager
SDI	Scalable Distribution Infrastructure
SMB	Service Message Block

SOA	Service-Oriented Architecture
SP	software package
SPB	software package block
SPE	Software Package Editor
SPO	SoftwarePackage object
SRT	Software Resource Template
SSH	Secure Shell
SSL	Secure Sockets Layer
TCA	Tivoli Common Agent
TEC	Tivoli Enterprise Console
TSFT	Tivoli Security Firewall Toolkit
TIL	Topology Installer Launcher
TMA	Tivoli management agent
TSFT	Tivoli Security Firewall Toolkit
UDP	user datagram protocol
WSRF	Web Services Resource Framework
WSUS	Windows Server Update Services
WUA	Windows Update Agent

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get Redbooks" on page 430. Note that some of the documents referenced here may be available in softcopy only.

- *Deployment Guide Series: IBM Tivoli Provisioning Manager version 5.1*, SG24-7261.

Other publications

These publications are also relevant as further information sources:

- *Tivoli Provisioning Manager for Software Problem Determination and Troubleshooting Guide, Version 5.1*, SC32-2256
- *Tivoli Management Framework Reference Guide, Version 4.1.1,* SC32-0806
- *Tivoli Management Framework User's Guide V4.1.1,* GC32-0805
- *IBM Tivoli Configuration Manager Patch Management Guide*, SC23-5263

Online resources

These Web sites are also relevant as further information sources:

- Tivoli Provisioning Manager information center Web site:

  `http://tioid2.torolab.ibm.com:8884/help/index.jsp`
- Microsoft Web site:

  `http://go.microsoft.com/fwlink`
- Web site to download the cabextract utility for AIX:

  `http://aixpdslib.seas.ucla.edu/packages/cabextract.html`

- Web site to download the Tivoli Management Framework V4.1.1 Fix Pack 6:

  `ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP06`

- Web site to download the Tivoli Configuration Manager V4.2.3 Fix Pack 04

  `ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.2.3/4.2.3-TIV-TCM-FP0004`

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software Coexistence and Migration Considerations

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

IBM Tivoli Configuration Manager and Tivoli Provisioning Manager for Software Coexistence and Migration Considerations

Learn how to deploy a coexistent environment

Learn about the combined product environment

Learn how to migrate to Tivoli Provisioning Manager and more

This IBM® Redbooks® publication focuses on migration of a data center from a Tivoli Framework environment and a Tivoli® Configuration Manager-based management model to the new Service-Oriented Architecture (SOA), provided by Tivoli Provisioning Manager for Software V 5.1. In addition to migration, we also discuss the coexistence environment, where Tivoli Provisioning Manager for Software drives the Tivoli Management Framework environment.

To give you ideas for various migration possibilities, we describe several customer environments and then describe the suggested approach for moving to a Tivoli Provisioning Manager environment. We also tell you how the two environments can coexist during the migration process. We provide a feature-by-feature comparison between common Frameworks, Tivoli Configuration Manager operations, and their equivalent in Tivoli Provisioning Manager.

This book is a reference for IT Specialists who implement data center migration from a Tivoli Management Framework and Tivoli Configuration Manager-based management model to Tivoli Provisioning Manager for Software V 5.1 or Tivoli Provisioning Manager V 5.1.

SG24-7426-00 ISBN 0738489204