Installing Tivoli Kernel Services

Version 1.1

Installing Tivoli Kernel Services

Version 1.1

Installing Tivoli Kernel Services

Copyright Notice

Trademarks

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

This product includes software developed by The Apache Group for use in the Apache HTTP Server project (http://www.apache.org/).

This product includes software developed by Greg Stein <gstein@lyra.org> for use in the mod_dav module for Apache (http://www.webdav.org/mod_dav/).

This product includes software derived from software developed by Henry Spencer.

This product includes software derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

Portions of certain components of the Program are copyrighted by MERANT, 1991-1999.

March 15, 2001

Contents

Preface

Installing Tivoli Kernel Services provides introductions to and explanations of installing the installation depot, bootprint servers, and specialized servers associated with Tivoli Kernel Services. This document uses a task-oriented approach to explain the installation process.

Who Should Read This Document

The target audience for this document is system administrators responsible for the installation of Tivoli Kernel Services. Users of this document should have knowledge of the following:

- Microsoft Windows and UNIX operating systems
- Database architecture and concepts
- Graphical user interfaces

Prerequisite and Related Documents

The documentation for Tivoli Kernel Services, with the exception of the integrated online help and the product README, is located in the tivolidocs subdirectory on both the Tivoli Kernel Services Installation CD-ROM and the Bootprint CD-ROM, in both HTML and PDF formats.

- *Introducing Tivoli Kernel Services Administration*

  Provides information about administering Tivoli Kernel Services through the use of the Tivoli Console.

- *Planning for Tivoli Kernel Services*

  Provides information about planning for Tivoli Kernel Services.

- *Tivoli Kernel Services Command Reference*

  Provides information about the command line interface to Tivoli Kernel Services.

- *Tivoli Kernel Services README*

Located in the root directory of the installation CD-ROM, this HTML document provides late-breaking information, such as problems and workarounds and patch availability.

- *Tivoli Kernel Services Command Quick Reference Card*

  A quick reference showing a summary of the command line interface for Tivoli Kernel Services

- *Tivoli Kernel Services Online Help*

  Integrated online help for all Tivoli Kernel Services administrative tasks. All information about performing Tivoli Kernel Services tasks are documented only in the Tivoli Assistant. When new products are installed that run under the Tivoli Console, corresponding Tivoli Assistant help topics are also installed and integrated into the existing information base.

- *Guided Tour of the Tivoli Console*

  An HTML-based introduction to the Tivoli Console, the graphical user interface used to administer Tivoli Kernel Services.

IBM Universal DB2 Enterprise Edition Information

The publications required to support IBM DB2 are available in HTML format on the IBM DB2 Universal Database Enterprise Edition CD-ROM for specific platforms included in the Tivoli Kernel Services package or from this IBM Web site: **http://www.ibm.com/software/data/db2/udb/**.

What This Document Contains

The *Installing Tivoli Kernel Services* document contains the following sections:

- "Introducing Tivoli Kernel Services" on page 1

  Provides an overview of Tivoli Kernel Services and describes the installation process.

- "Preparing to Install the Installation Depot" on page 7

  Provides information on the options available for installing the installation depot and describes the prerequisites that need to be met before installing Tivoli Kernel Services.

- "Installing the Installation Depot" on page 19

  Provides step-by-step instructions for using the installation program to install the installation depot. The various install options, including simple install, advanced install, and specialized server install, are described in detail.

- "Installing a Bootprint" on page 37

  Provides step-by-step instructions for using the bootprint installation program to install a bootprint server.

- "Running Tivoli Kernel Services" on page 45

  Provides information for running the installation depot, improving performance, and tailoring its operation to your needs.

- "Uninstalling Tivoli Kernel Services" on page 103

  Provides detailed instructions for uninstalling Tivoli Kernel Services.

Conventions Used in This Document

The document uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold	Commands, keywords, file names, authorization roles, URLs, or other information that you must use literally appear like **this**, in **bold**. Names of windows, dialogs, and other controls also appear like **this**, in **bold**.
Italics	Variables and values that you must provide appear like *this*, in *italics*. Words and phrases that are emphasized also appear like *this*, in *italics*.
`Monospace`	Code examples, output, system messages, XML tags, and Java class, method, and interface names appear like `this`, in a `monospace` font.

This guide uses the UNIX convention for specifying environment variables and for directory notation. When using the Windows NT

command line, replace $*variable* with *%variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths.

Note: When using the bash shell on a Windows NT system, you can use the UNIX conventions.

Accessing Publications Online

The Tivoli Customer Support Web site (**http://www.tivoli.com/support/**) offers a guide to support services (the *Customer Support Handbook*); frequently asked questions (FAQs); and technical information, including release notes, user's guides, redbooks, and white papers. You can access Tivoli publications online at **http://www.tivoli.com/support/documents/**. The documentation for some products is available in PDF and HTML formats. Translated documents are also available for some products.

To access most of the documentation, you need an ID and a password. To obtain an ID for use on the support Web site, go to **http://www.tivoli.com/support/getting/**.

Resellers should refer to **http://www.tivoli.com/support/smb/index.html** for more information about obtaining Tivoli technical documentation and support.

Business Partners should refer to "Ordering Publications" on page xvi for more information about obtaining Tivoli technical documentation.

Ordering Publications

Order Tivoli publications online at **http://www.tivoli.com/support/Prodman/html/pub_order.html** or by calling one of the following telephone numbers:

- U.S. customers: **(800) 879-2755**

- Canadian customers: **(800) 426-4968**

Providing Feedback about Publications

We are very interested in hearing about your experience with Tivoli products and documentation, and we welcome your suggestions for improvements. If you have comments or suggestions about our products and documentation, contact us in one of the following ways:

- Send e-mail to **pubs@tivoli.com**.
- Fill out our customer feedback survey at **http://www.tivoli.com/support/survey/**.

Contacting Customer Support

If you need support for this or any Tivoli product, contact Tivoli Customer Support in one of the following ways:

- Submit a problem management record (PMR) electronically from our Web site at **http://www.tivoli.com/support/reporting/**. For information about obtaining support through the Tivoli Customer Support Web site, go to **http://www.tivoli.com/support/getting/**.
- Submit a PMR electronically through the IBMLink™ system. For information about IBMLink registration and access, refer to the IBM Web page at **http://www.ibmlink.ibm.com**.
- Send e-mail to **support@tivoli.com**.
- Customers in the U.S. can call **1-800-TIVOLI8** (**1-800-848-6548**).
- Customers outside the U.S. should refer to the Tivoli Customer Support Web site at **http://www.tivoli.com/support/locations.html** for customer support telephone numbers.

When you contact Tivoli Customer Support, be prepared to provide the customer number for your company so that support personnel can assist you more readily.

Additional Notices

OpenSSL License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: ″This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)″
4. The names ″OpenSSL Toolkit″ and ″OpenSSL Project″ must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called ″OpenSSL″ nor may ″OpenSSL″ appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment: ″This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)″

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ″AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape's SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: ″This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).″ The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related:
4. If you include any Windows-specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: ″This product includes software written by Tim Hudson (tjh@cryptsoft.com)″

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ″AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License.]

Apache and Apache Tomcat Notices

This product includes software developed by The Apache Group for use in the Apache HTTP Server project (http://www.apache.org/).

This product includes software developed by Greg Stein <gstein@lyra.org> for use in the mod_dav module for Apache (http://www.webdav.org/mod_dav/).

This product includes software derived from software developed by Henry Spencer.

This product includes software derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

Portions of certain components of the Program are copyrighted by MERANT, 1991-1999.

1

Introducing Tivoli Kernel Services

This section provides an overview of Tivoli Kernel Services and the installation process. For a more detailed discussion of the concepts behind Tivoli Kernel Services and for information on planning and deployment, see the *Planning for Tivoli Kernel Services* document.

The Marketing View: A Highly Scalable and Highly Available Distributed Systems Environment

Tivoli Kernel Services provides applications with a robust set of components and functions for managing a wide variety of users, devices, and services. Built on common communications, messaging, data storage, and logging services, and enhanced with security and a new role-based user interface, Tivoli Kernel Services is designed to be both highly scalable and highly available.

Much like an operating system, Tivoli Kernel Services provides the following major functions:

- Base services, such as communications, data storage, logging, and security
- Device management, for SNMP and storage area network devices
- The new Tivoli Console, a role-based, intuitive graphical user interface built on Tivoli Presentation Services.

The Physical View: A Collection of Servers

For the purpose of understanding the installation process, think of a Tivoli Kernel Services installation as a set of powerful server machines whose functions are determined by the software that you install and deploy using Tivoli Kernel Services and the applications that are running on it.

In a Tivoli Kernel Services installation, servers are defined by which CD-ROM in the Tivoli Kernel Services package was used to set up the machine and by which components are deployed to that machine. Servers in an installation can be defined as:

- The installation depot
- Tivoli Kernel Services servers
- Specialized servers, which include the database server, the Tivoli Data Access Services (DAS) server, and the MQSeries server

As the system installer, you need to deploy these servers across your installation based on the information in the Planning Worksheet which you completed as part of the planning process outlined in the *Planning for Tivoli Kernel Services* document. If you have not yet completed the Planning Worksheet, please do so before starting the installation.

The Installation Depot

One of the servers in your installation is designated as the installation depot, or ID. The installation depot is probably the largest and most powerful machine in your installation and is usually located near the geographical center of your enterprise.

The installation CD-ROM is used to launch the installation depot install program, which prepares the ID machine to deploy Tivoli Kernel Services code to other servers in your installation.

Tivoli Kernel Services Servers

The installation depot is the most important server because it is the repository for all the components of Tivoli Kernel Services. But unless your installation is very small, this one server will not meet your needs.

To provide more capacity and allow for workload balancing, you can define other machines in the installation as Tivoli Kernel Services servers. These servers are initialized with a small piece of software, called the bootprint, which provides the basis for establishing the distributed computing environment of Tivoli Kernel Services.

The bootprint CD-ROM is used to launch the bootprint installation program, which prepares the Tivoli Kernel Services server to receive software from the installation depot. The installation depot must be operational and the Tivoli Kernel Services server must have connectivity to the installation depot both during the installation process and later, during normal operation.

There are three types of bootprints:

Tivoli Kernel Services server
: Tivoli Kernel Services servers are resources that have been dedicated to run Tivoli Kernel Services code. Tivoli Kernel Servers servers can be defined as gateways or general purpose servers.

Tivoli Kernel Services endpoint server
: Tivoli Kernel Services endpoint servers are resources that have been enabled to run Tivoli Kernel Services application software. Refer to the documentation provided with your application to determine when endpoint servers should be installed.

Tivoli Console server
: Tivoli Console servers are resources that have been enabled to run the Tivoli Console for administering Tivoli Kernel Services and its applications.

Specialized Servers

A number of other servers, which may or may not be installed on the installation depot machine depending on the install options selected, are needed for Tivoli Kernel Services.

In all cases a database server is needed to store configuration and security information for the installation. In this release of Tivoli Kernel Services, this is DB2 Universal Database Version 6.1 Enterprise Edition or DB2 Universal Database Version 7.1. You must manually install the database server prior to installing Tivoli Kernel Services on the installation depot.

To communicate with the database server, a Tivoli Data Access Services (DAS) server is used. This server must be installed on the same machine as the database server.

To provide messaging support across multiple servers, one or more MQSeries servers are used. An MQSeries server can be installed on the installation depot machine or on another machine in the installation.

The Logical View: A Collection of ORBs

Instead of focusing on the physical structure of Tivoli Kernel Services, another way to view things is from the perspective of a distributed management environment. The actual work in the distributed system is performed by object request brokers.

An object request broker, or ORB, manages interactions between clients and servers. A client running on one ORB makes a request for a function. That function is performed by a server, which might be another client running on the same ORB, a client running on a different ORB on the same physical machine, or even a client running on a different physical machine. The results are subsequently returned to the client. The client generally does not care where the operation was performed.

The ORB used in Tivoli Kernel Services is Java-based and CORBA compliant, and supports industry-standard protocols such as SSL,

JTA, and JNDI. Each ORB runs in its own Java virtual machine enabling the dynamic update of components running on an ORB without having to bring down the ORB or the entire distributed system.

The Installation Depot

The installation depot consists of two ORBs:

Installation depot ORB
: Provides the base functions of the installation depot, which includes things such as managing the deployment of components to other ORBs and handling database interactions.

Tivoli Presentation Services ORB
: Provides the functions for the Tivoli Console, the graphical user interface used for Tivoli Kernel Services Administration. Also referred to as the workstation-based Java implementation of the Tivoli Console.

The installation depot ORB and the Presentation Services ORB can be started automatically by the installation program, but **only** if you request that they be when you are prompted during the install process.

Tivoli Kernel Services Server

Each Tivoli Kernel Services server consists of a single ORB. The bootprint ORB communicates with the installation depot ORB to receive and deploy any required software.

2

Preparing to Install the Installation Depot

This section describes the two ways to install the installation depot and describes how to set up the DB2 database server before running the installation program.

Special Notice

This document should be used in conjunction with the Tivoli management software you have purchased. Refer to the documentation provided with that Tivoli management software before using the steps outlined in this document to install Tivoli Kernel Services

Types of Installations

There are two ways to install the installation depot:

Single machine install

Single machine install automatically puts all the specialized servers, with the exception of the database server, on the installation depot machine. This provides a quick and straightforward way to create an installation depot with a minimum amount of planning and preparation, but is only practical for small or specialized installations.

Multiple machine install

Multiple machine install allows you to decide where you want to place the specialized servers in your enterprise and gives you almost complete control in deciding how to lay out your installation.

Understanding the Single Machine Install

When you run the single machine install, the installation program performs the following tasks:

- Creates database tables of the default size with the default names on the DB2 server as follows:

slash	Directory services database
tmd	Security services database
ps	Tivoli Presentation services database
logging	Logging database
secaudit	Security audit database

  If databases already exist with these names, they are deleted and re-created.

- Installs and configures the Tivoli Data Access Services (DAS) server on the machine.
- Installs and configures an MQSeries server on the machine.
- Installs the Tivoli Kernel Services software on the installation depot machine, which includes:
  - Tivoli Kernel Services
  - Tivoli Presentation Services
  - IBM HTTP Server Powered by Apache
  - Java runtime environment
- Starts the installation depot ORB.

Note: If you run the single machine install again, it deletes and then re-creates the DB2 database tables. This effectively resets the

configuration and security information to the default values and results in the loss of any stored data.

Understanding the Multiple Machine Install

When you run the multiple machine install, the installation program performs a similar set of tasks, but with the added flexibility of allowing you to choose how things are deployed and configured.

The multiple machine install does the following:

- Installs the Tivoli Data Access Services (DAS) Server on any machine. A Tivoli DAS Server must be installed on every DB2 machine.
- Installs the MQSeries server on any machine.
- Installs the installation depot software on the machine, permitting you to:
  - Set a name for the installation depot and specify the port number to use for communications.
  - Specify the name for the directory database, indicate on what server it resides, what port number to use for communications, and the DB2 user ID and password to use for access.
  - Specify similar database parameters for the security information
  - Specify similar database parameters for the Tivoli Console information
  - Specify the name of the MQSeries server and the port number to use for communications
  - Specify the port numbers for the Java implementation of the Tivoli Console
  - Specify whether none, one, or both ORBs on the installation depot should be started automatically after the installation program completes.

One major difference between single and multiple machine install is that during a multiple machine install, the databases needed by Tivoli Kernel Services are *not* created automatically by the installation program. See "Creating Tivoli Kernel Services Databases" on page 14 for information on creating the databases before running the installation program.

Installing the Database Server

Before you do either a single or multiple machine install of the Tivoli Kernel Services installation depot, you must install a database server. The database servers supported in this release is DB2 Universal Database Version 6.1 Enterprise Edition and DB2 Universal Database Version 7.1.

If you will be doing a single machine install, the database server must be installed on the installation depot machine. Otherwise, the database server can be installed on any suitable machine that has network connectivity to the installation depot machine. The database server must only be used by one installation depot.

Note: If you are installing the DB2 database server and the Tivoli DAS server on an IBM AIX system, review "Database Connectivity Considerations on AIX" on page 118 before proceeding.

For information on installing and using DB2, refer to the HTML documentation provided on the DB2 CD-ROM. Printable documentation is available at the following Web site (all on one line):

```
http://www.ibm.com/cgi-bin/db2www/data/db2/udb/
     winos2unix/support/v6pubs.d2w/admin
```

DB2 Installation on Windows Systems

If you are not experienced in installing or using DB2, you can use the following procedure to install a DB2 server on a Windows system for use by Tivoli Kernel Services.

First, log on with a user ID with Administrator authority to run the installation program. Then locate the DB2 for Windows CD-ROM and insert it into the CD-ROM drive.

The installation program will usually auto-start and take you through a series of panels. Follow the prompts and take the defaults for options, communications ports, and directories. Select **Typical Install** when prompted.

The default DB2 user ID that is created is **db2admin** with a corresponding password of **db2admin**. You can use this or change it, if desired. Be sure to record the user ID and password you selected on the Planning Worksheet for use later during the installation depot installation program.

After the installation program completes, a DB2 server will be running on your Windows system. The DB2 database server is set up to automatically start when Windows starts.

To verify that the DB2 server is running, open the DB2 Control Center using the DB2 user ID and password you created.

DB2 Installation on Solaris Systems

If you are not experienced in installing or using DB2, you can use the following procedure to install a DB2 server on a Solaris system for use by Tivoli Kernel Services.

1. Log into the Solaris system as **root**.
2. Tune the Solaris kernel for DB2.

   Edit the `/etc/system` file and append these kernel configuration parameters to the end:

```
set msgsys:msginfo_msgmax=65535
set msgsys:msginfo_msgmnb=65535
set msgsys:msginfo_msgmap=258
set msgsys:msginfo_msgmni=256
set msgsys:msginfo_msgssz=16
set msgsys:msginfo_msgtql=1024
set msgsys:msginfo_msgseg=32768
set shmsys:shminfo_shmseg=16
set shmsys:shminfo_shmmni=300
set semsys:seminfo_semume=50
```

Depending on the amount of physical memory on your database server, append the appropriate kernel configuration parameters to the file:

256-512MB

```
set semsys:seminfo_semmni=512
set semsys:seminfo_semmap=514
set semsys:seminfo_semmns=1024
set semsys:seminfo_semmnu=1024
```

> 512MB

```
set semsys:seminfo_semmni=1024
set semsys:seminfo_semmap=1026
set semsys:seminfo_semmns=2048
set semsys:seminfo_semmnu=2048
```

Determine the setting for `semsys:shminfo_shmmax` as follows:

- Calculate 90% of the physical memory in bytes. For example, if the database server has 768MB of storage:

  ```
  768 x 0.90 x 1024 x 1024 = 724775731 bytes
  ```

- Compare the number you just calculated with these numbers, based on physical memory size:

256-512MB	268435456
> 512MB	536870912

- Set the higher value in the `/etc/system` file, such as:

  ```
  set shmsys:shminfo_shmmax=724775731
  ```

3. Create a group called **db2admin** and a user called **db2admin**. Due to a DB2 restriction, the user ID must be 8 characters or less in length. A user ID longer than 8 characters, such as ″Administrator″, will not work because the DB2 validation will look for a user ID called ″Administ″ and fail.

   Record the user ID and password on the Planning Worksheet for later use during the installation depot installation program.

   Note: The filesystem selected for the home directory of this user is used to store the databases. Ensure that the filesystem has enough space for the Tivoli Kernel Services databases. A rough estimate is approximately 150 MB.

4. Locate the DB2 for Solaris CD-ROM and insert it into the CD-ROM drive.
5. Start the DB2 installer, `db2setup`
6. Select **DB2 UDB Enterprise Edition** (the DB2 Universal Database server) as the component to install.
7. Do **not** create the DB2 instance when prompted to do so during the install.

After the DB2 installation program completes, create a DB2 instance called **db2admin** as follows:

1. Log into the Solaris system as **root**.
2. Add the following two lines to the `/etc/services` file to create a connection port and interrupt port for the DB2 instance called **db2**:

   ```
   db2c 50001/tcp
   db2i 50002/tcp
   ```

3. Locate the `db2icrt` command in the *DB2installdirectory*`/instance` directory.
4. Create a DB2 instance with the following command, substituting **db2admin** for *DB2user*, and **db2admin** for *DB2InstanceName*:

   ```
   db2icrt -u DB2user -a server DB2InstanceName
   ```

   You might see an error accessing the `/etc/rc.db2` file while running this command. This error can be ignored.
5. Log into the Solaris system as the DB2 user.
6. Enter the following commands as the DB2 user, substituting **db2c** for *DB2connection* and **db2admin** for *DB2InstanceName*:

   ```
   db2 update database manager configuration using svcename DB2connection
   db2set -i DB2InstanceName db2comm=tcpip
   ```

7. Logon as root and locate the `db2iset` command in the *DB2installdirectory*`/instance` directory.
8. Enter the following command to start the DB2 instance automatically when the database server boots, substituting **db2admin** for *DB2InstanceName*:

```
db2iset DB2AUTOSTART=TRUE -i DB2InstanceName
```

Useful DB2 Commands

Here is a set of DB2 commands you might find useful:

Operation	Command
Start DB2	db2start
Stop DB2	db2stop
List databases	list database directory
List tables	list tables
Query tables	select * from *table*
List all registered nodes on client	list node directory
Connect to a server called *DBserver*	attach to *DBserver* user *DB2user* using *DB2pw*
Connect to a database called *DBname*	connect to *DBname* user *DB2user* using *DB2pw*
List the configuration of a database	db2 get db cfg for *DBname*
List the configuration for the server	db2 get dbm cfg
Increase the transaction log size	db2 update db cfg for *DBname* using LOGPRIMARY *newvalue*

Creating Tivoli Kernel Services Databases

If you intend to use the multiple machine install option of the installation depot install, the appropriate databases and database schemas needed by Tivoli Kernel Services must be created on the database server.

Note: If you intend to use the single machine install option, skip the remainder of this section, as the installation program will create the necessary databases for you automatically.

To create the necessary databases and database schemas, Tivoli Kernel Services supplies a script called **tdacl**. There are two versions of the script: one for UNIX systems, and one for Microsoft Windows systems.

- The **tdacl.sh** script, provided for UNIX systems, is interactive in nature and allows you to select the names of the databases, whether old databases should be deleted, and so on.
- A limited-function batch file called **tdacl.bat** is provided for Windows systems. The batch file is non-interactive in nature and its behavior can be modified only by editing the batch file. The batch file deletes existing databases and then creates them using the default names and options. The function of the batch file is the same as running **tdacl.sh** on UNIX and taking all the defaults.

Before Creating the Databases

Before running **tdacl** to create the databases and database schemas, do the following:

1. Commit all pending database transactions and exit any application programs that are accessing the database server. This helps to minimize the risk of an interruption to the database server or any loss of data. Be certain that no batch or background jobs are running which could be accessing the database. You also might want to back up your important DB2 information before proceeding.
2. Log on to the database server with a DB2 user ID that has the authority to delete and create database tables. Due to a DB2 restriction, the user ID must be 8 characters or less in length. A user ID longer than 8 characters, such as ″Administrator″, will not work because the DB2 validation will look for a user ID called ″Administ″ and fail.
3. Create the proper environment for issuing DB2 commands.

   Windows Open a DB2 command window from the **Start** menu or by opening a Windows command prompt and entering `db2cmd`.

   UNIX Source your DB2 profile.

4. Locate the appropriate **tdacl** script for your database server in the **util** directory on the installation CD-ROM.

By default, the **tdacl** script creates the databases in the native code page of your DB2 database server. The native code page is determined by the locale and territory settings on the machine. If you wish to have the Tivoli Kernel Services databases created with a different code page, set the DB2CODEPAGE environment variable before running **tdacl**.

In a multilingual environment, you must set the code page to the multilingual UTF-8 code page. This is accomplished by setting the DB2CODEPAGE environment variable to 1208.

Creating Databases on UNIX Systems

On UNIX systems, locate the **tdacl.sh** script, which can be found in the `util` directory of the installation CD-ROM and run it.

The **tdacl.sh** script is interactive in nature. After entering your DB2 user ID and password, the script will ask a series of questions for each of the five databases that can be created:

- Directory services database
- Security registry database
- Tivoli Presentation Services database
- Logging database
- Security audit database

1. Enter **y** to install the database or **n** if you do not want to do so. (Entering **n** bypasses further processing on this database and begins processing the next one in the sequence.)
2. Enter the name for the database.

   Security registry database TMD

   Directory services database SLASH

   Tivoli Presentation Services database PS

Logging database	LOGGING
Security audit database	SECAUDIT

Note: DB2 database names must be 8 characters or less in length. The **tdacl.sh** script does not validate the length of the database name.

3. Enter **y** to drop the old version of the database or **n** to keep it. When you choose to drop an existing database, a ***drop database*** command is issued on your behalf.
4. Enter **y** to create the database or **n** to bypass creating it.
5. Enter **y** to create the database schema or **n** to bypass creating it.

At the end of processing the databases, the script will ask whether you want to perform tuning to optimize the performance of the databases. Enter **y** to perform the specified tuning or **n** to bypass the tuning.

Note: Tivoli Kernel Services might not function properly if your databases have not been properly tuned. For more information about tuning databases. see "Tuning Databases for Better Performance" on page 48.

Creating Databases on Windows Systems

On Windows systems, locate the **tdacl.bat** script, which can be found in the `util` directory of the installation CD-ROM.

If you want to specify different names for the databases, or otherwise modify the behavior of the script, make a copy of it and edit it to reflect your choices.

Note: If you wish to use different names, remember that DB2 database names must be 8 characters or less in length.

The **tdacl.bat** script takes two required parameters:

```
tdacl.bat DB2userID DB2password
```

where:

DB2userID is your DB2 user ID

DB2password is the corresponding password

The script will then run and create the Tivoli Kernel Services databases.

3

Installing the Installation Depot

This section contains the information you need to install a Tivoli Data Access Services (DAS) server, an MQSeries server, and the Tivoli Kernel Services installation depot.

Before You Begin

Before you run the installation program, be sure you have completed the steps described in "Preparing to Install the Installation Depot" on page 7. Also be sure that you have the completed Planning Worksheet from the *Planning for Tivoli Kernel Services* book. The Planning Worksheet allows you to quickly locate and subsequently enter the information required during the installation process.

Refer to the documentation provided with the application you purchased for additional installation information.

Required Authority for the Installation

The installation program must be run as **root** on UNIX systems, or from a user ID with **Administrator** authority on Microsoft Windows 2000 systems. Users must be logged on locally and not to a domain.

Note: If you plan to do a single machine installation, the user ID used for the installation must be 8 characters or less in length due to a DB2 restriction and be authorized to issue DB2 commands. Tivoli recommends using the DB2 user ID created

in "Preparing to Install the Installation Depot" on page 7, which already has Administrator authority and sufficient DB2 access authority.

The installation program copies files from the installation CD-ROM to the local disk of the installation depot machine. The user ID running the installation program must have read/write access to the destination directory or file system for the install. In addition, read/write access is also needed to the following temporary locations:

UNIX

```
/usr/local
/etc
```

Microsoft Windows 2000

C:\

On Microsoft Windows 2000 systems, Tivoli Kernel Services also creates a special Windows user ID called **tksUser**. This user ID is used to launch ORBs and must not be modified or deleted.

Checking the Environment

You should verify that all of the following conditions are true (or not applicable) before starting the installation program. The conditions are explained in more detail on the following pages.

Condition	Met or not applicable?
Sufficient temporary disk space	
UNIX system prerequisities met	
Computer name fully qualified on Windows 2000 systems	
Database server installed	
Database server running	
DB2 profile sourced on UNIX systems with Tivoli DAS server	
Proper level of Internet Explorer on Windows NT machines with an MQSeries server	

Condition	Met or not applicable?
Windows Explorer is not open to any of the Tivoli Kernel Services directories.	
Existing HTTP (World Wide Web) servers stopped and set to not run automatically.	
Network connectivity	

Sufficient temporary disk space

The machine on which you are installing the installation depot meets the storage and memory requirements described in the Planning for Tivoli Kernel Services document. In addition, at least 60 MB of temporary storage must also be available.

On UNIX systems, this space must be available in both the `/tmp` and `/var` directories. On Windows systems, this space must be available in the directory specified in the TEMP environment variable.

UNIX system prerequisities met

If installing on a UNIX system, be sure to meet the requirements outlined in the *Planning for Tivoli Kernel Services* document.

Computer name is fully qualified on Windows 2000 systems

If installing on Microsoft Windows 2000, ensure that the computer name is fully qualified. To verify this, right click **My Network Places** and select **Properties**. Go to the **Advanced** ->**Network Identification** section. If the **Full computer name** field is not fully qualified with a domain name, click **Properties** and **More...**, then enter the domain name for the machine.

The machine will need to be rebooted for this change to take effect.

Database server installed

The machine on which you are installing the installation depot either has a database server installed on it, or has

network connectivity to an existing database server machine. The **ping** command can be used to verify network connectivity.

Database server running

The database server must be running and accepting database transactions. You can verify this by starting the DB2 Control Center, logging on using your DB2 user ID and password, and verifying that your DB2 instance is running.

DB2 profile sourced on UNIX systems with Tivoli DAS server

If installing the Tivoli DAS server on a UNIX system, be sure to source the DB2 profile before running the installation program.

Proper level of Internet Explorer on Windows NT machines with an MQSeries server

If installing an MQSeries server on a Microsoft Windows NT 4.0 system, verify that you have installed either Microsoft Internet Explorer 4.01 with Service Pack 1, or a higher version of Microsoft Internet Explorer.

Windows Explorer is not open to any Tivoli Kernel Services directories

When installing on Microsoft Windows machines, make sure that the Windows Explorer is not open to any of the Tivoli Kernel Services directories. You may use the Windows Explorer to start the installation, but it does not need to stay open.

Existing HTTP (World Wide Web) servers stopped and set not to run automatically

Tivoli Kernel Services installs an HTTP server on the installation depot machine. Be sure to stop any existing HTTP or World Wide Web publishing servers and set them to run manually so that they do not interfere with the operation of the installation depot.

Network connectivity

In the case of a multiple machine install, that the installation depot machine has network connectivity to the database

server, Tivoli DAS server, and the MQSeries server. You can use the **ping** command to verify this connectivity.

Using the Single Machine Install

The single machine install option automatically installs all the necessary specialized servers and software on the installation depot.

- Tivoli Data Access Services (DAS) server
- MQSeries Server
- Java runtime environment
- IBM HTTP Server Powered by Apache
- Tivoli Presentation Services
- Tivoli Kernel Services

After installing the installation depot, be sure to read "Initially Starting the Installation Depot ORB" on page 31 before proceeding. After the installation program completes, the Tivoli Kernel Services installation depot ORB starts automatically. Interrupting this ORB during its initial start could result in a corrupted installation, requiring you to reinstall Tivoli Kernel Services.

Starting the Installation Program

To use the installation program to perform a single machine install, follow these steps:

Note: These instructions assume you have access to a Tivoli Kernel Services Installation CD-ROM. If you have received the installation program in an archived format, extract the necessary files and directories into a temporary directory and substitute that temporary directory for the root directory of the CD-ROM in the following directions. Tivoli Kernel Services cannot be installed from a remote or network drive.

1. Log on to the machine locally (not to a domain) with a user ID that has the appropriate access authority. See "Required Authority for the Installation" on page 19 for details. In addition, this user ID must also have the authority to create and delete tables on the

database server which is already installed on the installation depot. Tivoli recommends using the DB2 user ID when doing a single machine install.

2. Start the Tivoli Kernel Services installation program by running the appropriate install program from the root of the CD-ROM.

Platform	Install Program
Microsoft Windows	install.bat
UNIX	./install.sh

   Navigation through the installation program is handled in the usual way. Click **Next** to proceed to the next panel in the program. Click **Back** to return to the previous panel. Click **Cancel** to stop the installation program. Values entered into the program are lost after the **Cancel** has been confirmed by the user.

   Note: You might have to adjust your screen resolution so that the entire installation panel is visible.

3. On the **Choose the Type of Installation** screen, select what type of install to perform. Select **Single Machine install**.

4. On the **Enter the Namespace Name** screen, enter the name for your namespace. The name you select should be short, unique (if you have several installations, do not give them the same name), consist of printable characters, and logical. The namespace name is appended to objects in Tivoli Kernel Services, such as ORBs and ORB sets, and is displayed in the Tivoli Console and in trace and error logs.

5. On the **Specify the Tivoli Kernel Services Destination Directory** screen, specify the destination for the installation depot software. Click **Browse** to select a different location for the software.

   This directory will contain not only the Tivoli Kernel Services software but also the Tivoli Data Access Services (DAS) server software as well as the MQSeries server software.

6. On the **Enter the Database Server** screen, enter the user ID and password that must be used to access the database server that you have previously installed on the installation depot machine.
7. On the **Review Selected Installation Options** screen, review the summary of the choices you made. If the values shown are correct, click **Next** to begin the install. Otherwise, click **Back** and correct the values in error.

At this point, the installation depot installation program determines the amount of available disk space and begins copying files to the installation depot machine. This process takes a *long* time and due to the large number and small size of the files copied at the beginning of the installation process, the progress meter will stay at 0% for what seems like an excessive amount of time. Be patient.

The time the installation program takes to copy the files from the CD-ROM to the installation depot machine depends on the speed of the server, the speed of the CD-ROM drive, and other factors.

8. After the installation program has copied files to the installation depot machine, the **Configuring Tivoli Kernel Services** screen is displayed. During this configuration, a DB2 command line window opens over the Tivoli Kernel Services installation window and a series of DB2 commands are issued to configure the database. This window can be closed after the installation program completes.
9. When configuration is complete, the **Next** button on the **Configuring Tivoli Kernel Services** screen becomes active. Click **Next** to complete the installation.

Note: Even though the initial installation of the installation depot is complete, the initial configuration of Tivoli Kernel Services continues in the background. Do not reboot the machine, install other Tivoli applications, attempt to use the Tivoli Console, or install Tivoli Kernel Services bootprint servers until this configuration is complete. To determine when you can proceed, refer to "Initially Starting the Installation Depot ORB" on page 31.

After the installation program completes, you might still see up to three windows still open on the desktop. The DB2 CLP window, which was used to configure the database during installation, can be closed. Two other windows represent MQSeries services that are running in the background. Do not close these windows. When the installation depot machine is subsequently rebooted, these MQSeries services will start as services in the background and not have windows associated with them.

Note: If you run single machine install again, the DB2 database tables are deleted and then re-created. This essentially resets the configuration and security information back to the defaults; any data modified during the previous install or subsequent running of Tivoli Kernel Services is lost.

Multiple Machine Install

Multiple machine install gives you control over how the various servers and functions of Tivoli Kernel Services are deployed.

Starting the Installation Program

This section contains instructions on starting the installation program to perform the various tasks associated with an multiple machine install.

Note: These instructions assume you have access to a Tivoli Kernel Services Installation CD-ROM. If you have received the install software in an archived format, extract the necessary files and directories into a temporary directory and substitute that temporary directory for the root directory of the CD-ROM in the following directions. Tivoli Kernel Services cannot be installed from a remote or network drive.

1. Log on to the machine locally (not to a domain) with a user ID that has the appropriate access authority. See "Required Authority for the Installation" on page 19 for details.
2. Start the Tivoli Kernel Services installation program by running the appropriate install program from the root of the CD-ROM.

Platform	Install Program
Microsoft Windows	install.bat
UNIX	./install.sh

Navigation through the installation program is handled in the usual way. Click **Next** to proceed to the next panel in the program. Click **Back** to return to the previous panel. Click **Cancel** to stop the installation program. Values entered into the program are lost after the **Cancel** has been confirmed by the user.

3. Navigate to the **Choose the Type of Installation** screen and select **Multiple Machine Install**.
4. On the **Choose the Tivoli Kernel Services Software to Install** screen, select the items you wish to install on *this* machine. The options available are:

   - **Tivoli Kernel Services Installation Depot**

     Installs the Tivoli Kernel Services installation depot software.

   - **Tivoli DAS Server**

     Install a Tivoli Database Access Services (DAS) server.

   - **MQSeries Server**

     Install an MQSeries server.

   When more than one option is selected, the order of installation is as follows:

   a. Tivoli DAS Server (if specified)

   b. MQSeries Server (if specified)

   c. Tivoli Kernel Services Installation Depot (if specified)

   Your progress through the next series of screens will depend on the options you selected in the **Choose the Tivoli Kernel Services Software to Install** screen.

5. If you selected **Tivoli DAS Server**, you are prompted for the information needed for the installation on the following screens. Read the instructions on each screen, complete the requested information, and then click **Next** to continue:

   Note: On UNIX systems, the DB2 profile should be sourced before installing the DAS server.

   a. **Specify the Tivoli DAS Server Destination Directory**

      Enter the destination for the Tivoli DAS server software.

   b. **Enter the Tivoli DAS Server Information**

      Verify the host name for the Tivoli DAS server is correct. Specify the port number to be used for communications as well as the user ID and password that must be used to access the DB2 database server.

   c. **Enter the Directory Services Data Source Information**

      Enter the information for the Directory Services database created using the **tdacl** script.

   d. **Enter the Security Registry Data Source Information**

      Enter the information for the security registry database created using the **tdacl** script.

   e. **Enter the Tivoli Console Data Source Information**

      Enter the information for the Tivoli Console database created using the **tdacl** script.

6. If you selected **MQSeries Server**, you are prompted for the information needed for the installation on the following screens. Read the instructions on each screen, complete the requested information, and then click **Next** to continue:

   a. **Specify the MQSeries Server Destination Directory**

      Enter the destination for the MQSeries server software.

   b. **Enter the MQSeries Server Information**

      Verify the host name for the MQSeries server, and specify the communications port to be used.

7. If you selected **Tivoli Kernel Services Installation Depot**, you are prompted for the information needed for the installation on the following screens. Read the instructions on each screen, complete the requested information, and then click **Next** to continue:

   a. **Enter the Namespace Name**

      Enter the name to be used for your installation. The name you select should be short, unique (if you have several installations, do not give them the same name), consist of printable characters, and logical. The namespace name is appended to objects in Tivoli Kernel Services, such as ORBs and ORB sets, and is displayed in the Tivoli Console and in trace and error logs.

   b. **Specify the Tivoli Kernel Services Installation Depot Destination Directory**

      Enter the destination for the Tivoli Kernel Services installation depot software.

   c. **Enter the Information for the Tivoli Kernel Services Installation Depot**

      Verify the host name for the installation depot machine. Enter the communications port to be used for the Tivoli Kernel Services installation depot.

      By default, **Automatically start this installation depot after install completes** is checked. If you do not wish to have the installation depot ORB started automatically when the installation program completes, clear the check box.

   d. **Enter the Directory Services Data Source Information**

      Enter the pool size for the Directory Services database. This value should be set slightly higher than the maximum number of ORBs you expect in your Tivoli Kernel Services installation. For more information about pool size, see "Pool Size" on page 117.

   e. **Enter the Tivoli Console Information**

Enter the port number to be used by the Java implementation of the Tivoli Console running on the installation depot machine.

8. **Review Selected Installation Options**

   Review the options you selected in the installation program. If the values shown are correct, click **Next** to begin the install. Otherwise, click **Back** and correct the values in error.

At this point, the installation depot installation program determines the amount of available disk space and begins copying files to the installation depot machine. This process takes a *long* time and due to the large number and small size of the files copied at the beginning of the installation process, the progress meter will stay at 0% for what seems like an excessive amount of time. Be patient.

The time the installation program takes to copy the files from the CD-ROM to the installation depot machine depends on the speed of the server, the speed of the CD-ROM drive, and other factors.

9. After the installation program has copied files to the installation depot machine, the **Configuring Tivoli Kernel Services** screen is displayed. The installation program performs the initial configuration of the two ORBs on the installation depot.

   During this configuration, a DB2 command line window opens over the Tivoli Kernel Services installation window and a series of DB2 commands are issued to configure the database. This window can be closed after the installation program completes.

10. When configuration is complete, the **Next** button on the **Configuring Tivoli Kernel Services** screen becomes active. Click **Next** to complete the installation.

Note: Even though the initial installation of the installation depot is complete, the initial configuration of Tivoli Kernel Services continues in the background. Do not reboot the machine, install Tivoli applications, attempt to use the Tivoli Console, or install Tivoli Kernel Services bootprint servers until this configuration is complete. To determine when you can proceed, refer to “Initially Starting the Installation Depot ORB” on page 31.

After the installation program completes, you might still see up to three windows still active. The DB2 CLP window, which was used to configure the database during installation, can be closed. If you installed an MQSeries server, two other windows might be open representing MQSeries services that are running in the background. Do not close these windows. When the installation depot machine is subsequently rebooted, these MQSeries services will start as services in the background and not have windows associated with them.

After the configuration of the installation depot ORB and the Tivoli Presentation Services ORB is complete, the installation depot ORB is started in the background, if this was requested.

Initially Starting the Installation Depot ORB

After you have installed Tivoli Kernel Services, the installation depot ORB (orb.1) should be running in the background if you did a single machine install or if you chose to have the ORB start automatically in the multiple machine install.

The initial start of the installation depot ORB after install takes a long time and results in a large amount of data being writen to the security registry. It is very important that you allow the installation depot ORB to initialize completely before you:

- install a bootprint server
- start the Tivoli Presentation Services ORB (orb.2) on the installation depot machine
- install other Tivoli applications
- reboot the installation depot machine

Interrupting the installation depot ORB while it is writing data to the security registry could result in the ORB being left in an inconsistent state, requiring the reinstall of Tivoli Kernel Services.

Determining When the Installation Depot is Ready

The initial start of the installation depot ORB is complete when all of the following conditions are met. Details on verifying these conditions are provided in the following sections.

Note: Subsequent starts of the installation depot ORB occur significantly faster than the initial one. The build of the help set only occurs on the first start of the installation depot ORB; there is no need to verify it on subsequent starts of the installation depot ORB.

Condition	Met?
HTTP server installed by Tivoli Kernel Services is operational. The installation depot uses the HTTP server to provide components to the bootprint servers.	
Installation depot ORB is accepting **wcmd** commands	
Build help set utility has completed for the Tivoli Console. The Tivoli Assistant online help must be built on the installation depot before a Tivoli Console is started.	

HTTP server installed by Tivoli Kernel Services is operational

Point a Web browser to the following URL, substituting the fully qualified host name of the installation depot machine for *fully.qualified.hostname*:

`http://`*`fully.qualified.hostname`*`/Repository`

If a list of JAR files is displayed, the HTTP server is operational and ready to support the installation of one or more bootprint servers.

Installation depot ORB is accepting wcmd commands

To verify that the installation depot is accepting **wcmd** commands, do the following:

1. Open a command window and change to the directory where the Tivoli Kernel Services software is installed.

2. Go to the `orb.1` directory.
3. Set up the environment for issuing commands and then issue the following set of commands using the **superadmin** pre-defined user ID.

Platform	Commands

UNIX	`. ./setupEnv.sh`
	`wcmd -u superadmin info getOrbs`
	`wcmd -u superadmin svc ls lsm`
	`wcmd -u superadmin comp components`

Microsoft Windows	`setupenv`
	`wcmd -u superadmin info getOrbs`
	`wcmd -u superadmin svc ls lsm`
	`wcmd -u superadmin comp components`

when prompted for a password, use **password**.

Build help set utility has completed for the Tivoli Console

Monitor the Local ORB Creator and Killer (LOCK) output log, as indicated in "Monitoring and Troubleshooting Installation" on page 35. Look for the following message

```
FWP1734I The build help set utility has completed successfully.
```

After the help set has been created, online help is available to the Tivoli Console running on the installation depot and on Tivoli Console bootprint servers.

The help set is only created on the initial start of the installation depot ORB. This message will not occur on subsequent starts of the ORB.

If the installation depot ORB does not appear to be starting in a reasonable amount of time, use the information provided in "Monitoring and Troubleshooting Installation" on page 35 to check the Local ORB Creator and Killer (LOCK) output and error logs to assist in diagnosing the situation.

Using Silent Install

Silent install allows you to record the choices you make during the installation depot install and then subsequently play back those choices without manual intervention on one or more other servers. This is convenient in test environments where you might be installing and uninstalling the installation depot frequently.

Recording an Install for Subsequent Playback

1. Start the installation program with the record option, **-r**, and specify a filename.

Platform	Silent Install Record
Microsoft Windows	install.bat -r *filename*
UNIX	./install.sh -r *filename*

2. Proceed through the installation program and complete the panels as desired.
3. After you click **Finish** a pop-up window will indicate that the script has been saved. Clicking **OK** starts the install on the server.

Performing a Silent Install

1. Start the installation program with the playback option, **-p**, and specify the filename of an install that was previously recorded.

Platform	Silent Install Playback
Microsoft Windows	install.bat -p *filename*
UNIX	./install.sh -p *filename*

2. The install is performed without user intervention.

Monitoring and Troubleshooting Installation

During the installation process, the installation program and Tivoli Kernel Services writes data to log files. These log files can be useful in diagnosing problems with installing an installation depot, an MQSeries server, or a Tivoli DAS server. You can also monitor these files during normal system operation to check on the status of Tivoli Kernel Services.

In the table below, the following environment variables are used:

%TEMP%
Temporary directory that is used by the operating system.

%TKS_BASEDIR%
The location where the installation depot software was installed. The default directory location is `Tivoli`.

%ORBDIR%
The directory containing the files associated with a specific ORB. Use **orb.1** for the installation depot ORB, and **orb.2** for the Tivoli Presentation Services ORB.

%DASDIR%
The location where the Tivoli Data Access Services (DAS) server software was installed. For a single machine install, this directory is under the **%TKS_BASEDIR%** directory.

%MQDIR%
The destination directory on which MQSeries was installed. For a single machine install, this directory is under the **%TKS_BASEDIR%** directory.

File	Location
Installation program output log	**Windows 2000** `%TEMP%\tksOut.log` **UNIX** `/var/tmp/tksOut.log`
Installation program error log	**Windows 2000** `%TEMP%\tksErr.log` **UNIX** `/var/tmp/tksErr.log`

File	Location
Local ORB Creator and Killer (LOCK) output log	**Windows 2000** %TKS_BASEDIR%\%ORBDIR%\log\stdout.txt **UNIX** %TKS_BASEDIR%/%ORBDIR%/log/stdout
Local ORB Creator and Killer (LOCK) error log	**Windows 2000** %TKS_BASEDIR%\%ORBDIR%\log\stderr.txt **UNIX** %TKS_BASEDIR%/%ORBDIR%/log/stderr
ORB message log	**Windows 2000** %TKS_BASEDIR%\%ORBDIR%\log\message1.log **UNIX** %TKS_BASEDIR%/%ORBDIR%/log/message1.log
MQSeries server install log	**Windows NT and Windows 2000** %MQDIR%\.\ext\mqm\tksSi\tksInstall.log **AIX** /usr/tksInstall.log **Solaris** /opt/tksInstall.log
IBM HTTP server install log	**Windows 2000** %ORBDIR%\ext\HTTPServer\tksSi\tksInstall.log **UNIX** %ORBDIR%/ext/HTTPServer/tksSi/tksInstall.log
Tivoli DAS server install log	**Windows NT and Windows 2000** %DASDIR%\ext\das\server\tksSi\tksInstall.log **UNIX** %DASDIR%/ext/das/server/tksSi/tksInstall.log

On UNIX systems, the **tail** command can be used to monitor the progress of the Tivoli Kernel Services log files.

```
tail -f stdout
```

Similar tools are available on Windows systems.

4

Installing a Bootprint

This chapter contains the information you need to install a bootprint on a Tivoli Kernel Services server in your installation.

Special Notice

In order to determine the type of bootprint you need to install, please consult the documentation that came with the Tivoli management software product you purchased.

Before You Begin

Before you run the bootprint installation program, be sure that you:

1. Have performed all the steps described in "Installing the Installation Depot" on page 19, including verifying that the initial start of the installation depot ORB is complete.
2. Have the completed Planning Worksheet from the *Planning for Tivoli Kernel Services* document.

   The Planning Worksheet allows you to quickly locate and subsequently enter the information required during the bootprint install process.
3. Have started the installation depot ORB for the first time, allowed it to complete all its initialization, and optimized it as outlined in "Tuning Databases for Better Performance" on page 48.

Remember that you can not install a bootprint ORB on the installation depot machine.

Required Authority for the Bootprint Install

The bootprint installation program must be run as **root** on UNIX systems, or from a user ID with **Administrator** authority on Windows systems. Users must be logged on locally and not to a domain.

In addition, you must have Tivoli Kernel Services authorization to install a bootprint server. By default, Tivoli Kernel Services provides a user ID called **Installer** with a password of **password** that can be used to install bootprint servers.

The bootprint installation program copies files from the bootprint CD-ROM to the local disk of the bootprint server machine. The user ID running the bootprint installation program must have read/write access to the destination directory or file system for the bootprint install. In addition, read/write access is also needed to the following temporary locations:

UNIX

```
/usr/local
/etc
```

Windows C:\

On Microsoft Windows systems, Tivoli Kernel Services also creates a special Windows user ID called **tksUser**. This user ID is used to launch ORBs and must not be modified or deleted.

Checking the Environment

You should verify that the following conditions are true before starting a bootprint install.

Condition	True?
Installation depot ORB is running	
Network connectivity to the installation depot machine	
Network connectivity to an MQSeries server	

Condition	True?
Operating system prerequisites met	
Computer name fully qualified on Windows 2000 systems	

Installation depot ORB is running

Verify that the installation depot ORB is running by pointing a Web browser to the following URL, substituting the fully qualified host name of the installation depot machine for *fully.qualified.hostname*:

`http://`*`fully.qualified.hostname`*`/Repository`

If a list of JAR files is displayed, the installation depot is ready for the installation of one or more bootprint servers.

Note: If you have just installed the installation depot, be sure to read "Initially Starting the Installation Depot ORB" on page 31 to make sure Tivoli Kernel Services is ready to accept a bootprint install.

Network connectivity to the installation depot machine

Verify that the machine where you are installing the bootprint has network connectivity to the installation depot machine. You can use the **ping** command to verify this connectivity.

Network connectivity to an MQSeries server

Verify that the bootprint machine has network connectivity to an MQSeries server. If you did a single machine install of the installation depot, this step can be omitted since the MQSeries server is on the same machine as the installation depot. You can use the **ping** command to verify this connectivity.

Operating system prerequisites met

If installing on a UNIX, Solaris, or Windows NT 4.0 system, be sure that the requirements outlined in the *Planning for Tivoli Kernel Services* document are met.

Computer name is fully qualified on Windows 2000 systems

If installing on Microsoft Windows 2000, ensure that the

computer name is fully qualified. To verify this, right click **My Network Places** and select **Properties**. Go to the **Advanced** ->**Network Identification** section. If the **Full computer name** field is not fully qualified with a domain name, click **Properties** and **More...**, then enter the domain name for the machine.

The machine will need to be rebooted for this change to take effect.

Using the Bootprint Installation Program

This section contains instructions for running the bootprint installation program to set up the Tivoli Kernel Services server.

Note: These instructions assume you have access to a Tivoli Kernel Services Bootprint Install CD-ROM. If you have received the bootprint install software in an archived format, extract the necessary files and directories into a temporary directory and substitute that temporary directory for the root directory of the CD-ROM in the following directions. Tivoli Kernel Services cannot be installed from a remote or network drive.

Note: You cannot install more than one bootprint on a given machine.

1. Log on with a user ID that has the appropriate access authority. See “Required Authority for the Bootprint Install” on page 38 for details.
2. Start the Tivoli Kernel Services bootprint installation program by running the appropriate install program from the root of the CD-ROM.

Platform	Install Program
Microsoft Windows	install.bat
UNIX	./install.sh

Navigation through the bootprint installation program is handled in the usual way. Click **Next** to proceed to the next panel in the

program. Click **Back** to return to the previous panel. Click **Cancel** to stop the bootprint installation program. Values entered into the program are lost after the **Cancel** has been confirmed by the user.

Note: You might have to adjust your screen resolution so that the entire installation panel is visible.

3. On the **Choose the Type of Bootprint** screen, select what type of bootprint to install.

   There are three types of bootprints:

   Tivoli Kernel Services server
   Tivoli Kernel Services servers are resources that have been dedicated to run Tivoli Kernel Services code. Tivoli Kernel Servers servers can be defined as gateways or general purpose servers.

   Tivoli Kernel Services endpoint server
   Tivoli Kernel Services endpoint servers are resources that have been enabled to run Tivoli Kernel Services application software. Refer to the documentation provided with your application to determine when endpoint servers should be installed.

   Tivoli Console server
   Tivoli Console servers are resources that have been enabled to run the Tivoli Console for administering Tivoli Kernel Services and its applications

   Note: On Solaris, if you select the Tivoli Console bootprint, you must allow sufficient time (20-30 minutes) for the necessary components to deploy before you launch the Tivoli Console. On Windows operating systems, wait for the Tivoli Console icon to appear, then double-click on it to launch the Tivoli Console.

4. On the **Specify the Tivoli Kernel Services Bootprint Destination Directory** screen, specify the directory where you want to install the bootprint.

5. On the **Enter the Bootprint Information** screen, enter the installation depot and MQSeries server information.

   The name of the installation depot supplied on the screen by default cannot be used and the proper name must be supplied. Specify the name of the installation depot machine on this panel. If you did a single machine install of the installation depot, accept the default port of 9990; otherwise use the port value you specified when you installed the installation depot.

   You must specify the name and port number for the MQSeries server. If you did a single machine install of the installation depot, specify the name of the installation depot machine for the MQSeries server and use the default port number of 1414. If you did a multiple machine install, use the name of the appropriate MQSeries server and its corresponding port number.

   If you do not want the bootprint ORB to start automatically after the installation program completes, clear the **Automatically start this bootprint server after install completes** check box. On a Tivoli Console server, you will still need to manually launch the Tivoli Console regardless of how this option is set.

6. On the **Enter the Tivoli Kernel Services Installer Information** screen, enter the Tivoli Kernel Services user ID and password that has the proper authority to install a bootprint. Tivoli Kernel Services provides a user ID called **Installer** with a default password of **password** for this purpose. If these values have not been changed, you can simply accept the defaults provided on this screen.

7. On the **Review Selected Installation Options** screen, review the summary of the choices you made. If the values shown are correct, click **Next** to begin the install. Otherwise, click **Back** and correct the values in error.

The time the bootprint installation program takes to copy the files from the bootprint CD-ROM to the bootprint server depends on the speed of the server, the speed of the CD-ROM drive, and other factors.

After the files have been copied to the bootprint server, the server contacts the installation depot and completes final configuration. When the configuration is complete, the bootprint server ORB is started in the background if requested.

Using Silent Install

Silent install allows you to record the choices you make during a bootprint install and then subsequently play back those choices without manual intervention on one or more other servers. This is convenient in test environments and in deploying a series of bootprint servers configured the same way.

Recording an Install for Subsequent Playback

1. Start the bootprint installation program with the record option, **-r**, and specify a filename.

Platform	Silent Install Record
Microsoft Windows	install.bat -r *filename*
UNIX	./install.sh -r *filename*

2. Proceed through the bootprint installation program and complete the panels as desired.
3. After you click **Finish** a pop-up window will indicate that the script has been saved. Clicking **OK** starts the bootprint install on the server.

Performing a Silent Install

1. Start the bootprint installation program with the playback option, **-p**, and specify the filename of a bootprint install that was previously recorded.

Platform	Silent Install Playback
Microsoft Windows	install.bat -p *filename*
UNIX	./install.sh -p *filename*

2. The install is performed without user intervention.

Monitoring and Troubleshooting Installation

During the installation process, the installation program and Tivoli Kernel Services writes data to log files. These log files can be useful in diagnosing problems with installing a bootprint.

In the table below, the following environment variables are used:

%TEMP%
Temporary directory that is used by the operating system.

%TKS_BASEDIR%
The location where the bootprint software was installed. The default directory location is `TivoliBP`.

File	Location
Installation program output log	**Windows NT and Windows 2000** `%TEMP%\tksOut.log` **UNIX** `/var/tmp/tksOut.log`
Installation program error log	**Windows NT and Windows 2000** `%TEMP%\tksErr.log` **UNIX** `/var/tmp/tksErr.log`
Local ORB Creator and Killer (LOCK) output log	**Windows NT and Windows 2000** `%TKS_BASEDIR%\orb.1\log\stdout.txt` **UNIX** `%TKS_BASEDIR%/orb.1/log/stdout`
Local ORB Creator and Killer (LOCK) error log	**Windows NT and Windows 2000** `%TKS_BASEDIR%\orb.1\log\stderr.txt` **UNIX** `%TKS_BASEDIR%/orb.1/log/stderr`
ORB message log	**Windows NT and Windows 2000** `%TKS_BASEDIR%\orb.1\log\message1.log` **UNIX** `%TKS_BASEDIR%/orb.1/log/message1.log`

5

Running Tivoli Kernel Services

After you have installed the installation depot and one or more bootprints, some of the tasks you can perform include:

For other common tasks, see the following books.

- For an overview of the security system, see ″Planning Security″ in *Planning for Tivoli Kernel Services*.
- For locating the super administrator accounts shipped with the product and modifying those administrators, see ″Planning Security″ in *Planning for Tivoli Kernel Services*.

- For information on modifying access conditions, accounts, capabilities, users, roles, and security, see ″Administer Security″ in *Introducing Tivoli Kernel Services Administration.*
- For a list of all system roles and the capabilities and access conditions for those roles, see ″System-Defined Roles and SuperAdministrators″ in *Introducing Tivoli Kernel Services Administration*.
- For an overview of the console, see ″A New Tivoli Console″ in *Introducing Tivoli Kernel Services Administration*.

Interacting with ORBs Using the CLI

The command line interface, or CLI, of Tivoli Kernel Services allows you to directly interact with a running ORB using the **wcmd** command. A Tivoli Kernel Services command must be issued from a user ID with the proper authority to perform the requested action.

Tivoli Kernel Services provides a number of predefined user IDs for use.

User ID	Password	Role
superadmin	password	Automatically has complete access to all objects and all roles in the system, whether they are created by Tivoli Kernel Services, by an application, or by the system administrator. Use this user ID and password as you would use **root** in UNIX or **Administrator** in Windows.
sysadmin	password	Has access to all Tivoli Kernel Services objects and roles defined at the time of installation. Does not have access to application-created roles unless explicitly given that access.

User ID	Password	Role
Installer	password	Has the ability to perform a bootprint installation directed to the specified installation depot. The Installer user ID was discussed in "Installing a Bootprint" on page 37

When entering a command, you can explicitly specify a user ID by using the **-u** option. For instance, to issue the command to list ORBs on the installation depot ORB:

1. Open a command window and change to the directory where the Tivoli Kernel Services software is installed.
2. Go to the appropriate ORB directory, such as `orb.1` or `orb.2`.
3. Set up the environment for issuing commands and then issue the command using the **superadmin** pre-defined user ID.

Platform	Commands
UNIX	`. ./setupEnv.sh` `wcmd -u superadmin info getOrbs`
Microsoft Windows	`setupenv` `wcmd -u superadmin info getOrbs`

when prompted for a password, use **password**.

If you will be frequently using the CLI from a particular user ID on a particular UNIX or Windows machine, you can grant that user the necessary Tivoli Kernel Services authorization and not have to explicitly specify a user ID using the **-u** option on every wcmd. For instructions on doing this, see the *Security Implications for wcmd Commands* section of the *Tivoli Kernel Services Command Reference* document.

Note: The other wcmd examples in this chapter assume that the person issuing the commands is implicitly identified to Tivoli Kernel Services in this manner. If this is not the case, specify

the **-u superadmin** option to ensure you have the necessary authorization to perform the command.

Tuning Databases for Better Performance

Tivoli Kernel Services relies heavily on its underlying databases for managing configuration and security information. As significant amounts of data are written to these databases, the overall performance of the installation depot, the bootprint servers, and Tivoli Kernel Services in general can degrade.

Tivoli recommends that you tune the databases used by Tivoli Kernel Services:

- After the initial install of the installation depot and the first bootprint server
- After installing a number of bootprint servers
- After installing additional Tivoli management software
- After configuring a large number of Tivoli Kernel Services servers, ORBs, or ORB sets
- Periodically during normal system operation (at least once a week)

Introducing runstats

Tivoli Kernel Services provides a script, called **runstats**, that causes the database server to generate statistics for the Tivoli Kernel Services databases. This command updates the statistics about the physical characteristics of a database table and its associated indices. These characteristics include the number of records, the number of pages, and the average record length. The DB2 optimizer uses these statistics when generating query plans to access the data in the database.

Running `runstats` can result in a *significant* performance improvement. Tivoli strongly recommends that you run `runstats` at least weekly.

Optimizing Databases on UNIX Systems

To run `runstats` on a database server running on a UNIX system:

1. Execute the `runstats` shell script located in:

   `$ORBDIR/bin/generic/runstats.sh`

   substituting the location of the installation depot ORB for *$ORBDIR*
2. Enter the DB2 user ID and password when prompted.
3. Enter the information requested about the databases you want to run `runstats` against.

If you installed Tivoli Kernel Services using the default database names, you can run `runstats` in an automated fashion by issuing the following, all on one line:

```
$ORBDIR/bin/generic/runstats.sh DB2userid DB2password
  < $ORBDIR/bin/generic/runstats.in
```

If you chose different database names during installation, modify the `runstats.in` file and then use the previous command.

Optimizing Databases on Windows Systems

If you installed Tivoli Kernel Services using the default database names, you can run `runstats` on a Microsoft Windows system by doing the following:

1. Open a command prompt.
2. Enter the following command:

   `$ORBDIR/bin/w32-ix86/runstats.bat DB2userid DB2password`

   substituting the location of the installation depot ORB, such as `C:\Tivoli\orb.1`, for *$ORBDIR*.

If you chose different database names, make a copy of the `runstats.bat` file, and modify the copy to use the names you selected.

Verifying that the Installation Depot ORB is Running

Note: If this is first time the installation depot ORB has been started, be sure to read "Initially Starting the Installation Depot ORB" on page 31 before continuing.

Before starting the Tivoli Presentation Services ORB or starting a bootprint ORB, the installation depot ORB must be running. To verify this:

1. Point a Web browser to the following URL, substituting the fully qualified host name of the installation depot machine for *fully.qualified.hostname*:

   `http://fully.qualified.hostname/Repository`

   Verify that a list of JAR files is displayed.

2. Open a command window on the installation depot machine and change to the location where the installation depot software is installed.

3. Go to the `orb.1` directory.

4. Set up the environment for issuing commands:

Platform	Commands
UNIX	`. ./setupEnv.sh`
Windows	`setupenv`

5. Verify that the installation depot ORB is up and running by issuing the following command:

   ```
   wcmd svc ls lsm
   ```

   and check that the **PsNsImpl** service is listed.

Using the Tivoli Console

The Tivoli Console is the graphical user interface that is used for Tivoli Kernel Services Administration. The Tivoli Console can be run from either the installation depot or a Tivoli Console server. Before starting the Tivoli Console, verify that the installation depot ORB is running.

Running the Tivoli Console on the Installation Depot

To use the Tivoli Console on the installation depot machine, you must:

- Start the Tivoli Presentation Services ORB, if it is not already running
- Start the Tivoli Console

Starting the Tivoli Presentation Services ORB

To start the Tivoli Presentation Services ORB on the installation depot if it is not already running:

1. On the installation depot machine, open a command window and change to the location where the installation depot software was installed.
2. Go to the `orb.2` directory.
3. Start the Local ORB Creator and Killer (LOCK) service to launch the ORB.

   Note: On UNIX systems, you must modify the appropriate `LOCKCfg.properties` file:

   AIX `orb.2/bin/aix4-r1/LOCKCfg.properties`

   Solaris
   `orb.2/bin/solaris2/LOCKCfg.properties`

   to set the DISPLAY environment variable before starting the ORB. Set the DISPLAY environment variable to:

   `DISPLAY=`*`hostname`*`:0.0`

where *hostname* is the system where you want the Tivoli Console to be displayed. This could be the UNIX machine itself or a remote system.

Platform	Commands
AIX	`chdir bin/aix4-r1` `./LOCK`
Solaris	`chdir bin/solaris2` `./LOCK`
Windows	`cd bin/w32-ix86` `LOCK`

The Tivoli Presentation Services ORB also can be started as a service on Windows 2000 systems or as a daemon on UNIX systems.

Windows 2000

1. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**
2. Highlight **Tivoli Kernel Services orb.2** and click **Start**.

Windows operating systems

```
net start orb.2
```

Solaris

```
/etc/init.d/orb.2.LOCK start
```

AIX

```
startsrc -s LOCK.orb.2
```

Starting the Tivoli Console on the Installation Depot

To start the Tivoli Console:

1. Open another command window on the installation depot machine and change to the location where the installation depot software is installed.
2. Go to the `orb.2` directory.

3. Set up the environment for issuing commands:

Platform	Commands
UNIX	`. ./setupEnv.sh`
Windows	`setupenv`

4. Verify that the ORB is up and running by issuing the following command:

   ```
   wcmd svc ls lsm
   ```

   and check that **PsJcLauncherService** is listed.

   If it is not listed, wait until it is running. The **PsJcLauncherService** must be running before starting the Tivoli Console.

5. Start the Tivoli Console:

   ```
   wcmd jclauncher LaunchJC
   ```

   On Windows systems, you can also click the **Tivoli Console** icon on the Desktop to start the Tivoli Console

See *Introducing Tivoli Kernel Services Administration* for information on using the Tivoli Console to administer Tivoli Kernel Services.

Starting the Tivoli Console on a Tivoli Console Server

To start the Tivoli Console on a Tivoli Console server:

1. Start the bootprint ORB, if it is not already running. Starting an ORB is described in "Starting or Restarting Any ORB" on page 58.

   Note: If you are running on a UNIX system, see "Considerations for Running the Tivoli Console on UNIX Systems" on page 54 before starting the ORB.

2. Open a command window on the Tivoli Console server and change to the location where the Tivoli Console server software was installed.

3. Go to the `orb.1` directory of the bootprint.
4. Set up the environment for issuing commands:

Platform	Commands
UNIX	`. ./setupEnv.sh`
Windows	`setupenv`

5. Verify that the ORB is up and running by issuing the following command:

   ```
   wcmd svc ls lsm
   ```

   and check that **PsJcLauncherService** is listed.

   If it is not listed, wait until it is running. The **PsJcLauncherService** must be running before starting the Tivoli Console.

6. Start the Tivoli Console:

   ```
   wcmd jclauncher LaunchJC
   ```

   On Windows systems, you can also click the **Tivoli Console** icon on the Desktop to start the Tivoli Console

See *Introducing Tivoli Kernel Services Administration* for information on using the Tivoli Console to administer Tivoli Kernel Services.

Considerations for Running the Tivoli Console on UNIX Systems

On UNIX systems, the `LOCKCfg.properties` file associated with the ORB must be modified before starting the Tivoli Console.

1. Stop the bootprint ORB, if it is already running. (See "Stopping an ORB" on page 56 for details on doing this.)
2. Locate the appropriate `LOCKCfg.properties` file:

   AIX orb.1/bin/aix4-r1/LOCKCfg.properties

Solaris

orb.1/bin/solaris2/LOCKCfg.properties

3. Set the DISPLAY environment variable in the `LOCKCfg.properties` file to:

   `DISPLAY=`*`hostname`*`:0.0`

   where *hostname* is the system where you want the Tivoli Console to be displayed. This could be the UNIX machine itself or a remote system.

4. Start the ORB to pick up the change. (See "Starting or Restarting Any ORB" on page 58 for details.)

Signing on to the Tivoli Console

After the Tivoli Console initializes, you are prompted to sign on.

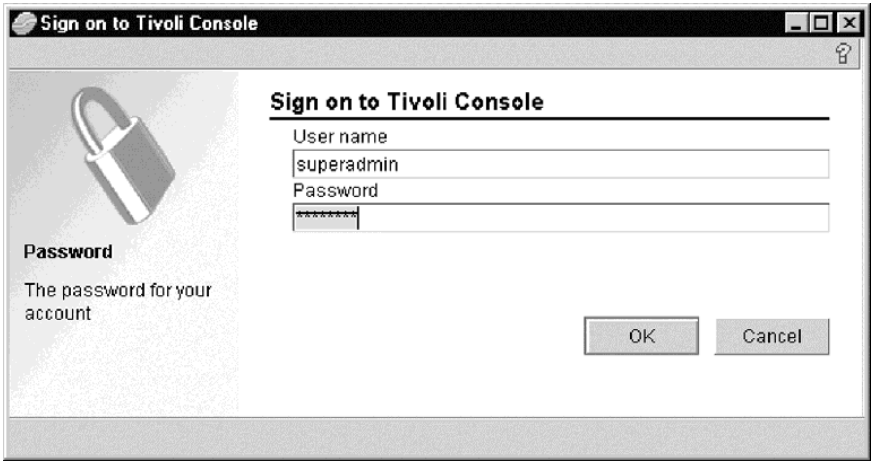

Figure 1. Tivoli Console sign on

Tivoli Kernel Services is shipped with three default user IDs and passwords. The default user IDs and their corresponding passwords are listed in the following table:

User ID	Password	Role
superadmin	password	Automatically has complete access to all objects and all roles in the system, whether they are created by Tivoli Kernel Services, by an application, or by the system administrator. Use this user ID and password when you initially sign on to the Tivoli Console.
sysadmin	password	Has access to all Tivoli Kernel Services objects and roles defined at the time of installation. Does not have access to application-created roles unless explicitly given that access. This user ID does not have any Tivoli Kernel Services portfolio tasks defined.
Installer	password	Has the ability to perform a bootprint installation directed to the specified installation depot. This user ID does not have any Tivoli Kernel Services portfolio tasks defined.

Use the **superadmin** user ID when you sign on to the Tivoli Console for the first time. This user ID gives you access to all the portfolio tasks and allows you to perform all actions associated with those portfolio tasks.

Stopping an ORB

The generic procedure for stopping an ORB is platform dependent.

Windows NT

1. Select **Start** → **Settings** → **Control Panel** → **Services**

2. Highlight the Tivoli Kernel Services ORB you want to stop and click **Stop**.

Windows 2000

1. Select **Start → Settings → Control Panel → Administrative Tools → Services**
2. Highlight the Tivoli Kernel Services ORB you want to stop and click **Stop**.

Windows operating systems

`net stop` *`ORBname`*, where *ORBname* is either **orb.1** or **orb.2**.

Solaris

`/etc/init.d/`*`ORBname`*`.LOCK stop`

AIX `stopsrc -s LOCK.`*`ORBname`*

Alternately:

1. Open a command window and change to the location where the Tivoli Kernel Services software is installed.
2. Go to the appropriate ORB directory, such as the `orb.1` or `orb.2` directory.
3. Enter the following commands:

Platform	Commands
UNIX	`. ./setupEnv.sh` `wcmd orb shutdown`
Microsoft Windows	`setupenv` `wcmd orb shutdown`

The `wcmd orb shutdown` command can be issued from any running ORB and directed to a specific ORB. See the *Tivoli Kernel Services Command Reference* for details.

Starting or Restarting Any ORB

The generic procedure for starting or restarting an ORB is also platform dependent.

Windows NT

1. Select **Start** → **Settings** → **Control Panel** → **Services**
2. Highlight the Tivoli Kernel Services ORB you want to start and click **Start**.

Windows 2000

1. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**
2. Highlight the Tivoli Kernel Services ORB you want to start and click **Start**.

Windows operating systems

`net start` *`ORBname`*, where *ORBname* is either **orb.1** or **orb.2**.

Solaris

`/etc/init.d/`*`ORBname.LOCK`* `start`

AIX `startsrc -s` *`LOCK.ORBname`*

Alternately:

1. Open a command window and change to the location where the Tivoli Kernel Services software is installed.
2. Go to the appropriate ORB directory, such as the `orb.1` or `orb.2` directory.
3. Run the Local ORB Creator and Killer (LOCK) service to launch the ORB:

Platform	Commands
AIX	`chdir bin/aix4-r1` `./LOCK`
Solaris	`chdir bin/solaris2` `./LOCK`

Platform	Commands
Windows	`cd bin/w32-ix86` `LOCK`

Stopping and Starting a Tivoli DAS Server

The Tivoli Data Access Services (DAS) server runs as a daemon on UNIX systems and as a service on Windows systems.

You can start or stop a Tivoli DAS server by doing the following:

Windows NT

1. Select **Start** → **Settings** → **Control Panel** → **Services**
2. Highlight **Tivoli Kernel Services DASRestarter**.
3. Click **Start** to start the Tivoli DAS server or **Stop** to stop the Tivoli DAS server.

Windows 2000

1. Select **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services**
2. Highlight **Tivoli Kernel Services DASRestarter**.
3. Click **Start** to start the Tivoli DAS server or **Stop** to stop the Tivoli DAS server.

UNIX To stop the Tivoli DAS server, enter

`<DasInstallDir>/ext/das/server/tksSi/stopserver`

To start the Tivoli DAS server, enter

`<DasInstallDir>/ext/das/server/tksSi/startserver`

substituting the directory where the Tivoli DAS server software was installed for <*DasInstallDir*>.

Stopping and Starting an MQSeries Server

An MQSeries server runs as a daemon on UNIX systems and as a service on Windows systems.

You can stop an MQSeries server installed by Tivoli Kernel Services by opening a command prompt and then entering the following commands:

```
endmqm -w QM_hostname
endmqcsv -w QM_hostname
endmqlsr -w hostname
```

substituting the fully qualified name of the MQSeries server for *hostname*.

You can start an MQSeries server installed by Tivoli Kernel Services by opening a command prompt and entering the appropriate command based on the operating system:

Windows

```
net start "Tivoli Kernel Services MQRestarter"
```

AIX

```
/usr/mqm/tksSi/startMqOnReboot start
```

Solaris

```
/etc/rc3.d/S50tksmq start
```

Monitoring System Activity

You can monitor the activities of Tivoli Kernel Services during normal system operation by periodically checking the following log and message files.

In the table below, the following environment variables are used:

%TKS_BASEDIR%
: The location where the Tivoli Kernel Services software was installed. The default directory location for the installation depot is `Tivoli`. The default directory location for a bootprint server is `TivoliBP`

%ORBDIR%
: The directory containing the files associated with a specific ORB. Use **orb.1** for the installation depot ORB or a bootprint ORB, and **orb.2** for the Tivoli Presentation Services ORB on the installation depot.

%DASDIR%

The location where the Tivoli Data Access Services (DAS) server software was installed. For a single machine install, this directory is under the **%TKS_BASEDIR%** directory.

%MQDIR%

The destination directory on which MQSeries was installed. For a single machine install, this directory is under the **%TKS_BASEDIR%** directory.

File	Location
Local ORB Creator and Killer (LOCK) output log	**Windows 2000** `%TKS_BASEDIR%\%ORBDIR%\log\stdout.txt` **UNIX** `%TKS_BASEDIR%/%ORBDIR%/log/stdout`
Local ORB Creator and Killer (LOCK) error log	**Windows 2000** `%TKS_BASEDIR%\%ORBDIR%\log\stderr.txt` **UNIX** `%TKS_BASEDIR%/%ORBDIR%/log/stderr`
ORB message log	**Windows 2000** `%TKS_BASEDIR%\%ORBDIR%\log\message1.log` **UNIX** `%TKS_BASEDIR%/%ORBDIR%/log/message1.log`
MQSeries server install log	**Windows NT and Windows 2000** `%MQDIR%\.\ext\mqm\tksSi\tksInstall.log` **AIX** `/usr/tksInstall.log` **Solaris** `/opt/tksInstall.log`
IBM HTTP server install log	**Windows 2000** `%ORBDIR%\ext\HTTPServer\tksSi\tksInstall.log` **UNIX** `%ORBDIR%/ext/HTTPServer/tksSi/tksInstall.log`
Tivoli DAS server install log	**Windows NT and Windows 2000** `%DASDIR%\ext\das\server\tksSi\tksInstall.log` **UNIX** `%DASDIR%/ext/das/server/tksSi/tksInstall.log`

On UNIX systems, the **tail** command can be used to monitor the progress of the Tivoli Kernel Services log files.

```
tail -f stdout
```

Similar tools are available on Windows systems.

6

Configuring Logging

This section contains information to help you understand how to use logging and tracing in Tivoli Kernel Services. This section is organized as follows:

- Logging Overview
- Logging Concepts
- Logging Architecture
- Logging Service

Logging Overview

Logging refers to the text messages and trace data created by Tivoli Kernel Services components and applications. These messages and data are sent to an output destination, such as a file, a database, a console screen, and so on, for later inspection.

Typically, text messages are used to relay information on how the system or application is performing. Messages also are used to alert the system administrator to exceptional conditions when they occur. Messages are internationalized based on the locale of the message viewer, and can be filtered based on a number of criteria.

Trace data, on the other hand, is used to capture transient information about the current operating environment when a component or application fails to operate as intended. Tivoli Customer Support personnel use the trace information captured to

determine the source of an error or unexpected condition and to diagnose the problem. Tracing is generally not enabled by default and, like text messages, can be filtered based on a large number of criteria.

Logging Concepts

To help you understand and use the Logging subsystem, this section provides an explanation of its concepts and terminology.

Loggers

Loggers are software objects that record events that occur while a component or application is running. The Logging subsystem supports *message loggers*, which handle text messages, and *trace loggers*, which handle trace data.

Message Loggers

Message loggers are used to record text messages from a component or application. These messages are internationalized for individual locales. There are four basic types of messages, all of which are enabled by default:

Informational (TYPE_INFO)

Informational messages indicate conditions that are worthy of noting but that do not require the user to take any precautions, or perform an action.

Warning (TYPE_WARN)

Warning messages inform the user that a condition has been detected that they should be aware of, but does not necessarily imply that the user needs to take action.

A warning message might be issued if some aspect of a requested action did not perform as expected, but instead that the default processing for that action was taken.

Error (TYPE_ERR)

Error messages inform the user that a serious condition has been detected that they need to take

precautions or actions to correct. This could be a failure, a processing error, or an unexpected condition that needs to be investigated further.

An error message might be issued if a component failed to write information to a database, for example.

Fatal (TYPE_FATAL)

Fatal error messages inform the user that a catastrophic error has occurred in a component or application. Fatal messages require more aggressive action by the user, such as shutting down or restarting a component, terminating an application, or recycling an ORB.

A fatal message might be issued if a component was shut down due to a recurring error accessing a network connection.

The following are examples of messages from a message logger:

```
11:22:23.250 FNGOB0011I Using ORB Security Manager
11:22:24.244 FNGOB0007I Using JVM version 1.2.2 from IBM Corporation
```

Tivoli Kernel Services provides a default configuration for a message logger. The default message logger configuration and the Tivoli Message Standard are described in more detail later in this document.

Trace Loggers

Trace loggers are used to capture information about the operating environment when a component or application fails to operate as intended. The information captured by trace loggers is used by Tivoli support personnel and is not subject to internationalization requirements. Because trace messages are intended for support personnel, they are generally written to a file that can be viewed during a postmortem examination. The types of trace events supported by the Logging Subsystem are shown in Table 1 on page 68.

Handlers

Handlers are software objects that direct messages recorded by a logger to a destination. Messages can be directed to a file, a database, a console screen, and so on. Handlers are associated with a logger and a set of output destinations through configuration settings and logging commands issued by a user. Tivoli Kernel Services provides the configuration definitions for the following types of handlers:

Console Handler — Writes log records to a console (`system.out`)

File Handler — Writes log records to a file

Multifile Handler — Writes log records to a rotating set of log files

Serial File Handler — Writes log records

Database Handler — Writes log records to a database

Server Handler — Sends log records in batch mode to a remote Logging Server for processing.

Formatters

Formatters are software objects used to format the output of information contained in log records. In general, formatters can be used to tailor things like date and time stamps to local conventions. A single formatter can be used by multiple handlers.

The following configuration definitions of formatters are provided by Tivoli Kernel Services:

minimal — Displays the time stamp and the internationalized message. A minimal formatter is used for message logs.

maximal — Displays all the fields associated with the data. A maximal formatter is used for trace logs.

Filters

Filters are applied to loggers and handlers to control what message and trace types a logger or handler will process. When applied to a logger, the filter determines which types of message and trace records the logger will process. When applied to a handler, the filter determines which types of message and trace records the handler will send to the destination. Filters work by comparing a log record type with a bit-mapped filter mask and either accepting or rejecting the record based on the filter.

The following configuration definitions of filters are provided by Tivoli Kernel Services:

AllMaskFilter
: A log record can pass through this filter only if all of the bits of the record type are included in the filter's mask.

AnyMaskFilter
: A log record can pass through this filter if any of the bits of the record type are included in the filter's mask. This is the default filter used by Tivoli Kernel Services.

ExcludeMaskFilter
: A log record can not pass through this filter if any of the bits of the record type are included in the filter's mask. This can be applied to a logger or handler to prevent log records containing certain record types from being processed.

msgFilter
: Allows any message to pass through the filter.

  This is an **AnyMaskFilter** with a mask type of **TYPE_DEFAULT_MESSAGE**, which includes **TYPE_INFO**, **TYPE_WARN**, **TYPE_ERR**, and **TYPE_FATAL**.

trcFilter
: Allows the default trace entries to pass through.

This is an **AnyMaskFilter** with a mask type of **TYPE_DEFAULT_TRACE**. Refer to Table 1 to determine what trace types are included.

Table 1. Trace masks

Trace Type	Description
TYPE_ALL	Specifies a mask that includes all possible types.
TYPE_API	Specifies the method is an entry point for this product, component, or package.
TYPE_CALLBACK	Defines a callback method trace point. Callbacks are typically used in listener classes, which have registered with another object to be notified when a specific record occurs.
TYPE_DEFAULT_TRACE	Defines the default trace types. The default includes the following: ■ TYPE_API | TYPE_CALLBACK | TYPE_ENTRY | ■ TYPE_EXIT | TYPE_ERROR_EXC | TYPE_MISC_DATA | ■ TYPE_OBJ_CREATE | TYPE_OBJ_DELETE | ■ TYPE_PRIVATE | TYPE_PUBLIC | TYPE_STATIC | ■ TYPE_SVC
TYPE_ENTRY	Defines method entry trace points. An entry trace point should exist for every significant method to track movement through a component or application.
TYPE_ERROR_EXC	Defines an error or exception condition trace point.
TYPE_EXIT	Defines method exit trace points. An exit trace point should exist for every significant method to track movement through a component or application.

Table 1. Trace masks (continued)

Trace Type	Description
TYPE_LEVEL1	Defines a low-level trace point that provides very few details. This is provided for trace implementations that prefer the notion of a trace level, in which level 1 implies minimal detail, level 2 implies more detail, and level 3 implies the most detail. In such a system, these three types would be used exclusively. This trace type is not included in TYPE_DEFAULT_TRACE.
TYPE_LEVEL2	Defines a medium-detail trace point that provides more detail than TYPE_LEVEL1. This trace type is not included in TYPE_DEFAULT_TRACE.
TYPE_LEVEL3	Defines a high-detail trace point that provides the greatest amount of detail, compared to TYPE_LEVEL1 or TYPE_LEVEL2. This trace type is not included in TYPE_DEFAULT_TRACE.
TYPE_MISC_DATA	Defines a miscellaneous data trace point. This is used by a component or application to record miscellaneous information that does not fit into one of the other pre-defined types.
TYPE_NONE	Specifies a null record type that will not pass through a filter; that is, this record type is not logged.
TYPE_OBJ_CREATE	Defines an object creation, or constructor, trace point.
TYPE_OBJ_DELETE	Defines an object deletion, or destructor, trace point. This type would generally be used in a finalize method.

Table 1. Trace masks (continued)

Trace Type	Description
TYPE_PERF	Defines a performance-monitoring trace point. This type can be used to measure the execution time of selected sections of an application. This trace type is not included in TYPE_DEFAULT_TRACE.
TYPE_PRIVATE	Defines a private method trace point. Private trace points would be defined in methods with private scope; that is, not public, protected, or package.
TYPE_PUBLIC	Defines a public method trace point. This typically includes package and protected scope, as all of these methods may be used by other classes.
TYPE_STATIC	Defines a static method trace point.
TYPE_SVC	Defines a service code trace point. Service code is generally low-level code which provides commonly used services to other classes.

Logging Architecture

Throughout the rest of this document, it is useful at times to refer to loggers, handlers, filters, and formatters using a generic term that includes one or more of these software objects. The term *logging element* will be used when a reference to a particular kind of logging object, such as a formatter object, is not necessary.

Each ORB has a log manager associated with it. Log managers are responsible for creating and managing the logging elements. The handlers may be local or remote with respect to the originating application.

Local Logging

Figure 2 on page 71 shows the basic architecture of local logging and the logging elements. In a local logging environment, all components and logging elements are under the control of a single ORB. Tivoli

recommends that filters be applied to loggers because this keeps the configuration simple and prevents unused data from being passed to handlers.

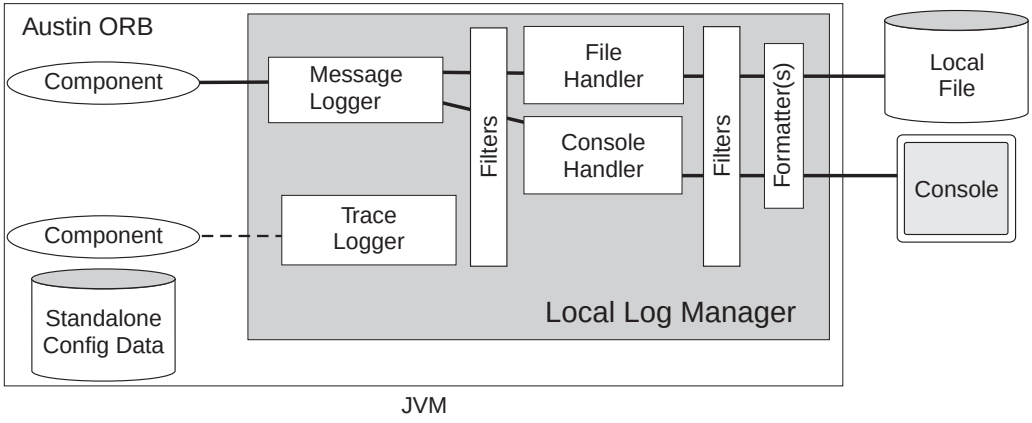

Figure 2. Local logging architecture

Default Message Logger Configuration

Figure 3 on page 72 provides a summary of the defaults for message loggers. Message logger output goes to one or more message files named `$ORBDIR/log/messagen.log` and to the console. The message logger uses a minimal formatter that only shows the timestamp and message text. It uses a `msgFilter` which enables logging for all message types.

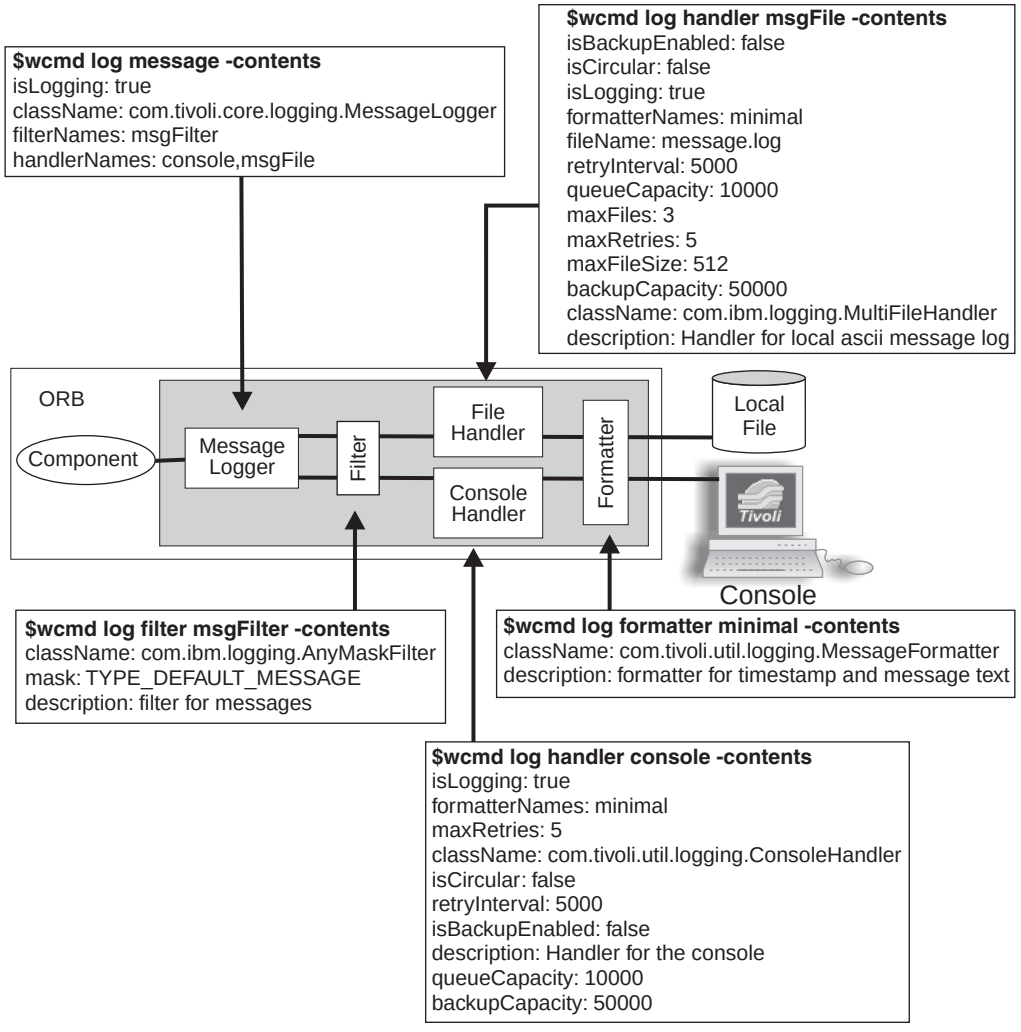

Figure 3. Default message logger configuration

Default Trace Logging Configuration

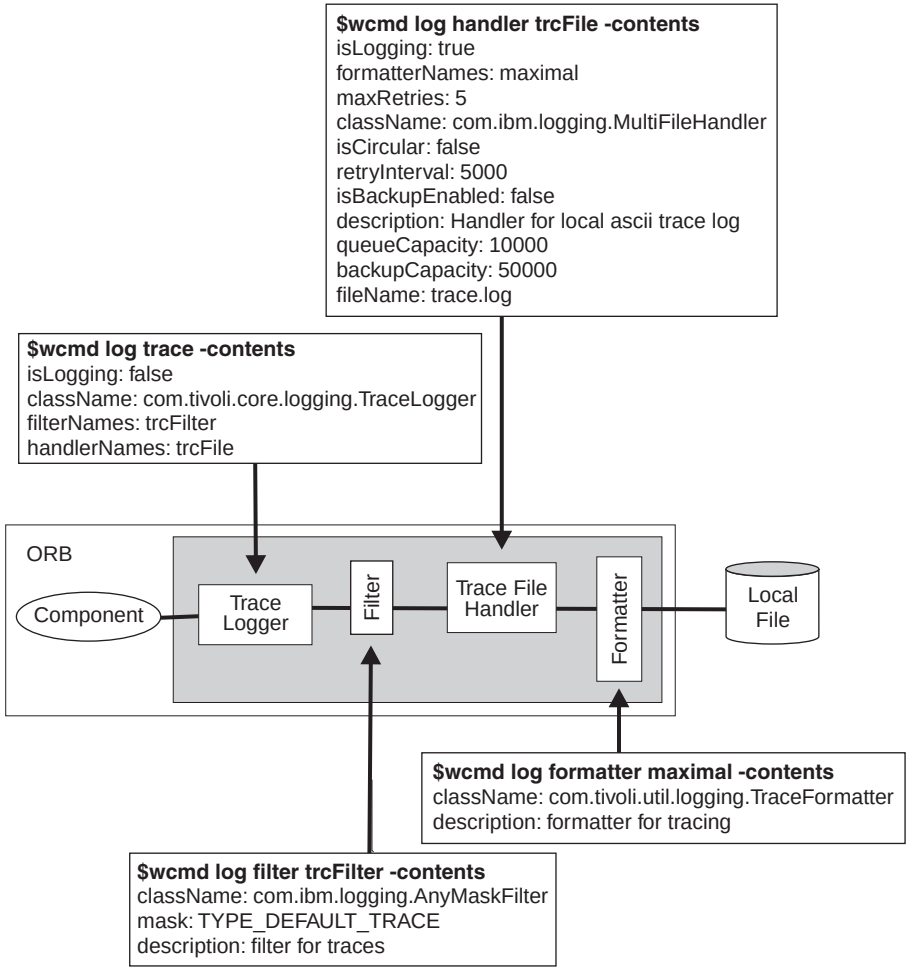

Figure 4. Default trace logger configuration

Figure 4 provides a summary of the defaults for trace loggers. Trace logger output goes to one or more files called `$ORBDIR/log/tracen.log`. The trace logger uses a trcFilter with a mask of TYPE_DEFAULT_TRACE. The trace logger uses a maximal formatter that shows the fields as illustrated in the example below.

```
2000.10.26 13:56:59.321
 com.tivoli.core.logging.DistributedLogManager
 getOrbAndElementName main
 MyCorp/rshah1.dev.tivoli.com_9990
  systemaccount system/services/principals/orb
  Entry, parm 1 = msgFile
```

The following is the explanation of the fields:

logDate	2000.10.26
logTimestamp	13:56:59.321
logClass	com.tivoli.core.logging.DistributedLogManager
logMethod	getOrbAndElementName
logThread	main
logServer	MyCorp/rshah1.dev.tivoli.com_9990 (This is the namespaceName/orbName.)
logClient	" "
logAccount	systemaccount
logPrincipal	system/services/principals/orb

The output shows that it is a trace point for the entry of method `getOrbAndElementName` and the argument for that method is `msgFile`.

Dynamic Configuration

All logging elements, and the logging service component itself, can be configured dynamically using either the Logging Command Line Interface (CLI) or the Tivoli Console. The Tivoli Console provides online help to aid you with configuring logging. For information about using the Tivoli Console for logging, refer to *Introducing Tivoli Kernel Services Administration* and to the online help in the Tivoli Console.

Information on using the logging CLI is provided in the following sections to help you gain familiarity with configuring the logging elements and the logging service.

Note: The wcmd examples in this section assume that the person issuing the commands is implicitly identified to Tivoli Kernel Services by the appropriate account and user information. If this is not the case, you must use the **-u** option on each wcmd and specify a Tivoli Kernel Services user ID that is authorized to perform the command. Additional information can be found in the *Tivoli Kernel Services Command Reference* document, in the section entitled *Security Implications for wcmd Commands*.

Structure of Logging Configuration

All the logging elements exhibit the property of coalescing. This means that there is a parent and child relationship. If a property on a child is not set then it inherits that property from the parent.

For example, Figure 5 on page 76 shows that the trace node/container has the property `isLogging=false`. Both its children, the log and directory, inherit this property. However, this property in the log group is overridden by setting it to true. Therefore the child of log, `managerTrace` has the behavior `isLogging=true`.

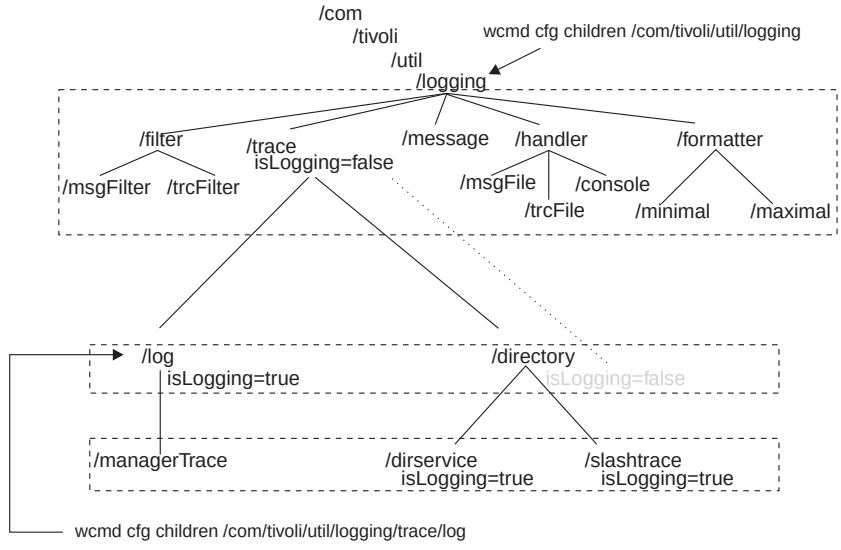

Figure 5. Logging configuration

The directory group does not override this property. This is because directory has a number of trace loggers, and they generate a lot of trace data. Instead, the individual trace loggers of directory called `dirservice` and `slashtrace` have the property `isLogging=true`.

Attention: It is important that you do **not** issue the following command, because it enables tracing for *every* trace logger in Tivoli Kernel Services and will result in severe performance degradation (or worse!):

```
wcmd log trace -set isLogging=true
```

Instead, issue one of these commands:

```
wcmd log trace -g log -set isLogging=true
wcmd log trace log log.managerTrace -set isLogging=true
```

In the above command the name of the trace logger includes its group. For example, it is `log.managerTrace` and its

directory.dirservice. When you issue any log CLI commands you need to include the name of the group followed with a dot and the name of the trace logger. You do not need to include the group with a dot for the wcmd cfg CLI commands.

In addition to the logging tree coalescing, there is also ORB coalescing in this order:

1. .orbdefaults: an ORB set that cannot be changed by the user. It contains the definitions provided by Tivoli Kernel Services and is read only.
2. .allorbs: an ORB set that can be customized by the end user
3. ORB sets: The logical collection of ORBs that you have defined
4. ORB: The ORB that you are working on

The following command shows how to enable tracing for log.managerTrace on .allorbs.

```
wcmd log -r .allorbs trace log.managerTrace -set isLogging=true
```

The resource -r can be any of these:

- This can be an ORB names (for example, orb.mymachine.mycorp.com_9990)
- This can be an ORB set (for example orbset.testset or .allorbs)
- This can also be an ORB OID. (for example, 3.af17bdf834eae996.1.d6caaf2223f8f495)

Using the Logging Command Line Interface

The logging CLI allows you to manipulate logging elements from the command line. A Log Manager responds to user requests for logging objects using the logging CLI. When users want to manipulate a logging object, they do so by specifying the name of the object they want to use. The following sections provide more detail about how to accomplish this.

Logging CLI Security

The logging CLI requires a lower authorization level than the configuration CLI and is the recommended method for manipulating logging elements. Refer to the `wcmd log trace` command in the *Tivoli Kernel Services Command Reference* for the authorization requirements.

Enabling Tracing of a Component

If you notice a problem in the message log and you discover that it is in the log component because the message ID had FNGLG. How do you know the names of the trace loggers of log component to enable?

1. Use the `wcmd log ls` command to list the names of the currently active trace loggers for the local ORB. Note this command lists all the instantiated logging elements. To list the names on a different machine, you can use this command:

   ```
   wcmd -i <orb id> log ls
   ```

2. For example, you can use the following command to determine the active message and trace loggers for the component log.

   ```
   wcmd log ls | grep log
   ```

3. Now, if you want your trace to go to the default trace file called trace1.log, all you have to do is enable tracing for the logging group:

   ```
   wcmd log trace -g log -set isLogging=true
   ```

4. You can verify this by using this command:

   ```
   wcmd log trace -g log -contents
   ```

5. The example below shows how to turn on tracing for the trace logger called `log.managerTrace`.

   ```
   wcmd log trace log.managerTrace -set isLogging=true
   ```

Using the Configuration Command Line Interface

The configuration CLI also allows you to manipulate logging elements from the command line. Refer to the `wcmd cfg` commands in the *Tivoli Kernel Services Command Reference*. It is recommended that you do not use the `wcmd cfg` CLI for enabling trace loggers. Use the `wcmd log` commands instead. You will need to

use the `wcmd cfg` commands when creating nodes that define new handlers, formatters and filters. The `wcmd cfg` CLI also allows you to browse the nodes of the `/com/tivoli/util/logging` tree.

Handler Configuration

Tivoli Kernel Services provides default definitions for the following handlers:

console	Output goes to the console (`system.out`)
msgFile	ASCII message multi file: `$ORBDIR/log/messagen.log`
trcFile	ASCII trace multi file: `$ORBDIR/log/tracen.log`
serialMsgFile	Serial message file: `$ORBDIR/log/serialMsgn.log`
serialTrcFile	Serial trace file: `$ORBDIR/log/serialTrcn.log`
logFile	ASCII message file: `$ORBDIR/log/loggingServicen.log`
logServerHandler	Handler to write to the logging service. See "Logging Service" on page 96 for more information.
log.dbHandler	Handler to write message logs to the database. See "Creating a Logging Database After Installation" on page 90 for more information.
log.dbReader	Handler to read and write message logs from the database.
log.dbDelete	Handler to write, read and delete from the database.

Getting a List of Handler Definitions

To get a list of all the handler definitions, issue the following command:

```
wcmd cfg children /com/tivoli/util/logging/handler
```

A list of the handlers currently defined, but not necessarily instantiated, is provided. A sample output looks like this:

```
/com/tivoli/util/logging/handler:

/com/tivoli/util/logging/handler/logServerHandler
/com/tivoli/util/logging/handler/SerialGlobalMsgHandler
/com/tivoli/util/logging/handler/ConsoleMsgHandler
/com/tivoli/util/logging/handler/SerialGlobalTraceHandler
/com/tivoli/util/logging/handler/console
/com/tivoli/util/logging/handler/RasPfSerialTraceHandle
/com/tivoli/util/logging/handler/FWPConsoleErrMsgHandler
/com/tivoli/util/logging/handler/RasPfMFileTraceHandler
/com/tivoli/util/logging/handler/log
/com/tivoli/util/logging/handler/logFile
/com/tivoli/util/logging/handler/msgFile
/com/tivoli/util/logging/handler/serialMsgFile
/com/tivoli/util/logging/handler/sec
/com/tivoli/util/logging/handler/MFileGlobalMsgHandler
/com/tivoli/util/logging/handler/trcFile
/com/tivoli/util/logging/handler/serialTrcFile
/com/tivoli/util/logging/handler/ConsoleTraceHandler
/com/tivoli/util/logging/handler/MFileGlobalTraceHandler
```

The handler definitions listed do not imply that the handler has been instantiated.

Adding Your Own Handler

The Tivoli Kernel Services logging subsystem provides handler classes that allow you to direct logger output to any number of destinations. Figure 6 on page 81 shows handler classes and illustrates the attributes and inheritance hierarchy for each type. Handlers are provided to direct log reports to databases, consoles, log servers, and to a variety of different file types.

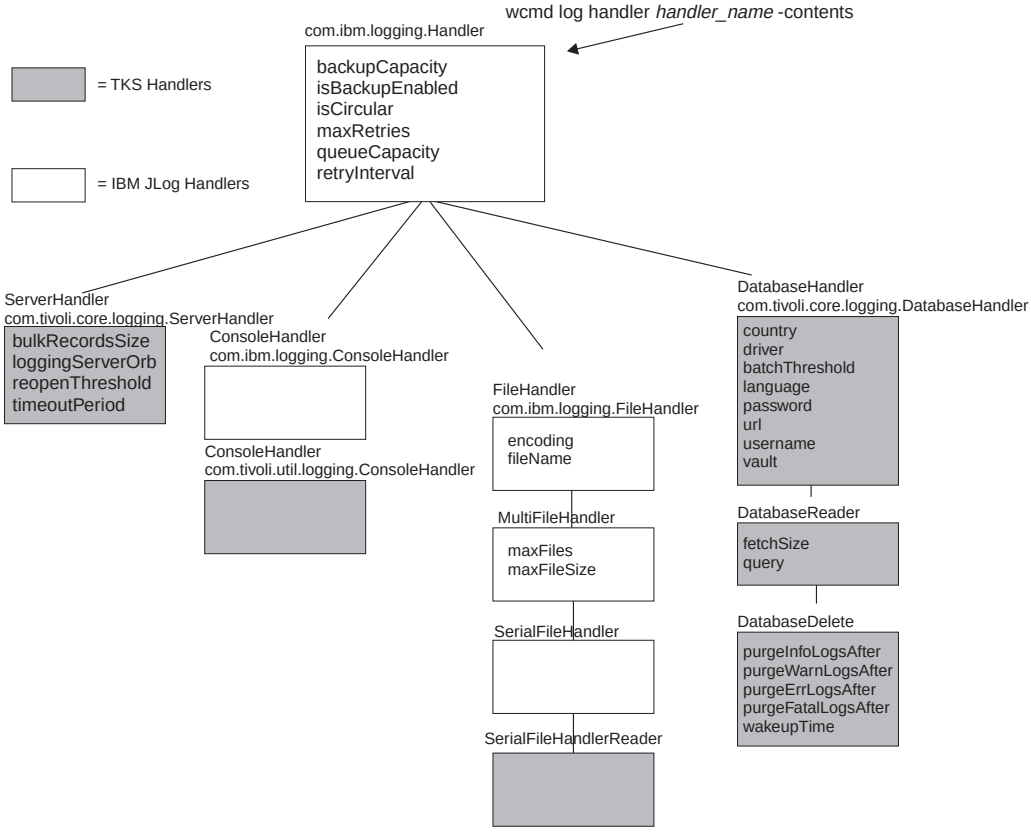

Figure 6. Handler classes and their properties

If you do not want the trace to go to the default trace file called `trace1.log`, then you have to setup your custom multifile handler.

1. Create a node for your handler and specify the properties that are needed for a multifile handler using the `wcmd cfg put` command. The following are the necessary properties for setting up a `MultiFileHandler`.

   ```
   wcmd cfg put /com/tivoli/util/logging/handler/<name you want>
   className=com.ibm.logging.MultiFileHandler
   fileName=<name of your file>
   formatterNames= maximal
   ```

   For example:

```
wcmd cfg put /com/tivoli/util/logging/handler/directoryHandler
   className=com.ibm.logging.MultiFileHandler
fileName=directory.log
formatterNames=maximal
```

2. Verify that you set all the properties correctly using the following command:

   ```
   wcmd log handler directoryHandler -contents
   ```

   By default, the `com.ibm.logging.MultiFileHandler` writes log records to three rotating log files, instead of to just one file, which is the case with other handlers. Each of the rotating files are 512 KB in size, by default. The size of the files can be configured as necessary. The logs are written in the `$ORBDIR/log` directory. Also, we are attaching a maximal formatter to the handler. Note, that Tivoli Kernel Services provides a definition for the maximal formatter.

Attaching a Handler to a Logger

To attach a handler to a logger, use this command:

```
wcmd log trace <-g group or just a single component>
 -set handlerNames=<container you used above>
```

For example:

```
wcmd log trace -g directory -set handlerNames=directoryHandler
```

Defining Your Own Filter

If you do not want to use the default trace filter called `trcFilter` then you have to setup your custom filter:

1. Create a node for your filter and specify the properties that are needed for a filter using the `wcmd cfg put` command. The following are the necessary properties needed to setup a `JLog AnyMaskFilter`.

   ```
   wcmd cfg put /com/tivoli/util/logging/filter/<name you want>
   className=com.ibm.logging.AnyMaskFilter
   mask=
   ```

   For example:

```
wcmd cfg put /com/tivoli/util/logging/filter/directoryFilter
className=com.ibm.logging.AnyMaskFilter
mask= DEFAULT_TRACE_MASK TYPE_PERF
```

2. The `directoryFilter` also allows you to trace performance trace points. You can verify that you set the correct properties by issuing the following command:

   ```
   wcmd filter directoryFilter -contents
   ```

Attaching a Filter to the Logger

To attach your filter to a logger, use this command syntax:

```
wcmd log trace <-g group or just a single component>
   -set filterNames=<container you used above>
```

For example:

```
wcmd log trace -g directory -set filterNames=directoryFilter
```

Defining Your Own Formatter

If you do not want to use the minimal or maximal formatter defined by Tivoli Kernel Services then you have to setup your custom formatter following this procedure:

1. Create a node for your formatter and specify the properties that are needed for a formatter using the `cfg put` command. The following are the necessary properties needed to setup a formatter:

   ```
   wcmd cfg put /com/tivoli/util/logging/formatter/<name you want>
   className=com.tivoli.util.logging.MessageFormatter

   wcmd cfg put /com/tivoli/util/logging/formatter/directoryformatter
   className=com.tivoli.util.logging.MessageFormatter
   separator=$
   logServer=true
   ```

2. Verify that you set all the properties correctly using the following:

   ```
   wcmd log formatter directoryformatter -contents
   ```

See the information on the `wcmd log formatter` command in the *Tivoli Kernel Services Command Reference* for a list of possible formatter settings and keys.

Enabling Bootstrap Tracing

What do you do if the ORB does not come up and you do not see any logs called `message1.log in $ORBDIR/log`? You can turn on tracing for the directory, configuration, network security and logging components. The following steps will allow you to turn on tracing for the initial boot sequence:

1. Edit `LOCKCfg.properties` in `$ORBDIR/bin/`*`interp`*.
2. Add the following property to `JVMARGS`:

   ```
   -Dcom.tivoli.util.logging.bootTrace=true
   ```

3. The message and trace will go to the `bootstrap.log` file in the `$ORBDIR/log` directory.

Changing Your Log File Location

It is possible to change the location of the logging directory from `$ORBDIR/log` to any other location you desire.

1. You need to edit `LOCKCfg.properties` in `$ORBDIR/bin/`*`interp`*.
2. Change `LOGDIR` to point to where you want to store the logs.

Remote Handlers

Figure 7 on page 85 depicts a distributed logging scenario where you use a remote handler. In a distributed logging environment, components can run in one ORB and the logging elements that process the messages from that component can reside in another ORB. How you distribute logging elements will depend on the complexity of your Tivoli Kernel Services installation and customer requirements. For example, loggers can run in the same ORB as the component that uses them and the handlers that deliver logged messages can reside in distant ORBs.

This figure illustrates how a component in the Austin node can use local loggers for messages and trace data and have the trace data recorded to a local file as well as having it sent to a central location. In this example, text messages would be sent to the local console, a local file, or both. For simplicity of illustration, filters are not shown in the figure.

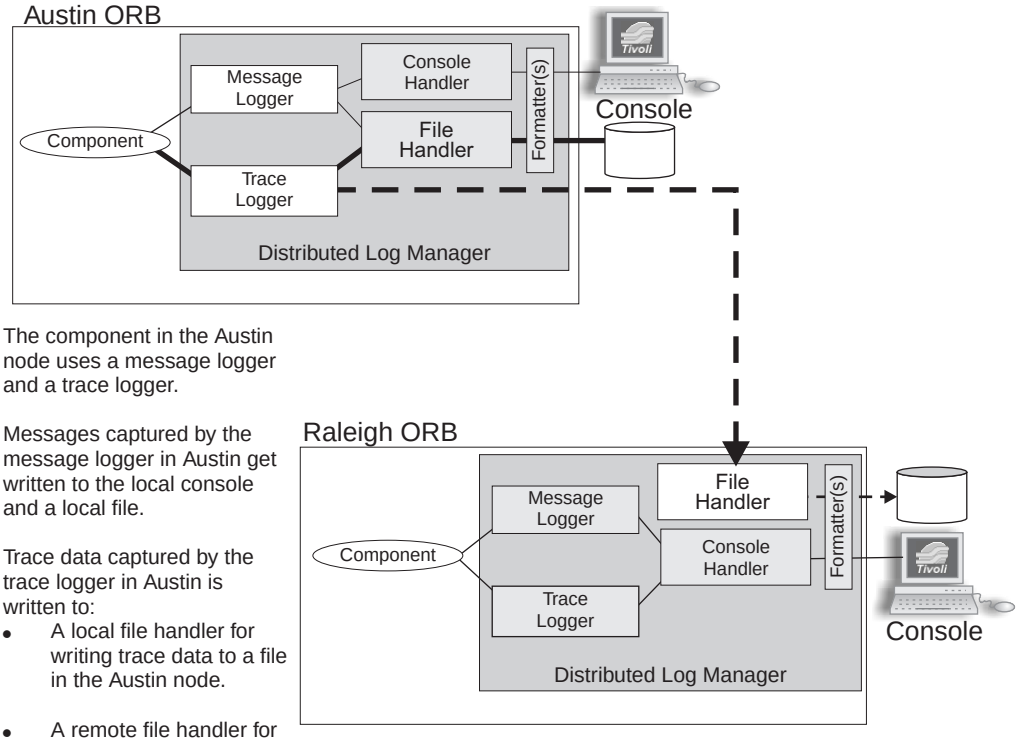

Figure 7. Remote handler

Systems Administrators can change the association between loggers and handlers and between formatters and handlers using CLI or GUI operations. For example, in Figure 7, an administrator could attach a local console handler or file handler to the trace logger and also attached to handlers in a remote ORB to enable logging of trace data to two destinations. A message logger could also be similarly configured, but for simplicity of illustration, only the trace logger is depicted as logging trace data to two locations.

How to Use a Remote Handler

The following command lets you attach a remote handler:

```
wcmd log trace <yourTraceLogger>
   -set handlerNames=<orbName/yourHandler>
```

For example:

```
wcmd log trace log.managerTrace
   -set handlerNames=rshah1.dev.tivoli.com_9990/trcFile
```

You can use the following command to determine the names of all the ORBs in your namespace:

```
wcmd info getOrbs
```

Turning Off Logging of a Component

To stop logging, follow this procedure.

1. Before you change the logging values, view the current values by issuing these commands:

   ```
   wcmd log handler console -contents
   ```

   ```
   wcmd log trace <traceLoggerName> -contents
   ```

2. Return the logging to the default values

   ```
   wcmd log handler console -set formatterNames=minimal
   wcmd log trace <traceLoggerName> -set isLogging=false
   ```

   An optional command:

   ```
   wcmd log trace <traceLoggerFile> -set handlerNames=trcFile
   ```

Tivoli Message Standard

Messages issued by Tivoli Kernel Services and Tivoli Presentation Services adhere to the Tivoli Message Standard. The Tivoli Message Standard specifies the standard format for messages issued from a Tivoli component or application and is intended to provide a consistent and meaningful way for identifying messages across the entire Tivoli product suite.

Message ID Format

The message ID format for Tivoli Kernel Services is **XXXYY####Z** where:

XXX Is the three character product prefix for the component or application producing the message. For Tivoli Kernel

Services, the product prefix is **FNG**. For Tivoli Presentation Services, the product prefix is **FWP**.

YY Is an optional 1 or 2 character alphabetic subsystem code. Tivoli Presentation Services does not provide a subsystem code. The Tivoli Kernel Services subsystem codes are based on the component, as shown below.

Component	Subsystem Code
Component	CP
Configuration	CF
Console	MC
Data Storage	DS
Directory	DR
Failover	FT
Gateway	GW
Logging	LG
Messaging Service	TS
ORB	OB
Planning and Distribution	PD
Security	SC
Service Manager	SM
SNMP	SP
Time Synchronization	TM

Is the required message number. The message number consists of 4 numeric digits. Message numbers are unique per subsystem code (**XX** value).

Z Is the message type.

- **I** Informational (TYPE_INFO)
- **W** Warning (TYPE_WARN)
- **E** Error (TYPE_ERR)
- **F** Fatal (TYPE_FATAL)

Here are some sample Tivoli Kernel Services messages.

```
15:06:25.457 FNGCP0031I Started component dirservice version 5.1.0-200011262259.
15:06:25.457 FNGOB0004I ORB transition from state 2 to state 3.
15:06:28.231 FNGSC4001I Root Cache Service started.
15:06:28.261 FNGCP0027I Starting component SecurityCacheService version 5.1.0-200011262259
15:06:28.441 FNGCP0031I Started component SecurityCacheService version 5.1.0-200011262259.
```

The second message, with the message number of **FNGOB0004I** can be interpreted as:

- Being issued from Tivoli Kernel Services (product prefix of **FNG**)
- Being issued from the ORB component (subsystem code **OB**)
- Message number **0004**
- Informational message (message type **I**)

Late Binding of Messages

The `DistributedLogManager` will allow late binding of the messages. The logging subsystem passes records between loggers and handlers. The records contain the base message file name and the message key. The actual binding of the message happens when formatting the message. Binding occurs where the handler resides. Therefore, the message is generated in the locale of the Handler.

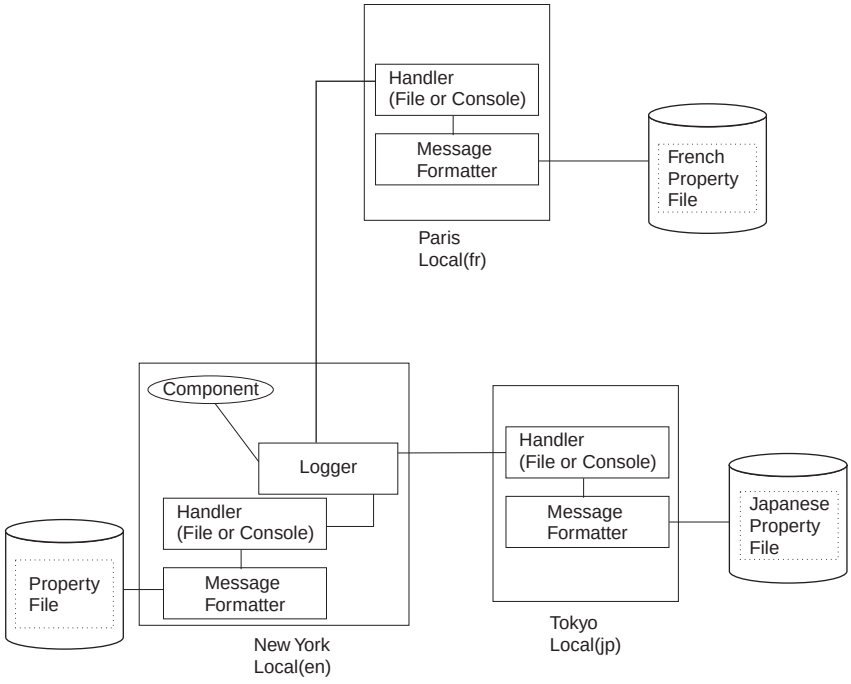

Figure 8. Binding of messages

Figure 8 shows an example of a generating a message in a different locale than where it was logged. In the figure:

1. A component in the New York region issues a log message destined for a file or console in the Paris region.
2. The logger in the New York region receives the request and the New York ORB sends the message to the handler and formatter in Paris.
3. The formatter in Paris looks for a local copy of a property file with French messages. If it finds one, the message is routed to the Paris logging console or logging file.

If a local property file `(locale(fr))` is not found, the Paris ORB's classLoader sends a request to the Installation Depot (which, in this example, exists in the New York ORB's node) to send the French messages property file to the Deployment Depot in the French node

where the messages are cached and accessed by the message formatter. The message is extracted from the cached file and is displayed on the Paris console or written to the Paris log file.

Creating a Logging Database After Installation

If you plan to use a database handler to log your messages, you need to have a logging database created and initialized. If you used the **tdacl.bat** script, or the **tdacl.sh** script and took the default values, a logging database called **logging** and a security audit database called **SECAUDIT** were created for you.

However, if you chose *not* to create a logging database using the **tdacl.sh** script, you can create a logging database after installation by using the **crtlogdb** script.

Tivoli Kernel Services supports the use of multiple logging databases, you can use the **crtlogdb** script to create a second or third logging database as well.

To create a logging database, use the following procedure to run the **crtlogdb** script:

1. Log into the DB2 database server machine using your DB2 user ID and password.
2. Open a DB2 command window.
3. Change to the `util` directory on the installation CD-ROM.
4. Run the **crtlogdb** script to create the logging database.

   UNIX `../crtlogdb.sh`

   Windows `crtlogdb.bat` *`DB2user DB2pw DBname option`*

   where:

DB2user	is your DB2 user ID
DB2pw	is your DB2 password
DBname	is the name of the logging

database you want to create. This is the name of the database in DB2.

The default logging database is called **logging** and the default security audit database is **SECAUDIT**. If you are creating another database, choose a different name for it.

Note that the database name must be 8 characters or less in length.

option is one of the following:

database creates the database

tables creates the schema

all creates the database and its schema

If you get an error message indicating that you do not have the authority to perform the requested action, log on using a DB2 user ID and password with the proper authority and then retry the **crtlogdb** script.

Using a Logging Database

This section describes how to configure the log handler to work with the Tivoli Data Access Services (DAS) server. Before using this procedure, you need to have created a logging database by running either:

- the **tdacl** script (See "Creating Tivoli Kernel Services Databases" on page 14), or
- the **crtlogdb** script (see "Creating a Logging Database After Installation" on page 90)

Complete these instructions before configuring a database handler using the Tivoli Console tasks; otherwise you will receive an error

and the Tivoli Console tasks will not function properly. Note that the vault file must be created on the same ORB from which you are going to be logging. For example, if you want to log from the Tivoli Console running on the installation depot machine, you must create the vault file on orb.2.

To create an alias for the database for Tivoli Data Access Services (DAS), do the following:.

1. Stop the ORBs.

   ```
   wcmd orb shutdown
   ```

2. Stop the Tivoli Data Access Services (DAS) server.
3. Go to the location on the Tivoli DAS server machine where the Tivoli DAS server software was installed.
4. Modify the `dasDsnCfg.tks` file as follows:

   - Under the `[$SOURCES]` heading, add the following line, substituting the name of your DB2 database for *DBname*:

     DBname`=ENABLED`

   - Add a new section at the bottom of the file, replacing *DBname* with the name of your DB2 database that you created with **crtlogdb**.

     ```
     [DBname]
              UID=<DB2 user ID>
              DBMS=DB2/2
              PWD=XV&UXa]b
              DB=DBname
              DEFAULT=FALSE
              QUAL=
              DRV=DB2CLI
     ```

     Note: The name of the stanza, [*DBname*] is the Tivoli DAS data source name and is case sensitive. This document assumes that the Tivoli DAS server data source name is the same as the DB2 database name, but that is not required.

5. Modify the dasOrbCfg.tks file as follows:

   - Add the name of your Tivoli DAS data source to the existing data source line.

```
DAServer.datasources=slash,tmd,ps,DBname
```

- Add the following lines at the bottom of the file, replacing *DBname* with the name of your Tivoli DAS data source:

```
DAServer.datasource.DBname.use_xa=no
DAServer.datasource.DBname.threadpool=20
DAServer.datasource.DBname.queue_size=102
```

6. Restart the Tivoli DAS server.
7. Create the vault file for the logging database by entering the **wcmd dir setdbinfo** command on the installation depot. When the command prompts you for the logging database alias, make sure your entries match identically (including case) the values you added to the Tivoli DAS configuration file.

   This command allows you to create a database specifically for the logging service, a security audit database, or a general logging database. You also must run this command on all ORBs where a database handler may be configured. For example, to run this command on ORB 1 you would enter:

   ```
   wcmd dir setdbinfo -o 9990
   ```

   To run this command on ORB 2 you would enter:

   ```
   wcmd dir setdbinfo -o 9991
   ```

   The `wcmd dir setdbinfo` command must be run on the ORB from which you want to log. For example, if you are doing logging from the Tivoli Console running on the installation depot, be sure to issue this command on orb.2

   Select **Logging Service datastore** to create the logging service database using a default vault file of `loggingVault`. Select **Security audit datastore** to create a security audit database using a default vault file of `secAuditVault`. Otherwise, select **Other** and enter your own name for the vault file when prompted.

   After selecting the type of datastore, answer the remaining prompts:

- Enter the fully-qualified name of the Tivoli DAS server when prompted for the database server.
- Specify the port number to use if it is not the default of 8001, otherwise just press Enter.
- Set the database alias to the Tivoli DAS data source name, in this example **logging**.

  Note: The database alias is case sensitive and must be specified the same way as it was defined in the `dasDsnCfg.tks` file.
- Enter your DB2 user ID and password. Press Enter to use the default of **db2admin.**
- Confirm that you are using the default database driver of **com.tivoli.das.jdbc.daDriver**

and press Enter.

8. If you created a general logging database, you must map the vault file to the specific log handler. (This mapping occurs automatically for the logging service database and the security audit database.) Map the vault file to the log handler by issuing the following command, all on one line, specifying your log handler and vault file:

```
wcmd log handler <databaseHandlerName>
-set vault=<vaultFileName>
```

Converting Serialized Log Files to ASCII Files

Tivoli Kernel Services stores some log files as serialized files. You can convert these serialized files into ASCII file format. To convert these files, Tivoli Kernel Services provides a batch file and a shell script.

Setting Up Your Environment

To use this tool, set up your environment as follows:

1. Make sure that the JAR file needed to convert the logs is set as an environment variable. This variable, JARDIR, needs to be set to the following path:

Windows	`C:\Tivoli\orb.1\cd`, where `C:\Tivoli` represents the directory where you installed Tivoli Kernel Services.
UNIX	`/Tivoli/orb.1/cd`, where `/Tivoli` represents the directory where you installed Tivoli Kernel Services.

. This allows log@2.0.2.jar to be used to convert the files.

2. Run **setupEnv.sh** on UNIX systems, or **setupenv.bat** on Windows systems, on the ORB that contains the logs you want to convert.
3. If you are converting serialized message logs, all of the resource bundles for those message logs must be in your CLASSPATH.

   Note: The resource bundles required by Tivoli Kernel Services are already in the CLASSPATH. To process logs generated by a Tivoli application, you might need to add the appropriate resource bundles to the CLASSPATH.
4. Move the batch file or shell script into the same directory as the file you are converting.

   The **convertSerialLog** shell script is located in the following directory:

Windows	`<`*Tivolidirectory*`>\orb.1\bin\w32-ix86`
UNIX	`<`*Tivolidirectory*`>/orb.1/bin/generic`

   where <*Tivolidirectory*> is the directory where you installed Tivoli Kernel Services.

   The serialized log files are located in the `orb.`*n*`/log` directory by default.

Converting the Files

After the environment is set up, you can convert the files by entering the following command, all one line:

```
convertSerialLog <base name of serialized log files>
    <output ASCII file>
```

For example, to convert the following serialized log files and place the result into the ASCII file `output.txt`:

```
GlobalSerialMsg1.log
GlobalSerialMsg2.log
.
.
GlobalSerialMsgN.log
```

Enter the following command:

```
convertSerialLog GlobalSerialMsg.log output.txt
```

Note that you only specify the *base name* of the serialized log files on the command. Omit the serial log number (the number appended to the base name.)

Logging Service

Tivoli Kernel Services has the capability of distributing its logging service. You can centralize your logging under a single operator. Or, if your installation includes a remote branch location that is geographically separated from the installation depot, then having a local operator monitor logging information is probably adviseable. And certainly, as your installation grows, the need for multiple logging services becomes more accute. You might want to deploy mulitple instances of the logging service and the `logServerHandler`.

The logging service and the `logServerHandler` can be used to put messages from different ORBs in a single (remote) log. Tivoli recommends that messages be put in a single (remote) log. Trace data should not be put in this log since it will degrade the performance of the system. You should use local logs to collect trace data. By default message Tivoli Kernel Services logs are written to the console and local file called message1.log. Every ORB must use a local `logServerHandler` to send messages to a logging service. By default the `logServerHandler` queues 10 log records on the local side and then send them to a remote logging service. This handler provides error handling for remote exceptions. The remote exceptions could be due to the following reasons:

- The network is down,

- The ORB is down.
- The logging service is down.

When a remote exception occurs, the `logServerHandler` will write to a local serialized file until it can write again to a logging service.

When to Use the Logging Service

You should use the logging service in these situations:

- If it is desirable to have all of the log messages collected in a logging database. For instance, perhaps if you want to keep a record of all the messages from all the ORBs across the system in a single place. Then you should use the logging service.
- Depending on the size of your installation and how many messages are produced, you may need multiple instances of the logging service to be deployed.

How to Use a Single Logging Service

The logging service is started on the ID by default. You can verify that by looking for message similar to the one below:

```
14:32:13.681 FNGCP0031I Started component logging
    version 5.1.1-200010142257
```

You can configure all ORBs to send their output to the logging service using the following Logging commands. For example, configure the default message logger to write messages to a local `logServerHandler`. This `logServerHandler` will connect to the logging service. The following command configures the output from all message loggers on all ORBs to be sent to the console, a local file called message1.log and the logging service. This is a very powerful command as it will affect every ORB. Be careful when using `-r .allorbs`. Remember that you will need the `logConfigure` role for this command to succeed:

```
wcmd log message -r .allorbs
    -set handlerNames=console,msgFile,logServerHandler
```

If you have a logging database installed, you can direct all the output from a logging server to the database using the following command. You need to issue this command to the ORB where the

logging service is running. Note, that we are using a log.dbDelete handler because we want to the ability to delete the logs every 7 days.

```
wcmd log message log.serverLogger -set handlerNames=log.dbDelete
```

You can verify that the data is in the database using the Tivoli Console log viewer or connecting to the database and issuing the following command.

```
Select message_text from fng_logdata
```

When To Use Multiple Logging Services

A single logging service could be sufficient for small installations with perhaps a hundred or so servers. However, it would not scale for a large system. In this case, you want to deploy multiple instances of the logging service. The multiple logging services can send their output to a single logging database as shown in Figure 9 on page 99.

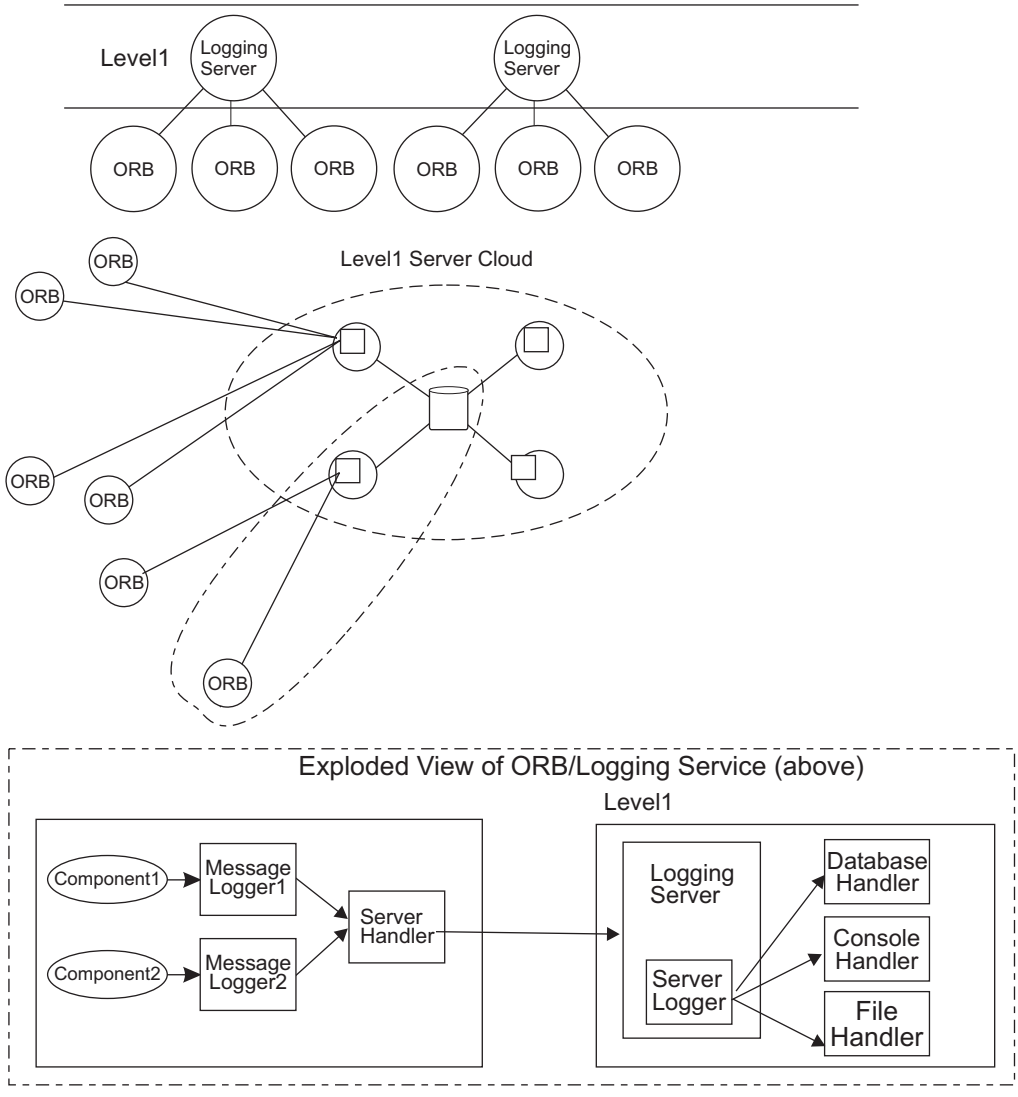

Figure 9. Using multiple logging services

How to Deploy Another Logging Service

Follow these steps to deploy a second logging service:

1. You can deploy the logging service from the installation depot ORB only. Therefore, if you are trying to install it on a bootprint ORB, you need to know the name of the installation depot ORB. Issue this command on the bootprint ORB to get a list of all your ORBs:

   ```
   wcmd info getOrbs
   ```

2. To determine the version of the logging service that needs to be deployed, direct the following command to the installation depot ORB:

   ```
   wcmd -i <idOrbName> svc ls LSM
   ```

3. Deploy the logging service on the bootprint ORB from the installation depot ORB. Use the version that you determined from step 2 and which is represented by *v.r.m* in the command:

   ```
   wcmd -i <idOrbName> cds deploy logging@v.r.m <bootprintORBname>
   ```

4. Start the logging service on the bootprint ORB using this command, substituting the appropriate version for *v.r.m* in the command:

   ```
   wcmd accmgr startService -s logging -v v.r.m
   ```

5. Check if the service started on the bootprint ORB:

   ```
   wcmd svc ls LSM
   ```

6. You need to set up the logging database, according to the instructions in “Creating a Logging Database After Installation” on page 90.

7. Then attach the database handler to the message logger associated with the logging service:

   ```
   wcmd log message log.serverLogger -set handlerNames=log.dbDelete
   ```

How to Configure a logServerHandler to Use a Specific Logging Service

By default, the `logServerHandler` gets the logging service from the service manager. However, when you have multiple instances of a logging service, we may want to explicitly specify which set of ORBs will use which logging service. You need to complete the following steps to specify which logging service to use.

1. Identify the ORBs that need to use a specific logging service. Create an ORB set that includes those ORBs. You can execute this command on the bootprint ORB:

   ```
   wcmd info crtorbset loggingset
   ```

2. You can get the names and OIDs of all the ORB using the following command:

   ```
   wcmd info getOrbs -oid
   ```

3. You need to specify the orbName for the loggingServerOrb.

   ```
   wcmd log handler -r loggingset logServerHandler
    -set loggingServerOrb=<orbName>
   ```

4. Add the ORBs to this ORB set. For example,

   ```
   wcmd info joinOrbset <orbset> <orb>
   ```

   Where <orb> can be the name of the ORB.

How to Chain Logging Services

To chain logging Services, we have to attach the logServerHandler to the log.serverLogger of a level1 LoggingServer and so on as shown in Figure 10 on page 102.

1. Attach the logServerHandler to the Level 1 logging service's message logger:

   ```
   wcmd log message log.serverLogger -set handlerNames=logServerHandler
   ```

2. Specify the orbName of the next level (Level 2) logging service:

   ```
   wcmd log handler -r loggingset logServerHandler
    -set loggingServerOrb=<orbName>
   ```

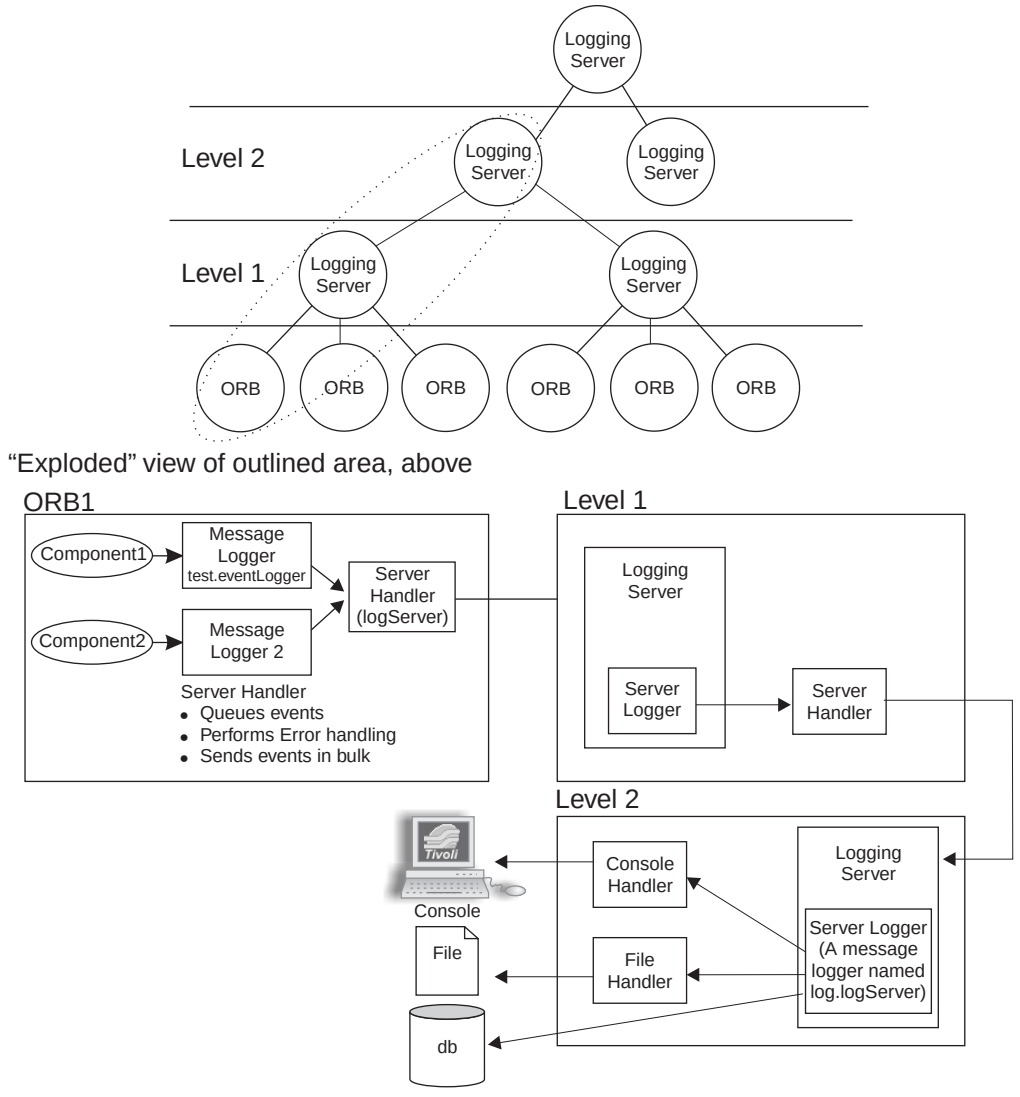

Figure 10. Chaining the logging services

7

Uninstalling Tivoli Kernel Services

This chapter contains the procedure for uninstalling the installation depot, a bootprint server, a Tivoli Data Access Services (DAS) server, or an MQSeries server.

Uninstalling the Installation Depot

The installation depot uninstall program is used to remove the installation depot software from the machine.

The Tivoli Data Access Services (DAS) server is uninstalled if it was installed using the single machine installation option. Otherwise, you must manually remove the Tivoli DAS server.

The MQSeries server also is uninstalled if it was installed using the single machine installation option. Otherwise, you must manually uninstall the MQSeries server.

Note: You must uninstall the MQSeries server only after all Tivoli Kernel Services ORBs have been stopped.

Windows 2000

1. Stop all ORBs running on the installation depot. See "Stopping an ORB" on page 56 for details.
2. Select **Start → Programs → Tivoli Kernel Services 1.1.0 → Uninstall**

3. Click **Uninstall** to remove the installation depot.

AIX and Solaris

1. Stop all ORBs running on the installation depot. See "Stopping an ORB" on page 56 for details.
2. Open a command window and go to the location where the installation depot software was installed.
3. Go to the `uninstall` directory.
4. Enter the following command:

   ```
   ./juninst
   ```

Uninstalling a Bootprint Server

To remove a bootprint server:

Windows

1. Stop the ORB running on the server. See "Stopping an ORB" on page 56 for details.
2. Select **Start → Programs → Tivoli Kernel Services 1.1.0 Bootprint → Uninstall**
3. Click **Uninstall** to remove the bootprint server.

UNIX

1. Stop the ORB running on the server. See "Stopping an ORB" on page 56 for details.
2. Open a command window and go to the location where the bootprint software was installed.
3. Go to the `uninstall` directory.
4. Enter the following command:

   ```
   ./juninst
   ```

Uninstalling a Tivoli DAS Server

To uninstall a Tivoli Data Access Services (DAS) server, run the appropriate script, all on one line:

Windows

```
<DASInstallDirectory>\ext\das\server\tksSi\
  uninstallDasServer.bat
```

UNIX

```
<DASInstallDirectory>/ext/das/server/tksSi/
uninstallDasServer
```

substituting the name of the directory where you installed the Tivoli DAS server software for *<DASInstallDirectory>*.

Uninstalling an MQSeries Server

To uninstall an MQSeries server, use the following steps:

Note: Only uninstall the MQSeries server after all Tivoli Kernel Services ORBs have been stopped.

1. Stop all of the MQSeries processes in the following order using the designated commands:
   a. Enter `endmqcsv -i <queueMgr>` to stop the MQSeries Command Server.
   b. Enter `endmqm -i <queueMgr>` to stop the MQSeries Queue Manager.
   c. Enter `endmqlsr -m <queueMgr>` to stop the MQSeries Queue Listener.
   d. The MQSeries Trigger Monitor stops automatically.
2. **Windows**
   a. Enter
      ```
      net stop "IBM MQSeries"
      ```
      to stop the MQSeries Service.

b. Enter

```
net stop "Tivoli Kernel Services MQRestarter"
```

to stop the MQSeries Restarter Service.

c. Enter

```
<MqInstallDir>\tksSi\bin\tksSvc.exe
removeService "Tivoli Kernel Services MQRestarter"
```

to remove the MQSeries Restarter Service, where **MqInstallDir** is the directory where MQSeries is installed.

d. Enter

```
<MqInstallDir>\uninst\amqiunin.exe
```

to run the MQSeries uninstaller, where **MqInstallDir** is the directory where MQSeries is installed. When prompted, select the **Uninstall MQSeries completely** option.

e. If the uninstall succeeds, enter `rmdir /S /Q <MqInstallDir>` to remove any remaining files, where **MqInstallDir** is the directory where the MQSeries server was installed.

Solaris

a. Delete the following file: */etc/rc3.d/S50tksmq*

b. Enter

```
/usr/sbin/pkgrm mqm-upd02
```

to remove the MQSeries PTF.

c. Enter

```
/usr/sbin/pkgrm mqm
```

to remove MQSeries.

d. Enter

```
/usr/bin/rm -rf /var/mqm
```

to remove the mqm data directory.

e. Enter

```
/usr/bin/rm -rf /opt/mqm
```

to remove the mqm directory.

AIX

a. Enter

```
/usr/sbin/rmitab "tskmq"
```

to remove the MQSeries inittab entry.

b. Enter

```
/usr/bin/chgrpmem -m - root mqm
```

to remove the root user from the mqm group.

c. Enter the following commands to remove the MQSeries packages:

```
/usr/sbin/installp -uw mqm.base.runtime
/usr/sbin/installp -uw mqm.base.sdk
/usr/sbin/installp -uw mqm.base.samples
/usr/sbin/installp -uw mqm.java.share
/usr/sbin/installp -uw mqm.java.bindings
/usr/sbin/installp -uw mqm.server.rte
/usr/sbin/installp -uw mqm.msg.en_US
```

d. Enter

```
/usr/bin/rm -rf /var/mqm
```

to remove the mqm data directory.

e. Enter

```
/usr/bin/rm -rf /usr/mqm
```

to remove the mqm directory.

3. On UNIX systems, MQSeries semaphores and shared memory segments remain behind when MQSeries is uninstalled. These must be cleaned up manually before trying to re-install

MQSeries. You can do this either by rebooting the machine, or by using the **ipcs** and **ipcrm** commands to list and remove all the remaining MQSeries IPCs.

8

Configuring Tivoli Kernel Services

This section describes special configuration considerations for Tivoli Kernel Services.

For information about configuring the properties of specific Tivoli Kernel Services components see "Configuring the Properties of Components" on page 121.

Note: The wcmd examples in this section assume that the person issuing the commands is implicitly identified to Tivoli Kernel Services by the appropriate account and user information. If this is not the case, you must use the **-u** option on each wcmd and specify a Tivoli Kernel Services user ID that is authorized to perform the command. Additional information can be found in the *Tivoli Kernel Services Command Reference* document, in the section entitled *Security Implications for wcmd Commands*.

Using Multiple MQSeries Servers

The Messaging Service component of Tivoli Kernel Services uses MQSeries servers to route messages between ORBs. Each MQSeries server can support about a dozen ORBs. To support a larger number of ORBs, MQSeries servers can be interconnected into a multi-tier hierarchy. The following example illustrates how this can be done in Tivoli Kernel Services.

A Hypothetical Installation

A hypothetical corporation wants to install three MQSeries servers to support their Tivoli Kernel Services installation. The proposed environment is illustrated in Figure 11 on page 111. The solid lines indicate message flows. The dotted lines indicate control flows.

The installation depot, through the use of the **MSManager** component, is responsible for configuring and monitoring the three MQSeries servers. The three MQSeries servers are *not* also bootprint servers. MQSeries servers and bootprint servers should run on separate machines in a production environment for optimal performance.

The next section describes how to create this environment using Tivoli Kernel Services.

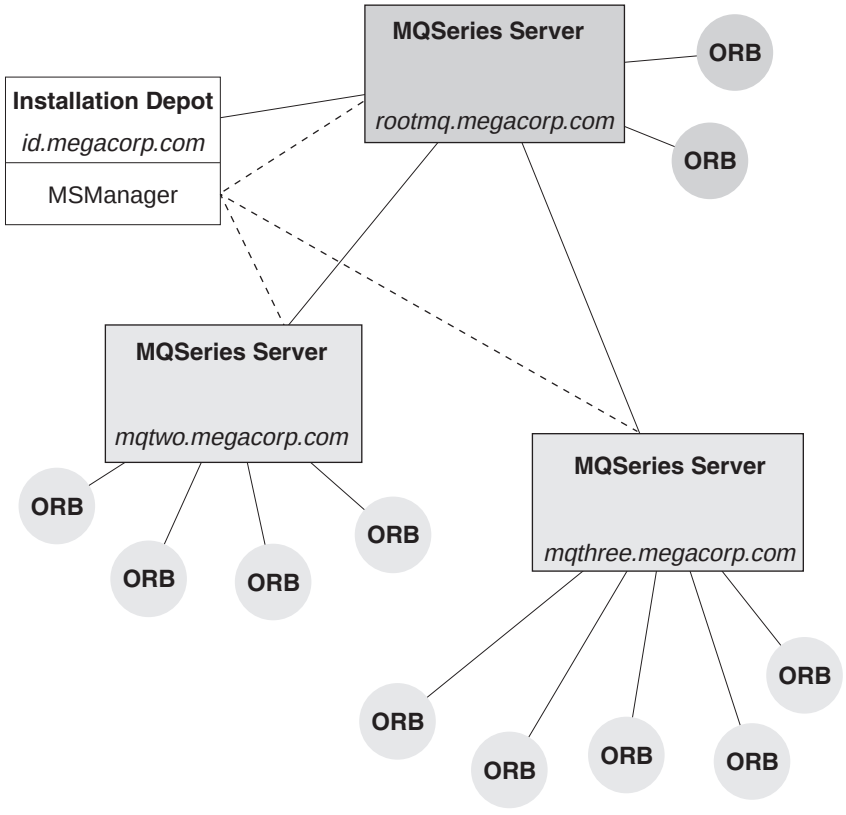

Figure 11. An installation with 3 MQSeries servers

Configuring the Messaging Service

To create the environment illustrated in Figure 11, do the following:

1. Install an MQSeries server on each of the desired machines:
   - **rootmq.megacorp.com**
   - **mqtwo.megacorp.com**
   - **mqthree.megacorp.com**

   using the Tivoli Kernel Services installation program as described in "Multiple Machine Install" on page 26.

2. Install the installation depot on **id.megacorp.com** using the multiple machine install option of the installation program as described in "Multiple Machine Install" on page 26.

   Entering rootmq.megacorp.com as the installation depot's MQ Host will cause the MQSeries server on rootmq.megacorp.com to be configured as the root MQSeries server.

All the commands below are directed at the installation depot.

3. Configure the other MQSeries servers as children of the root MQSeries server by issuing these two commands:

```
wcmd msm addserver mqtwo.megacorp.com:1414 rootmq.megacorp.com:1414
wcmd msm addserver mqthree.megacorp.com:1414 rootmq.megacorp.com:1414
```

4. To verify the proper configuration, issue the following command:

```
wcmd msm status
```

   A typical response would be:

```
Remote manager running id.megacorp.com_9990
rootmq.megacorp.com:1414 connected monitored by      id.megacorp.com_9990
    mqtwo.megacorp.com:1414 connected monitored by   id.megacorp.com_9990
    mqthree.megacorp.com:1414 connected monitored by   id.megacorp.com_9990
```

   The indentation of the last two lines indicates that those MQSeries servers are children of the first.

Configuring MQSeries Servers on Bootprint Servers

When installing new bootprint servers, specify the appropriate MQSeries server and port number based on your installation. In our hypothetical installation in Figure 11 on page 111, **mqtwo.megacorp.com** has 4 bootprint servers assigned to it, **mqthree.megacorp.com** has 5, and the root MQSeries server has the installation depot, 2 bootprint servers, and 2 child MQSeries servers using it.

To change the MQSeries server associated with a bootprint server after it has been installed:

1. Stop the desired bootprint ORB.

2. Issue the following command on the installation depot to change the MQSeries server for the bootprint server to **mqtwo.megacorp.com**:

   ```
   wcmd put /com/tivoli/tes/servers=mqtwo.megacorp.com:1414
   ```

3. Re-launch the ORB.

Removing MQSeries Servers

You can remove the MQSeries server on **mqtwo.megacorp.com** from use with the following command:

```
wcmd msm removeserver mqtwo.megacorp.com:1414
```

You must remove all the children of the root MQSeries server before you can remove it from the Tivoli Kernel Services installation.

Using Multiple Distributed Service Managers

To provide better fault tolerance and make the overall installation more robust, you might want to deploy more than one distributed service manager, or DSM. By default, a distributed service manager is started on the installation depot ORB but cross-server communication using the Messaging Service is not enabled.

Setting Up Another DSM

To start another distributed service manager:

1. Deploy a distributed service manager to a Tivoli Kernel Services server running a bootprint. This can be done through the Tivoli Console, or by issuing the following command directed to the installation depot ORB and specifying the appropriate bootprint:

   ```
   wcmd cds deploy dsm@5.1.1 <bootprint OID>
   ```

2. Be patient. It takes a few minutes for the deployment to complete. To verify that the deployment was successful:

   ```
   wcmd -i <bootprint OID> svc ls LSM
   ```

3. Once another distributed service manager has been deployed, the DSM on the installation depot needs to be notified so that it will use the Messaging Service for communications. This can be done by directing the following command, all on one line, to the installation depot ORB:

```
wcmd cfg put /com/tivoli/com/serviceManager singleDSM=false
```

4. The DSM just deployed on the bootprint also needs to be configured to use the Messaging Service. You can either issue the same command directly on the bootprint ORB, as was done in step 3 on page 113, or direct it there using its OID and one of these two equivalent commands:

   ```
   wcmd -i <bootprint OID> cfg put /com/tivoli/com/serviceManager singleDSM=false
   wcmd cfg put <bootprint OID>/com/tivoli/com/serviceManager singleDSM=false
   ```

5. Once the DSMs are using the Messaging Service, you can connect one or more local service managers, or LSMs, to the new DSM by setting the **DSMPrimary** information on the bootprint entering one of the following two equivalent commands, each command on one line:

   ```
   wcmd -i <bootprint OID> cfg put
   /com/tivoli/com/serviceManager DSMPrimary=//<bootprint IP address>:<port number>
   wcmd cfg put <bootprint OID>
   /com/tivoli/com/serviceManager DSMPrimary=//<bootprint IP address>:<port number>
   ```

6. To verify that the LSM on the bootprint is connected to the DSM on the bootprint:

   ```
   wcmd -i <bootprint OID> svc ls DSM
   ```

Returning to a Single DSM Environment

To return to a single DSM environment:

1. Reconnect the LSM on the bootprint back to the DSM on the installation depot using one of the following two equivalent commands:

   ```
   wcmd -i <bootprint OID> cfg put /com/tivoli/com/serviceManager
   DSMPrimary=//<ID IP address>:<port number>
   wcmd cfg put <bootprint OID>/com/tivoli/com/serviceManager
   DSMPrimary=//<ID IP address>:<port number>
   ```

2. Optionally, you can terminate the use of the Messaging Service, which will release unneeded system resources, using one of the following two equivalent commands:

```
wcmd -i <bootprint OID> cfg put
/com/tivoli/com/serviceManager
singleDSM=true
wcmd cfg put <bootprintOID>
/com/tivoli/com/serviceManager
singleDSM=true
```

A

Hints, Tips, and Troubleshooting

This appendix contains hints, tips and trouble-shooting information.

Pool Size

Pool size configures the size of the connection pool for the directory services directory. It is set during the installation of the Tivoli Kernel Services installation depot.

The default pool size value is 10. The minimum value allowed is 2 and the maximum recommended value is 65. In general, the pool size value should approximate the number of ORBs and should be slightly larger. The optimal value can be difficult to estimate because it depends on the number of simultaneous directory requests occurring on the ORBs. In some cases, even one ORB alone may be sporadically busy enough to overload the directory such that the 10 default connections are not enough to handle the load. In addition, be sure you properly configure the directory services database, by default named **slash**, to handle the size of the connection pool you set.

Frequent occurrences of the **InsufficientResourcesException** error from the directory is a symptom that the directory provider is overloaded and cannot service all of the simultaneous calls being placed to the directory. The immediate response should be to check the size of the connection pool configured for the directory using the following command:

```
wcmd cfg get /com/tivoli/core/directory/slash
DBPOOL.CONNECTION.POOL.SIZE
```

If the pool size is less than 65 connections (65 is a recommended but not an absolute limit) then the pool size should be increased. At the same time as the pool size is increased, the database parameters that control the number of simultaneous applications that can connect to the directory database should be re-examined to see if they are appropriately set.

If the **MAX_NUMBER_OF_ACTIVE_APPLICATIONS** for the directory database is smaller than the **DBPOOL.CONNECTION.POOL.SIZE** it must be increased as well.

If the **DBPOOL.CONNECTION.POOL.SIZE** is already set to 65 and InsufficientResourceExceptions messages are frequent, then additional or replicate copies of the directory should be configured to handle the workload in that Tivoli Kernel Services environment.

Database Connectivity Considerations on AIX

You might experience database connectivity problems when running your DB2 database server and the Tivoli Data Access Services (DAS) server on an AIX machine. This is caused by the use of shared memory by DB2 when communicating with the Tivoli DAS server on AIX and an AIX IPC problem.

If you encounter connectivity problems in this environment, you can configure the server to behave as a remote client, which causes DB2 to use TCP/IP sockets instead of shared memory for communication.

To configure the DB2 database server to use TCP/IP sockets for communication with the Tivoli DAS server, perform the following steps.

1. Stop all ORBs running in the installation.
2. Stop the Tivoli DAS server.

3. Edit the /etc/services file to indicate the connection and interrupt ports that will be used for your DB2 instance. If your DB2 instance is called db2inst1, you might add the following:

```
db2cdb2inst1     50001   # connection port for db2inst1 instance

db2idb2inst1     50002   # interrupt port for db2inst1 instance
```

4. Ensure that your service is registered with the database manager by issuing the following command, all on one line, and specifying the same name you used for the connection port in /etc/services:

```
db2 update database manager configuration using
     svcename db2cdb2inst1
```

5. Catalog the local database server as a remote server node on the configured connection port by substituting the name of the database server for *dbmachine* and using the appropriate port number in the following command.

```
db2 catalog tcpip node dbmachine remote dbmachine server 50001
```

6. Catalog the Tivoli Kernel Services databases as remote aliases:

```
db2 catalog database SLASH as SLASHA at node dbmachine
db2 catalog database PS as PSA at node dbmachine
db2 catalog database TMD as TMDA at node dbmachine
```

   (repeat for any other databases in use, such as SECAUDIT or LOGGING)

7. Reconfigure the Tivoli DAS server by modifying the configuration file to use the new alias names instead of the actual database names. This file is:

```
/usr/local/das/ext/das/server/tksSi/dasDsnCfg.tks
```

   Edit each of the stanzas to change the DB parameter to use the alias. For example, for the slash database, you would change the value of the DB parameter from **slash** to **slasha**:

```
[slash]
UID=db2admin
DBMS=DB2/2
PWD=XV&UXa]b
```

```
DB=slasha
DEFAULT=FALSE
QUAL=db2admin
DRV=DB2CLI
```

8. Stop and start DB2:

```
db2stop
db2start
```

9. Restart the Tivoli DAS server, and then restart the installation depot and the other ORBs.

B

Configuring the Properties of Components

This appendix contains information about configuring the properties of Tivoli Kernel Services components. You can reconfigure the properties of a component by using the Tivoli Console or the **wcmd cfg** command.

Using the Tivoli Console

To configure components using the Tivoli Console, select the **Manage ORBs** task in the **Administer Management Software** task group in the portfolio. From the Manage ORBs window, shown in Figure 12 on page 122, select an ORB to display its components.

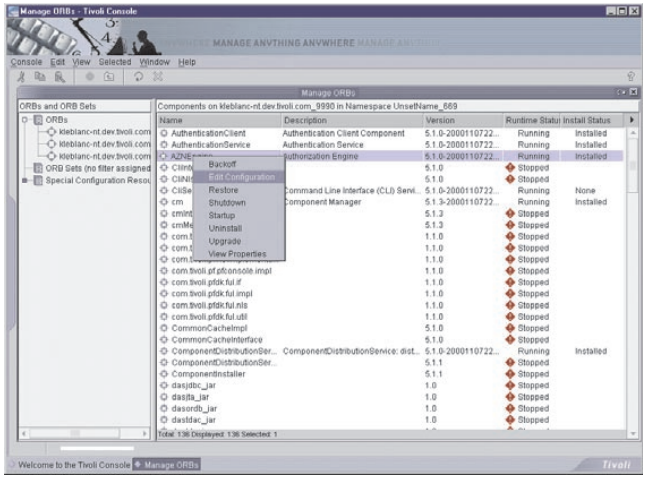

Figure 12. Manage ORBs window

From the Manage ORBs window, you can select any installed component and configure it for the ID, a subset of ORBs, or for a single ORB. When you right-click a component in the Manage ORBs window, the **Edit Configuration** context menu item is displayed. If you select this menu item and the component has configuration data that can be modified, the Edit Configuration window, shown in Figure 13 on page 123, is displayed. If there is no configuration data, the following message is displayed: `Component has no properties to edit`.

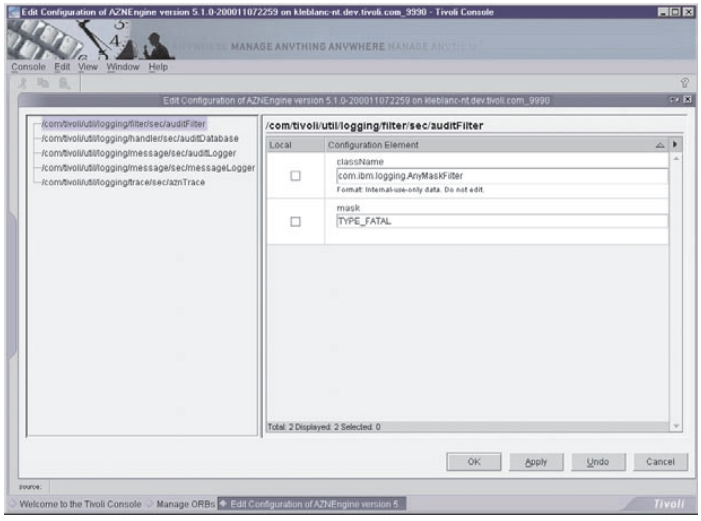

Figure 13. Edit configurable properties

From this window you can configure or change the properties of a component. The component's configuration information is displayed in a table in alphabetical order.

The left pane of the window displays the absolute path to where the configuration properties are stored. The right pane displays the configuration properties stored in the path location. Three columns are displayed in the right pane:

- **Local** - Select to define the property locally on that resource. Properties may be locally defined by default, and the check box will already be selected.
- **Configuration Element** - Displays configuration property name and a text field for editing configuration element content.

CAUTION:
Some component properties displayed in this column are shown for informational purposes only and must not be changed by the user. Although such properties are identified by a `Internal-use-only-data. Do not edit.` warning text, they are NOT protected from inadvertent modification by any type of read-only attribute. Users are therefore strongly urged to exercise extreme caution when editing component properties. ANY CHANGE TO A PROPERTY IDENTIFIED AS NON-EDITABLE CAN RESULT IN A FATALLY CRIPPLED SYSTEM.

- **Value** - An optional column that displays when a component property uses a variable, signified by a $, to display its value. The column displays the exact value used by the ORB for that component property. This field is non-editable.

Select a table row to display the property value source in the **source** field in the lower left-hand corner of the window. This field is updated every time a new row is selected.

Click **OK** to enter any window changes and close the window.

Click **Apply** to enter window changes.

Click **Undo** to clear the last window entry. If it is clicked a second time, the last window entry is restored.

Click **Cancel** to close the window without entering any changes.

Using the wcmd cfg Command

Note: The wcmd examples in this section assume that the person issuing the commands is implicitly identified to Tivoli Kernel Services by the appropriate account and user information. If this is not the case, you must use the **-u** option on each wcmd and specify a Tivoli Kernel Services user ID that is authorized to perform the command. Additional information can be found

in the *Tivoli Kernel Services Command Reference* document, in the section entitled *Security Implications for wcmd Commands*.

This command sets the specified keys and values and flushes the node.

Syntax

wcmd cfg put [*resource*] *pathname key=value* [*key=value...*]

Description

The **wcmd cfg put** command sets the specified keys and values and flushes the preference node (flushing assures configuration settings changed in RAM are also changed in persistent configuration storage). If the resource, node, and key specified did not already exist, the cfg put command creates them.

Options

key=value

Specify one or more key and value pairs. Multiple keys and values can be specified by separating each pair with a space character. For example, /com/tivoli/widgets key1=x key2=y key3=z.

pathname

Specify the full name of a preference node.

resource

Identifies the ORB or ORB set in which a key will be set. If *resource* is omitted, the local ORB is assumed.

Authorization

Write access to the system/services/config subtree is necessary to perform this operation.

Examples

- By default, trace logging is turned to `off`. The default trace logger is named managerTrace. To enable the default logger in an ORB, enter the following command:

$wcmd cfg put /com/tivoli/util/logging/trace/log/managerTrace isLogging=true

- The following shows the syntax to add multiple keys:

```
$wcmd cfg put /full/name/of/path key1=true key2=false
```

See Also

wcmd cfg get, wcmd cfg nodeExists, wcmd cfg remove, wcmd cfg removeNode, wcmd cfg removeResource in the *Tivoli Kernel Services Command Reference*.

Component List

cm@5.1.3

Editable Property	Description	Required?	Default Value	How to Specify
cacheComp	Controls whether local copies of JAR files are cached for components	No	False	True or false
cacheCompDirectory	Controls where local copies of JAR files are cached for components	Yes, if cacheComp is specified; otherwise, no	$ORB BASE DIR/ cache	The pathname of the JAR file cache
downLoadRetry	The number of times to retry when attempting to download a jar file for a component before giving up and allowing the componentManager to attempt to the use the file directly from the webserver	No	3	Integer

ComponentInstaller@5.1.1

Editable Property	Description	Required?	Default Value	How to Specify
processQueueDelayInSeconds	The number of seconds that a component using the queue will wait before processing of the queue begins again	No	60	Integer specified in seconds
maximumManuallyMaintained	The maximum number of components the user would like to have to maintain manually. An automatic process of removing unnecessary jar files will go into effect when the number is exceeded	No	25	Integer
componentRemovalEnabled	Enable or disable the removal of jar files. When enabled, unneeded jar files will automatically be removed.	No	True	Boolean
isLogging	The logger used by this component	No	False	Boolean

dirservice@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
DIRSERVICE. SLASH. PROVIDER	Specifies the location of one or more slash services. This contains a semi-colon-separated list of the ORB URLs where the slash contexts are running.	Yes	None. Installation dependent.	See below.
An example URL is: **//localhost:9990** where 9990 is the port the ORB listens on. The ORB specified should be one where the slash server is running. The semicolon-separated list configured for each dirservice should be ordered differently if the same URLs are used. It is also possible to provide completely different URLs to each dirservice so that each refers clients to different slash contexts. The dirservice will always provide to clients the first slash URL in the list that works. If the first URL in the list does not work, then the dirservice will move to the second one and so on until either a working URL is found, or an exception indicating no working slash is available is returned.				
EVENTS.MAX. QUEUE.SIZE	Determines the number of directory event messages that will be buffered internally by directory code.	No	1000	The default queue size should be adequate for most installations. In installations where high burst rates of directory activity occur, or where slow MQSeries servers are used, the value may need to be increased to prevent messages from overflowing the queue. Overflowing the queue is not a serious error if it only happens occasionally.

Editable Property	Description	Required?	Default Value	How to Specify
DBPOOL. CONNECTION. POOL.SIZE	The number of connections available for a slash service to use when accessing the database. The size of the connection pool determines, to a point, the number of simultaneous clients that can be serviced by a slash service.	No	10	The number of connections in the directory connection pool affects the concurrency of the directory. The default value of 10 is a minimal size and only supports 10 to 15 ORBs. The pool size should be increased as the number of ORBs in the installation increases. Once the pool size reaches 100 or more connections, the system administrator should consider adding a replicate slash context to the installation. The database installation must also be properly configured to allow the specified number of connections. A connection pool that is too small will result in retry exceptions being returned frequently to directory clients.

Editable Property	Description	Required?	Default Value	How to Specify
MAXDELETION TRANSACTION	The largest number of directory entries that can be removed in one transaction. This parameter can be tuned by a system administrator as long as the corresponding database transaction logs are large enough to handle the transaction size.	No	If not defined externally, the maximum transaction size is defaulted to 300 items.	It is recommended that this value be left at 300. The value may be increased if sufficient database transaction log space is made available to the directory database. Increasing the maximum number of records that can be deleted in one transaction without sufficient transaction log space will result in error messages from the database product.
SLASH.CONTEXT. CACHE.SIZE	This value controls the size of the slash services internal cache.	No	1000	The default value of 1000 should be sufficient. If the slash service is running on a machine with extra memory, this value may be increased to possible improve performance.
SLASH.ENABLE. CACHING	Determines if internal caches used by the directory will be enabled.	No	True	This value should be changed with caution and should typically be left set to the default value. The alternate value is “false”.

Editable Property	Description	Required?	Default Value	How to Specify
SLASH.MASTER. OR.SLAVE	Specifies whether the slash service running on the particular ORB is connected to the single master copy of the directory database or to a slave/replicate copy of the directory database. There can only be one master slash service in an installation.	Yes	None. A value must be specified - either *master* or *slave*.	For the master copy of the directory database, the value should be set to "master". For slave/replicate copies of the directory database the value should be set to "slave".
SLASH.USE.VAULT	Internal Use Only.	No	True	Not applicable.
SMART ENUMERATOR. NUMBER BLOCKS.BUFFER	Determines the number of groups of data returned to a directory client in one message. In activities that result in large return sets from the directory, data is returned in small groups to prevent overloading client machines memory with large sets of data.	No	4	As this value is increased, the efficiency of the directory improves with respect to returning large result sets to clients. The default value should be acceptable. This value is also used to determine the size of the server cache when clients request large amounts of data. The number of objects in the cache is SMART ENUMERATOR. NUMBER. BLOCKS.BUFFER * SMART ENUMERATOR. NUMBER. ELEMENTS. BLOCK (default is 600).

Editable Property	Description	Required?	Default Value	How to Specify
SMART ENUMERATOR. TIMEOUT. SECONDS	The number of seconds a client may hold a list connection to the directory without using it before the connection is automatically closed.	No	45	This value, specified in seconds, should be changed only with caution. In environments where very slow network links are used, or where slow computers are used, the value may need to be increased to prevent timeout messages on directory clients. This value applies to all directory clients. The actual value required is installation dependent but the default should suffice in nearly all installations.
SMART ENUMERATOR NUMBER. ELEMENTS.BLOCK	Determines the number of individual items to be placed in a group of items that will be returned to a directory client.	No	150	Increasing this value improves the efficiency of the directory with respect to returning large result sets to clients. The default value should be acceptable.

dsm@5.1.1

By default, the Service Manager is configured to run in a single DSM configuration where the DSM is located on the ID machine and the LSMs running on the ID, as well as on the bootprint machines, are connected to the DSM running on the ID machine.

Editable Property	Description	Required?	Default Value	How to Specify
singleDSM	Enables/disables the use of the Messaging Service. The Messaging Service is only used by the Service Manager component to exchange messages between DSMs.	Yes	True (single DSM mode that does not require any exchange of information with other DSMs) When set to false: (wcmd -i <oid> cfg put /com/tivoli/core/service Manager singleDSM=false) the DSM running on the machine specified by the -i <oid> initializes the Message Service layers in order to be able to send and receive messages from other DSMs.	Boolean

DepotComponentInstaller@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
maxNumberRetriesFor Action	Specifies the number of times the local install will retry before giving up	No	10	Integer
retryDelayInSeconds	The amount of time to wait after a local install fails before trying again	No	30	Integer specified in seconds
repositoryDir	The directory that is hosted by the web server. The repository directory is where the DepotComponentInstaller will place all components when it desires to make them available to other ORBs.	No	$ORB BASEDIR /cd	String pathname
pendingDir	The temporary directory used by the DepotComponentInstaller to download components from other sources before making them available in the repository directory	No	$ORBBAS EDIR/cmp	String pathname
maxNumberRetriesFor Action	Specifies the number of times the local uninstall will retry before giving up	No	5	Integer
retryDelayInSeconds	The amount of time to wait after a local uninstall fails before trying again	No	60	Integer specified in seconds

Editable Property	Description	Required?	Default Value	How to Specify
useWebServer	Indicates that a web server will be used to distribute components to other ORBs	No	True	Boolean
webServerDirectoryAlias Name	Used to define the web server directory name that will be used to serve components to other ORBs	No	Repository	String that can be interpreted as a valid directory alias by the web server
pollingIntervalInMinutes	The amount of time the DepotComponentInstller will wait before checking for any new deploy or retract actions	No	10	Integer specified in minutes
parentDepots	A semicolon-separated list of depots that this depot uses in order to find new components	No	3.1234 5678901 23456.1. 123456 7890123 456	A semicolon-separated list of depots
URL	The URL hosted by the web server which is a prerequisite of the DepotComponent Installer. Other ORBs get components from the Depot using this URL.	No	http:// <host name> /Repo sitory	String that can be interpreted as a valid URL

GatewayIPService@5.1.1

Editable Property	Description	Required?	Default Value	How to Specify
numberOf Threads	The number of threads that the gateway will use in its thread pool	No	64	Integer
numberOf Queues	The number of queus that the gateway will use in its thread pool	No	32	Integer
pingTTL	Time to live for a ping	No	64	Integer
maximumHop Count	Maximum hops before command quits	No	128	Integer
pingTimeout	timeout interval for ping responses in milliseconds	No	15	Integer
pingDataSize	Size of ping packet	No	32	Integer

GatewaySNMPService@5.1.1

The following properties are used by the SNMP protocol stack to clean up unused sessions. The protocol stack includes a 'sweeper' that is invoked at regular intervals and looks for sessions that have not been used to send messages for a certain amount of time. All sessions that have been idle for longer than the allowed time are terminated. Attempts to use the session after it is closed will result in an error being passed back to the user. The units for both of these properties are specified in minutes. This fine granularity is allowed primarily for testing. Unless you are sure that you have no applications that poll on long intervals, the values should remain similar to the defaults (24 and 12 hours respectively).

Editable Property	Description	Required?	Default Value	How to Specify
allowedIdleTime	The amount of time, in minutes, that a session can remain idle before it is terminated. It may actually live longer than this, since the lastUsed value is only checked at each sweep through the sessions, but a session can never be left idle longer than cleanUpInterval + allowed IdleTime.	No	720	Integer
cleanUpInterval	The amount of time, in minutes, between sweeps through the open sessions. This should be kept long to minimize the overhead caused by the algorithm, and to minimize the killing of sessions that are actually in use.	No	1440	Integer
numberOfThreads	The number of threads that the gateway will use in its thread pool	No	64	Integer
numberOfQueues	The number of queues that the gateway will use in its thread pool	No	32	Integer

lsm@5.1.1

By default, the Service Manager is configured to run in a single DSM configuration where the DSM is located on the ID machine and the LSMs running on the ID, as well as on the bootprint machines, are connected to the DSM running on the ID machine. However, the administrator can configure the Service Manager components (LSMs and DSM(s)) to achieve fault-tolerance or simply backup purposes through three different parameters: singleDSM, DSMPrimary and DSMBackups.

Editable Property	Description	Required?	Default Value	How to Specify
DSMPrimary	Specifies the IP address and port number of a machine running a DSM that the local LSM has to be connected to.	Yes	IP address and port number of the ID machine	String ″//aaa.bbb.ccc. ddd\:int where ″aaa.bbb.ccc. ddd″ is a fully qualified hostname and ″int″ the port number
DSMBackups	In a fault-tolerant environment, this property allows the administrator to specify a list of alternate DSMs that the local LSM can connect to when the connection to the primary DSM has failed.	Yes	None	Strings, ″//aaa.bbb.ccc. ddd\:int where ″aaa.bbb.ccc. ddd″ is a fully qualified hostname and ″int″ the port number. When more than one DSM is specified, the DSM references have to be semicolon-separated.

LocalComponentInstaller@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
maxNumberRetriesFor Action	Specifies the number of times the local deploy will retry before giving up	No	12	Integer
retryDelayInSeconds	The amount of time to wait after a local deploy fails before trying again	No	30	Integer specified in seconds
basicTestEnabled	Specifies if the local deploy should attempt to run the basicTest method on every component that is deployed	No	True	Boolean
maxNumberRetriesFor Action	Specifies the number of times the local retract will retry before giving up	No	8	Integer
retryDelayInSeconds	The amount of time to wait after a local retract fails before trying again	No	60	Integer specified in seconds
pollingIntervalInMinutes	The amount of time the LocalComponent Installer will wait before checking for any new deploy or retract actions	No	1400	Integer specified in seconds
componentDepots	A semicolon-separated list of depots that the current ORB looks to for its components.	No	3.1234 5678901 23456.1. 123456 7890123 456	A semicolon-separated list of depots

MACImpl@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
visibleDefault or an NLS label. See note below.	Default value to be used by the generic configuration GUI to be used as the default setting of the visible attribute. This default value will only be used for those nodes and properties that don't have the "visible" flag set.	No	The default setting of false should be used unless the user needs extensive debug information.	Boolean
Note: If an NLS label appears as the property name, the raw property name can be accessed by using the View -> Show Configuration Details option.				

NelService@5.1.1

NelService has two types of parameters; static keys and keys that are derived from orboids.

The table below lists and describes the static properties that reside in the /com/tivoli/core/mmgatew/gateway/nels preference node.

Editable Property	Description	Required?	Default Value	How to Specify
DefaultGateway SNMP	The default gateway that the NelService will use if no other SNMP configured gateway is defined to support the requested endpoint. If a default gateway is supplied, then requests for endpoints that cannot be reached by the configured gateways will not be issued.	No	An empty string	Value is the oid identifing the ORB where a SNMP gateway will be/is running. Example: DefaultGateway SNMP= ″3.1343423. 2345235235″
DefaultGateway IP	The default gateway that the NelService will use if no other IP configured gateway is defined to support the requested endpoint. If a default gateway is supplied then requests for endpoints that cannot be reached by the configured gateways will not be issued.	No	An empty string	Value is the oid identifing the ORB where a IP gateway will be/is running. Example: DefaultGatewayIP= ″3.1343423. 2345235235″

The table below lists and describes the non-static IP properties that reside in the /com/tivoli/core/mmgatew/gateway/nels/GatewayIP preference node.

Editable Property	Description	Required?	Default Value	How to Specify
<orboid>	The ORB oid where an IP gateway is going to run. The key is variable and there can be mulitple definitions. However, multiple scope ranges for the same gateway must be on the same line.	Yes. If you have an application requiring IP gateway services then you need to either configure the gateway or a default.	None	orboid= ″<subnet address>:<scope mask>:<nat id>;<repeat> Multiple scopes for the same gateway are specified by using a semicolon to separate them. Each scope is defined with a subnet address which is an address in that subnet, a subnet mask, and a nat id.

An example of this command:

wcmd cfg put /com/tivoli/core/mmgatew/gateway/nels/GatewayIP 3.1.4.5.2252345325.235k=″146.84.29.109:255.255.255.0:0;69.2.2.2:255.255.0.0:0″

which defines the gatewayipservice running at the big orboid to have 2 scopes (there can 1 or n scopes). this statement means that endpoints addresses like 146.84.29.x at a public nat (not a private network - 0 means public) and addresses 69.2.x.x (nat 0 again) can be sent to this gateway.

The format of the command is:

<orboid>=″<subnet address>:<scope mask>:nat id;<repeat for however many address ranges this gateway handles>″

The table below lists and describes the non-static SNMP properties that reside in the /com/tivoli/core/mmgatew/gateway/nels/GatewaySNMP preference node.

Editable Property	Description	Required?	Default Value	How to Specify
<orboid>	The Orb oid where an IP gateway is going to run. The key is variable and there can be mulitple definitions. However, multiple scope ranges for the same gateway must be on the same line.	Yes. If you have an application requiring IP gateway services then you need to either configure the gateway or a default.	None	orboid="<subnet address>:<scope mask>:<nat id>;<repeat> Multiple scopes for the same gateway are specified by using a semicolon to separate them. Each scope is defined with a subnet address which is an address in that subnet, a subnet mask, and a nat id.

An example of this command:

wcmd cfg put /com/tivoli/core/mmgatew/gateway/nels/GatewaySNMP 3.1.4.5.2252345325.235k="146.84.29.109:255.255.255.0:0;69.2.2.2:255.255.0.0:0"

which defines the gatewayipservice running at the big orboid to have 2 scopes (there can 1 or n scopes). this statement means that endpoints addresses like 146.84.29.x at a public nat (not a private network - 0 means public) and addresses 69.2.x.x (nat 0 again) can be sent to this gateway.

The format of the command is:

<orboid>="<subnet address>:<scope mask>:nat id;<repeat for however many address ranges this gateway handles>"

PsCommonInt@1.1.0

Editable Property	Description	Required?	Default Value	How to Specify
compInitPauseTime or an NLS label. See note below	Length of time in seconds to wait before rechecking if NS or MCR initialization complete	Yes	10	A string with numeric characters only
compInitRetryCount or an NLS label. See note below	Number of times to check if the MCR or NS has completed initialization	Yes	100	A string with numeric characters only
orbInitPauseTime or an NLS label. See note below	Length of time in seconds to wait before rechecking if the NS ORB or MCR ORB start up is complete	Yes	10	A string with numeric characters only
orbInitRetryCount or an NLS label. See note below	Number of times to check if the MCR ORB or NS ORB has completed start up	Yes	100	A string with numeric characters only
Note: If an NLS label appears as the property name, the raw property name can be accessed by using the View -> Show Configuration Details option.				

PsJcImpl@1.1.0

Editable Property	Description	Required?	Default Value	How to Specify
locale or an NLS label. See note below.	Indicates the language used for the PS Java client logon screen	No	en_US	A string representing a Java Locale. Either a two letter language (for example ″en″) or a language and locale (for example ″en_US″) can be supplied.
Note: If an NLS label appears as the property name, the raw property name can be accessed by using the View -> Show Configuration Details option.				

SecurityCacheService@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
CACHE_TYPE	This parameter is set automatically and should not be changed. This parameter determines whether the Authorization is acting like a ″ROOT″ or ″CHILD″. The Authorization cache that resides on the same ORB as the SecurityDirectory Service should be set as ″ROOT″ and all other caches be set as ″CHILD″.	Yes	None	String
CAPACITY	The maximum number of entries allowed in the Authorization cache. If the maximum number of entries is exceeded, the oldest are discarded. If the value is set to zero (0), no limits will be placed on the size of the cache.	Yes	100,000	Integer
PRIMARY_ PARENT	Specifies the ORB ID of the desired parent Authorization cache. This should be automatically set at install time. By default, this will be set as the Install Depot for all non-Install Depot ORBs.	Yes	None	ORB ID
BACKUP_ PARENT	Specifies the ORB ID of one or more backup Authorization caches. Multiple backup parent ORB ID's should be separated by a semicolon.	No	None	A semicolon-separated list of ORB IDs.

ServletMgrImpl@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
/com/tivoli/si/ users/servlets	Contains the versioned component names with a value of true (eg. abc@1.1.1=true). This is a list of installed components that ServletMgrImpl should handle as serlvet components containing a war file for distribution to the servlet engine. These key=value pairs come from the xml of the servlet components.	Yes	The only value that should be used is the one specified	Should not be changed
/com/tivoli/si/ users/servlets/ Installed	Contains a list of the versioned components from the above list that have been installed on the servlet engine. The key is a versioned component name and the value is its war file name (eg.abc@1.1.1=abcservlet. war). These key=value pairs are maintained at runtime by ServletMgrImpl as servlet components are deployed or retracted to or from its ORB.	Yes	The only value that should be used is the one specified	Should not be changed

SimpleJarComponentInstaller@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
maxNumberRetriesFor Action	Specifies the number of times the local deploy will retry before giving up	No	12	Integer
retryDelayInSeconds	The amount of time to wait after a local deploy fails before trying again	No	30	Integer specified in seconds
basicTestEnabled	Specifies if the local deploy should attempt to run the basicTest method on every component that is deployed	No	True	Boolean
maxNumberRetriesFor Action	Specifies the number of times the local retract will retry before giving up	No	8	Integer
retryDelayInSeconds	The amount of time to wait after a local retract fails before trying again	No	60	Integer specified in seconds
pollingIntervalInMinutes	The amount of time the LocalComponent Installer will wait before checking for any new deploy or retract actions	No	1400	Integer specified in seconds
componentDepots	A semicolon-separated list of depots that the current ORB looks to for its components.	No	3.1234 5678901 23456.1. 123456 7890123 456	A semicolon-separated list of depots

timesync@5.1.0

Editable Property	Description	Required?	Default Value	How to Specify
TIME_SERVER	The NTP server to be used by timesync components running on each ORB to determine the correct time	No[1]	None	String
TIME_SERVER_ PORT	The port number to be used when contacting an NTP server.	No[2]	123	Integer
POLL_INTERVAL	Number of seconds after which the timesync component will contact the appropriate NTP server to determine clock offset and adjust the local clock. 21600 seconds equals 6 hours.	Yes	21600	Long[3]
SET_CLOCK	Specifies whether the timesync component is supposed to adjust the local clock's value. If this value is set to "false", the timesync component will still contact the NTP server and determine the clock offset... but will not modify the local clock.	Yes	True	Boolean
SERVER_LIST	A space-delimited list of host names or IP addresses that represent NTP servers. The timescan component will choose the best server from this list for use by the timesync component to set local clocks.	Yes[4]	List of all known stratum 2 NTP servers	String

Editable Property	Description	Required?	Default Value	How to Specify
SHORT_LIST	A space-delimited list of NTP servers that are used to ″quickly″ set the TIME_SERVER variable while statistical analysis is done of the total SERVER_LIST. This variable contains one well-known NTP server per continent.	No	List of one stratum 2 NTP server per continent	String
SCAN_CYCLE_ INTERVAL	Number of seconds after which the timescan component will recheck connectivity to all NTP servers and select the best one based on statistical sampling. 2628000 seconds corresponds roughly to one week.	Yes	2628000	Long[3]
BASELINE_ ITERATIONS	Number of samples taken by the timescan component when setting a baseline to determine which NTP servers are not within operational limits.	No[5]	50	Integer
SCAN_ ITERATIONS	Number of samples taken by the timescan component to determine which NTP server is ″best″ (has the shortest, non-fluctuating network delay to the machine on which timescan is running).	No[5]	50	Integer

Editable Property	Description	Required?	Default Value	How to Specify
INITIAL_WAIT_ TIME	Number of seconds the timesync and timescan components will wait after being started before they begin their processing (allows ORBs to initialize before time synchronization components run). 300 seconds is 5 minutes.	Yes	300	Long[3]
INACTIVE_ WAIT_ SECONDS	Number of seconds the time synchronization components wait if no NTP servers can be located.	Yes	21600	Long[3]

[1] In most normal operations, this value should not be set. However, if the timescan component is not used (for some reason), the user may set the NTP server directly using this parameter. If the timescan component is running, it will set this parameter for all ORBs.

[2] This value is defined by the NTP RFC, and should not be modified. However, it is listed here in case there is some unforeseen scenario (firewall, etc.) where the value may need to be changed.

[3] Values for this parameter must be positive.

[4] A user would want to change this value in the case where the correct NTP servers are located on the customer's network, within firewalls.

[5] 50 is the minimum number of samples from which a valid statistical analysis can be made. The user can increase the number (and even decrease it), but 50 is a good minimum.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.