License Use Management

IBM

Using License Use Management Runtime for HP-UX**, IRIX**, and Solaris**

Version 4.5.5 (September 7, 1999)

SH19-4361-02

License Use Management

IBM

Using License Use Management Runtime for HP-UX**, IRIX**, and Solaris**

Version 4.5.5 (September 7, 1999)

SH19-4361-02

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xiii.

ISO 9001 Certification

This product was developed using an ISO 9001 certified quality system.

Certification has been awarded by the Italian quality system certification group, CSQ (Certification No. CISQ/CSQ 9150.IBM7).

CSQ is a member of the mutually recognized organization of European assessors, ITQS, which assesses and certifies quality systems in the field of information technology enterprises.

Third Edition (September 1999)

This major revision obsoletes and replaces SH19-4361-01. The major changes are described in “Summary of Changes” on page xix. In the hard copy version of this book, technical changes are marked by a vertical line in the left margin. In the .HTM version, technical changes appear in purple. Post-publication technical changes are marked in brown.

This edition applies to Version 4.5.5 of IBM License Use Management Runtime for HP-UX, IRIX, and Solaris, a part of License Use Management, Program Numbers 5697-D34 (HP-UX), 5697-D36 (IRIX), and 5697-D35 (Solaris), and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

License Use Management Information Development
Rome Tivoli Lab
IBM Italia S.p.A.
Via Sciangai, 53
00144 Rome
Italy
Fax Number : (+39) 06 5966 2077
Internet ID: ROMERCF at VNET.IBM.COM

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
P.O. Box 12195
3039 Cornwallis
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
IBM
OS/2

LicensePower and iFOR are registered trademarks of Isogon Corp.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

HP-UX is a registered trademark of Hewlett-Packard Company.

IRIX is a trademark of Silicon Graphics, Inc.

Solaris is a registered trademark of Sun Microsystems.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks, and Authenticode a trademark, of Microsoft Corporation in the U.S. and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

About This Book

This book provides a guide to setting up the environment required to support licensed software products. Refer to Chapter 1, “Introduction to License Use Runtime” on page 1 for details on licensed products. This book describes License Use Management Runtime for HP-UX, IRIX, and Solaris (referred to as License Use Runtime). It contains information about how to install and configure the servers where licenses will be installed and the clients that will use the products, and how to set up a network licensing environment. It explains how to manage the licenses for licensed software products you have installed.

Important!

On HP-UX, IRIX, and Solaris, License Use Management Version 4 is not a follow-on of any previous version from any vendor. In particular, no migration is done from any other licensing products, and it is not possible to reuse existing configurations.

If any other license management product is installed on the machines in your environment, it is essential that you keep the two licensing environments separate and independent. Pay particular attention to the parts of this book that explain the considerations involved. Be sure you read Chapter 2, “Planning Your Network Licensing Environment” on page 35 carefully before you begin configuring servers and clients. Considerations for coexistence with other license management products are in the sections “Selecting Your Servers” on page 35, “Installing on a Machine where NCS is Already Installed” on page 37, “Planning Direct Binding” on page 39, and “Planning Cells” on page 40.

Before reading this guide, follow the procedures described in the documentation accompanying the licensed software product you acquired.

Who Should Use This Book

This book is intended for:

The system administrator who is responsible for setting up and administering the license management environment using License Use Runtime.

The License Use Runtime system administrator must have general knowledge of:

- HP-UX, IRIX, and Solaris
- The TCP/IP transport protocol
- The network environment

The end users who run the software products on their client machines and, if required, install and configure License Use Runtime with the support of the system administrator.

About This Book

End users require only general knowledge of the HP-UX, IRIX, and Solaris operating environment.

How This Book is Organized

This book is divided into the following sections:

Chapter 1, “Introduction to License Use Runtime,” provides an overview of License Use Runtime features and benefits, describes supported license types, license policies, and license-enabling models, and presents some simple scenarios of the use of licensed products.

Chapter 2, “Planning Your Network Licensing Environment,” provides the basic NCS concepts you need to set up your network and manage licenses with License Use Runtime, and gives you information to assist you in organizing your network.

Chapter 3, “Installing License Use Runtime,” gives you instructions on how to install and uninstall License Use Runtime.

Chapter 4, “Getting Started with License Use Runtime,” provides information on configuring License Use Runtime, administering product licenses, and exercising customer-controlled policies, using the License Use Runtime tools.

Chapter 5, “License Use Runtime Commands,” documents the License Use Runtime command line interface.

Chapter 6, “Hints and Tips,” provides hints and tips to better take advantage of License Use Runtime, and information on how to use the provided backup and recovery procedure.

Chapter 7, “Troubleshooting,” helps you to improve performance and to handle problems, should they arise when you use license-enabled products.

Appendix A, “License Use Runtime Configuration File,” provides reference information on the configuration file.

| Appendix B, “Using the Nodelock File,” explains how to edit a nodelock file.

| Appendix C, “Features and Functions Added in Version 4” lists the features and
| functions that have been added in Version 4.

“Glossary” defines terms used in this manual.

Where to Find More Information

This section lists other sources of information related to License Use Runtime.

Command Help

To get help with the syntax of a particular command, go to the appropriate directory:

```
/opt/lum/ls/os/hpux/doc
/opt/lum/ls/os/svr4.sgi/doc
/opt/lum/ls/os/solaris/doc
```

Enter this command:

Enter the following command:

```
man command_name
```

This displays the command syntax. For example, to get help with the command `i4blt`, type:

```
man i4blt
```

Note that to make it possible to display the man pages, you may need to update the MANPATH environmental variable in the .profile to specify the appropriate directory first:

```
/opt/lum/ls/os/hpux/doc
/opt/lum/ls/os/svr4.sgi/doc
/opt/lum/ls/os/solaris/doc
```

Online Books

The online books for License Use Runtime, listed in the rest of this section, are
Tcy:mm

.,messn pagelp wianFordisnasectiosy lim a sectionndare

About This Book

Using License Use Runtime

This book is available as an online book, as well as in hard copy. The file is named:

`lumusg`

A printable copy of this book in PostScript and PDF formats is available for download from the IBM LUM Web site http://www.software.ibm.com/is/lum.

Using Application Developer's Toolkit

For information about how to license-enable software products for use with License Use Management, see *Using License Use Management Application Developer's Toolkit*, SH19-4362. If Application Developer's Toolkit is installed in your environment, it is available as an online book in .HTM format. The file is named:

`lumtkt`

License Use Runtime README File

For changes to License Use Runtime or to this book that were made after the book went to press, see the README.ARK file on the CD-ROM or in the download package from the LUM Web site.

IBM LUM Web Site

Visit the IBM License Use Management Web site at http://www.software.ibm.com/is/lum for information and news about IBM License Use Management, and to download License Use Runtime publications and code.

Notational Conventions

This book uses the following notation in text:

Bold	Bold print indicates names of statements and parameters.
Italics	Italic print is used for variables for which you must supply a value, for introducing new terms in the text, and for emphasis.
`Monospacing`	Monospacing indicates system messages and examples.


This icon marks important information that can affect the operation of the product or the completion of a task.

Summary of Changes

This section provides an overview of major changes made to this book.

Changes in the Third Edition

The third edition of this book incorporates changes documented in the README.ARK and README.JCS files for Version 4.5.1 and Version 4.5.2, and changes made to Version 4.5.5 of the product.

General Changes

This section summarizes the general, pervasive changes made to this book.

Viewing Books Online

Books are no longer supplied in INF format. View the HTML-format versions of books in your web browser.

Performance of the Network License Server Process

Performance of the network license server process on the UNIX platforms, which was improved in Version 4.5.1, is faster than in releases prior to Version 4.5.1.

Additional Operating Systems

Support has been added for the following operating systems:

- Windows NT 4.0 Server, Terminal Server Edition (Windows Terminal Server), on the x86 platform. The behavior of License Use Runtime on Windows Terminal Server is the same as on Windows NT (x86).
- Windows NT 4.0 on the Alpha platform. On this platform, License Use Runtime functions are available only from the command line.
- Windows NT 4.0 Server, Terminal Server Edition, on the Alpha platform. On this platform, License Use Runtime functions are available only from the command line.

For information, see “License Use Runtime Platforms” on page 4 and Chapter 3, “Installing License Use Runtime” on page 55.

New Versions and Releases of Operating Systems

Support for previously supported operating systems has been extended to include the following new versions and releases:

- HP-UX 11.0 32-bit
- SGI IRIX 6.5
- Sun Solaris 2.7

For information, see “License Use Runtime Platforms” on page 4 and Chapter 3, “Installing License Use Runtime” on page 55.

Summary of Changes

Specific Changes

This section summarizes the changes made to this book to reflect new and changed function and support.

Custom Configuration

Vendors can now offer combinations of products, tailored to the needs of each user, under a single custom configuration license.

A section has been added to Chapter 3, Installing License Use Runtime that introduces the concept of custom configuration.

A section has been added to Chapter 6, Hints and Tips that suggests how you might better manage a custom configuration.

A section has been added to Chapter 7, Troubleshooting that suggests what you might do should you have a problem with a custom configuration.

Tuning and Performance Enhancements

New environment variables are provided to assist in tuning to optimize performance of the license server. See “Tuning the Environment to Manage the Workload” on page 147.

Trace Enhancements

Tracing of network and TCP/IP activity has been enhanced, and a time stamp has been added to each trace record.

License Creation Tool Enhancement

The License Creation Tool enables you to create licenses whose start date is one day earlier than the date the tool is run. This makes it possible for licenses to be used immediately in any time zone. See “i4lct - License Creation Tool” on page 130 for details.

Editing a Nodelock File

A new appendix, Appendix B, “Using the Nodelock File” on page 181, explains how to edit and use a nodelock file.

List of Features and Functions Added in Version 4

A new appendix, Appendix C, “Features and Functions Added in Version 4” on page 183, lists the features and functions added in Version 4. Do not use obsolete commands or APIs, which are supported for backward compatibility, with these newer features and functions.

Chapter 1. Introduction to License Use Runtime

License Use Runtime is part of IBM License Use Management, a combination of tools for software asset protection. The License Use Management tools enable software vendors and their customers to ensure that customers comply with the terms and conditions of license agreements. They check compliance through runtime monitoring of the usage of software assets.

License Use Management at a Glance

License Use Management consists of two products:

The License Use Management Application Developer's Toolkit contains the tools that are needed to implement licensing technology in an application program (called *license-enabling* the application). To do the enablement, vendors code API calls in their products and embed the code that services the API calls. The products thus become *license-enabled*. Vendors can license-enable C-language programs, Java applications, and Java applets.

The Application Developer's Toolkit offers the vendor great flexibility in:

- Level of control exercised by the enabled application
- Type of customer licensing environment for which the application is intended
- Implementation of various policies

The Application Developer's Toolkit is a priced product of IBM. The software vendor who acquires the kit receives a copy of the License Use Management software, and gets royalty-free rights to redistribute License Use Runtime within the license-enabled application.

License Use Management Runtime (License Use Runtime) contains the tools that are needed in an end user environment to manage licenses and to get up-to-date information about license usage. The License Use Runtime software is free of charge and is available for download from the IBM License Use Management (LUM) Web site http://www.software.ibm.com/is/lum.

License Use Management Vendor Perspective

License Use Management benefits software vendors by enabling them to:

Ensure that customers use software licenses within entitled limits

Base product prices on actual usage

Protect intellectual property from unauthorized use

Increase overall revenue as customers acquire all the licenses they need

Distribute software for a trial period with trial licenses that can be replaced by production licenses, thus minimizing distribution cost

License Use Management at a Glance

License Use Management Customer Perspective

License Use Management benefits the customers of software vendors by enabling them to:

- Ensure that they have enough licenses to satisfy their business requirements and, at the same time, that they are not paying for more licenses than they need
- Base software charges within the enterprise on actual usage
- Demonstrate license use compliance to internal and external auditors
- Protect organizations from inadvertent violations of license agreements
- Change software assets to alternative pricing policies that the vendor offers

Basic Concepts of License Use Management

A *license*, in the context of License Use Management, is permission to use an instance of a licensed software product or service, according to the basis on which the vendor charges for the product or service. The objective of License Use Runtime is to control the use of licenses in a customer's environment.

(Note that the term *license* does not refer to the license agreement that governs use of and rights to a product.)

In the license-enabling process, the vendor can:

- Select among various types of licenses (see “License Types” on page 7).
- Decide whether to distribute licenses one-by-one or in packages of multiple licenses from which individual licenses can be extracted (see “License Passwords” on page 5).
- Implement direct controls over the use of licenses, or make it possible for the customer to control use of licenses (see “Password Use Control Levels” on page 6).
- Impose, or allow the user to impose, various types of control over administration of licenses (see “License Policies” on page 11).

Vendors deliver licenses to customers in the form of a *license password*. The password contains an encryption of some terms of the acquisition of the software product. For example, the password may specify:

- How many licenses or concurrent copies of the product the customer can use
- The expiration date of the licenses
- The type of license

License Use Runtime checks that customers have a license that authorizes them to use the product when the product is executed, not when it is installed.

Depending on the terms for software product acquisition, vendors can implement licenses in two fundamental ways: *nodelocked* licenses and *network* licenses.

License Use Management at a Glance

A nodelocked license is stored on the workstation where the license-enabled product is installed, for the exclusive use of that node.

With network licenses, you set up a client/server configuration for License Use Management. Many License Use Runtime clients can share the licenses for enabled products. The licenses are stored on one or more *network license servers*. Each client workstation must be connected to a server. When the user at a client starts a licensed program, License Use Runtime at the license server determines whether a license is available.

License-enabled Java applications and applets must have network licenses. A Web server machine, rather than the end user machine where the application or applet runs, serves as the network license client. See “Planning for Java Applications and Applets” on page 43.

License Use Runtime includes an administration tool, called the *Basic License Tool*, which manages both nodelocked and network licenses on all the license servers in your network. The Basic License Tool enables you to:

- Add licenses to or delete licenses from the server database
- Display information about the licenses installed
- Distribute the licenses among the license servers available on the network
- Reserve licenses for the exclusive use of certain users
- Generate reports on license usage and server events

The Basic License Tool has a command-line interface. For more information about what the Basic License Tool does, see “License Administration Tool” on page 17.

License Use Runtime Platforms

License Use Runtime Platforms

Table 1 shows which platforms License Use Runtime supports, and how to get the License Use Runtime code:

Table 1. License Use Runtime Platforms

AIX 4.3.3	License Use Runtime 4.5.5 base code is part of the base operating system, and is installed on every machine when the operating system is installed. Optional packages and filesets can be installed from the AIX installation media. Alternatively, you can install Version 4.5.5 from the License Use Management Version 4.5.5 CD-ROM or from the product package downloaded from the IBM LUM Web site http://www.software.ibm.com/is/lum.
AIX 4.3.2 **AIX 4.3.1** **AIX 4.3.0**	On AIX 4.3.2, License Use Runtime 4.5.0 base code is part of the base operating system, and is installed on every machine when the operating system is installed. On AIX 4.3.1, License Use Runtime 4.0.1 base code is part of the base operating system, and is installed on every machine when the operating system is installed. On AIX 4.3.0, License Use Runtime 4.0 base code is part of the base operating system, and is installed on every machine when the operating system is installed. Optional packages and filesets can be installed from the AIX installation media. To upgrade to License Use Runtime Version 4.5.5 without upgrading to AIX 4.3.3, download the code from the IBM LUM Web site http://www.software.ibm.com/is/lum.
AIX 4.1 **AIX 4.2**	The iFOR/LS license management product is part of the base operating system in AIX 4.1 and 4.2, and is installed on every machine when the operating system is installed. To upgrade to License Use Runtime Version 4.5.5, download the code from the IBM LUM Web site http://www.software.ibm.com/is/lum.
Windows NT 4.0 (x86) **Windows NT 4.0 Alpha** **Windows NT Server 4.0, Terminal Server Edition (x86)** **Windows NT Server 4.0, Terminal Server Edition Alpha** **Windows 98** **Windows 95** **OS/2 Warp Version 4** **Sun Solaris 2.6 and 2.7** **HP-UX 10.20 and 11.0** **Silicon Graphics IRIX 6.3, 6.4, and 6.5**	License Use Runtime Version 4.5.5 can either be redistributed with the license-enabled product or be downloaded from the IBM LUM Web site http://www.software.ibm.com/is/lum.

On AIX and OS/2, License Use Management Version 4.0 replaced previous License Use Management releases. On Windows, HP-UX, IRIX, and Solaris, Version 4.0 was the first IBM License Use Management release.

License Passwords

A license password (or *license key*) is an encrypted character string that specifies the characteristics of the license. This information, determined by the vendor, includes:

The specific number and type of license contained in the password
The date when the licenses become active
The date when the licenses expire

Vendors can create two types of password: *simple* and *compound.*

A simple password, once enrolled on a license server, represents one or more licenses that the license server can grant when an end user starts the product.

A compound password, once enrolled on a license server, is a single password from which you can extract multiple simple passwords. Each extracted simple password represents one or more licenses. The compound password is a means of:

Efficiently distributing multiple licenses from the vendor to the customer.

Distributing licenses to different license servers, when required. The compound password must be installed on a specific license server. Extracted passwords can be distributed as required to other license servers that are not specified in the compound password.

Providing a sales representative with a set of licenses that the representative can distribute to different customers.

A compound password contains an expiration date that the vendor sets. The duration of extracted licenses cannot be longer than the time remaining before the compound password expires.

The vendor includes the password, along with other information about the application, in the *enrollment certificate file* (ECF).

Figure 1 on page 6 summarizes the relationship among the license, the license password, the compound password, and the enrollment certificate file.

Password Use Control Levels

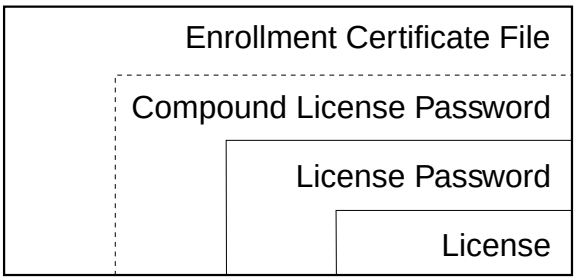

Figure 1. Licensing Concepts Summary

Password Use Control Levels

Vendors can enable their products according to either of the following predefined use control levels:

Vendor-Managed Use Control (VMU)
Customer-Managed Use Control (CMU)

Vendor-Managed Use Products

With *vendor-managed use* products, the vendor manages compliance with the terms of the acquisition of the software product.

When you acquire licenses for a vendor-managed use product, the product vendor will ask you to supply the unique identifier (target ID) of each machine where you intend to install the product licenses. For nodelocked licenses, this is the identification of the workstation where the enabled product is to be installed; for network licenses, this is the network license server. The password, tied to the specified workstation, cannot be used on another workstation. You must also supply the number of licenses you want to acquire. The vendor uses this information to create the password that you use to install and activate the licenses.

Vendors typically ship a vendor-managed use product with a simple password. They can also use compound passwords for this purpose. You can extract and distribute a limited number of licenses from the compound password, up to the maximum the vendor set in the compound password.

If you want to change the terms and conditions of the contract (for example, to increase the number of licenses), you provide the vendor with similar information for each of the machines on which you intend to install the licenses and get a new password.

A vendor can produce a vendor-managed use license password with target ID set to ANY, so that you can install it on any machine. The vendor can deliver such a password with the product package without your specifying how many licenses you want to acquire. Instead, the vendor sets an upper limit, possibly very large, on the number of licenses you can install on each license server. By generating and

delivering such a password, the vendor has decided not to perform the checks that are normally associated with vendor-managed use.

Customer-Managed Use Products

To provide vendors with greater flexibility in the way they deliver licensed software, License Use Runtime supports *customer-managed use* control. With products that are enabled in this way, the vendor does not directly associate licenses with a particular license server (or group of license servers). The vendor does not set an upper limit on the number of licenses that you are entitled to use. Instead, it is your responsibility to set that upper limit, depending on the terms of the software product acquisition.

License Use Runtime provides you with the information on the usage of the enabled products, thereby helping you stay within the boundaries of the acquisition agreement. Transactions, such as enrollment, distribution of licenses, updates, and deletions, are logged in a tamper-proof License Use Runtime database.

Vendors typically ship a customer-managed use product with a compound password that you can use to extract and distribute the number of licenses you have acquired. They can also use simple passwords for certain types of licenses.

License Types

This section describes the types of license the vendor can select. When you receive a license-enabled product, check the product documentation to determine the license type.

Nodelocked Licenses

A nodelocked license allows the use of a product on the particular machine for which the license was created for as long as the license remains valid. Vendors typically use nodelocked licenses for standalone, rather than client/server, applications.

A vendor who is enabling a product that uses nodelocked licenses can choose between two license-enabling models: *non-runtime-based* and *runtime-based.*

If the vendor chooses non-runtime-based enabling, the license-enabled product itself, rather than License Use Runtime, manages use of the nodelocked license. The password for such a product is stored in a file that is called the *nodelock file.* When you start the application, it checks the nodelock file to ensure you have a valid license.

If the vendor chooses runtime-based enabling, management of the nodelocked license is performed by the *nodelocked license server* on the local machine. You interact with the nodelocked license server through the Basic License Tool. It enables you to view and update information about the nodelocked licenses on the machine and get reports about their use.

See “License-Enabling Models” on page 14 for more information about license-enabling models.

License Types

Vendors can enable their products to use the following kinds of nodelocked licenses:

- Simple nodelocked licenses
- Concurrent nodelocked licenses
- Use-once nodelocked licenses
- Per-server licenses

Simple Nodelocked Licenses

A *simple nodelocked* license allows an unlimited number of simultaneous uses of the licensed application on the local machine. Simple nodelocked licenses are valid only for vendor-managed use products. A word processor is a typical example of a product that uses nodelocked licenses.

Concurrent Nodelocked Licenses

As with a simple nodelocked license, the *concurrent nodelocked* license is local to the node where the application has been installed. It allows a limited number of simultaneous uses of the licensed application. A typical example of a concurrent nodelocked license is a client/server application. The application server is able to recognize the number of clients connected to it and ask for a license for each of them.

Vendors can use concurrent nodelocked licenses for both vendor-managed and customer-managed products.

When you enroll a customer-managed product, you must specify how many concurrent nodelocked licenses you have acquired for the product. The administrator can modify this number at any time.

Use-Once Nodelocked Licenses

A *use-once nodelocked* license permits a single use of a licensed product on a particular machine during the period the license is valid. Every time the product is started, one license is consumed.

A typical use of use-once nodelocked licenses is to distribute promotional or demonstration versions of software.

Vendors also provide use-once nodelocked licenses to supplement concurrent nodelocked licenses during times when user demand for those products exceeds the number of available concurrent nodelocked licenses. The vendor designs the product so that when all concurrent nodelocked licenses for the product are in use, a user can request an available use-once license.

Vendors can use use-once nodelocked licenses for both vendor-managed and customer-managed products.

When you enroll the licenses for a customer-managed product, you must specify how many use-once nodelocked licenses you have acquired for the product. The administrator can modify this number at any time.

Per-Server Licenses

Per-server licenses are exactly like concurrent nodelocked licenses, except that at any time, you can change them into per-seat licenses (see “Per-Seat Licenses” on page 11 and “Switching from Per-Server to Per-Seat Licenses” on page 14).

Vendors use per-server/per-seat licenses to enable client/server applications constructed for multiple-server solutions. With both per-server and per-seat licenses, the server of a licensed client/server application can request licenses for its clients. The application clients need not be license-enabled.

With per-server licensing, each application server license is associated with a specific number of application clients. This represents the maximum number of application clients that may concurrently request services from that application server. The application client licenses are stored locally on the application server machine and are granted temporarily to requesting application clients. Multiple application servers grant licenses independently of one another; if the same application client connects to more than one application server, the application client is granted more than one license. You should therefore probably use per-server licenses only in an environment where:

Each application client connects to only a single application server, or
Each application client uses the application infrequently for brief periods.

When your environment grows in such a way that application clients are connecting to multiple application servers, you will probably convert your per-server licenses to per-seat. With *per-seat* licensing, unused application client licenses are kept in a central repository, which all the application servers share. They also share a central list of application clients to which a license has been assigned. When a license is assigned to an application client, the license remains assigned to the application client even when it is not using the product. If an application client connects to multiple application servers, it is assigned only one license.

Per-server licenses are valid only for customer-managed use products.

Network Licenses

Network licenses, rather than being restricted to a single machine, are stored on a network license server and shared among multiple network license clients.

Vendors can enable their products to use the following kinds of network licenses:

Concurrent licenses
Reservable licenses
Use-once licenses
Per-seat licenses

Concurrent Licenses

A *concurrent* license is a network license that can be temporarily granted to run the licensed application on a client.

License Types

When the product is running, that license remains unavailable to other users of the product. When the product stops running, the license is returned to the server, where it becomes available to other users.

Concurrent licenses allow as many users to run a licensed application simultaneously as there are valid licenses for the product available from the network license servers in your licensing environment.

A typical use of concurrent licenses is for products with relatively expensive licenses that each user will use only some of the time. The customer orders fewer licenses than there are users to optimize use of the licenses. Such applications may be either client/server applications, for which the client is enabled, or non-client/server applications.

Vendors can use concurrent licenses for both vendor-managed and customer-managed products.

Reservable Licenses

A *reservable* license is a network license that you can reserve for the exclusive use of a user, a group, or a node. The reservation is for a specified time period. A reservable license that has been reserved is called a *reserved* license. A reservable license that has not been reserved is called an *unreserved* license.

When a reserved license is granted from the network, the license is stored on the workstation where the licensed application is running. Thereafter, the license can be used on the workstation, even if the workstation is disconnected from the network, until the reservation expires.

A typical use of reservable licenses is for the client part of a client/server application that is likely to run on a portable computer that is often disconnected from the network. Another typical use is for a compiler being used in software development. During a build process involving many compilations, it is more efficient to reserve a compiler license for a day or two than to make a separate request for a compiler license for every compilation.

You can reserve some of the reservable licenses for an application and leave others unreserved. Unreserved licenses are treated like concurrent licenses.

Vendors can use reservable licenses for both vendor-managed and customer-managed products.

Use-Once Licenses

A *use-once* license is a network license that permits a single use of a licensed product during the time the license is valid. Every time the product is started, one license is consumed.

A typical use of use-once licenses is to distribute promotional or demonstration versions of software.

Vendors also provide use-once licenses to supplement concurrent licenses when user demand for those products exceeds the number of available concurrent licenses. The vendor designs the product so that when all concurrent licenses for the product are in use, a user can request an available use-once license.

Vendors can use use-once licenses for both vendor-managed and customer-managed products.

Per-Seat Licenses

Vendors use per-server/per-seat licenses to enable client/server applications constructed for multiple-server solutions. With both per-server and per-seat licenses, the server of a licensed client/server application can request licenses for its clients. The application clients need not be license-enabled.

With per-seat licensing, unused application client licenses are kept in a central repository, which all the application servers share. They also share a central list of application clients to which a license has been assigned. When a license is assigned to an application client, that assignment is permanent. If an application client connects to multiple application servers, it is assigned only one license.

You will probably want to use per-seat, rather than per-server, licenses in an environment where application clients connect to multiple application servers. (See also “Per-Server Licenses” on page 9.)

Per-seat licenses are valid only for customer-managed use products.

License Policies

Vendors can enable their products to implement various policy decisions regarding how licenses are managed.

Vendor-Controlled Policies

The vendor can implement the *try-and-buy, multiuse rules, product wait queues*, and *license annotation* license policies.

Try-and-Buy Policy

The vendor can enable a product with a special simple nodelocked license for customers to use during an evaluation period. The evaluation period (with duration set by the vendor) starts either when the product is enrolled or when the product is run for the first time.

Multiuse Rules

Multiuse rules define the conditions under which multiple invocations of a product require only a single license. These rules are applicable only to concurrent, concurrent nodelocked, and per-server licenses.

The vendor can enable a product so that after a license has been granted to a particular user, group, or node, a second invocation of the product does not require a

second license. For example, if a user invokes a compiler repeatedly, a multiuse rule might specify that the second and subsequent invocations do not require additional licenses.

Multiuse rules may be based on any combination of the following tests that the server performs when a concurrent license is requested:

- The request for a license is associated with the same user as a previous request.
- The request for a license is associated with the same group as a previous request. The vendor can also change the meaning of the “same group” rule to implement a vendor rule. For example, the vendor might implement a multiuse rule that applies when a request is associated with the same display as a previous request. Vendors can also modify the meaning of “same group” in other ways, to implement whatever multiuse rules they design. Any vendor-specific rule overrides the “same group” rule.
- The request for a license is associated with the same node as a previous request (applicable to concurrent licenses only).
- The request for a license is associated with the same job ID as a previous request.

For information about how to implement multiuse rules, see “Defining Rules for Multiple-Use Concurrent Licenses” on page 140.

Product Wait Queues

Some license-enabled products with concurrent licenses may use wait queues.

When a user invokes such a product, and there are no concurrent licenses currently available, the product can be enabled to ask if the user wants to wait for a license. If the user responds affirmatively, the user is added to the wait queue on each License Use Runtime network license server that provides concurrent licenses for the product. User names are added to the wait queues in chronological sequence. When a license becomes available, it is granted to the first user in the queues. The next user in the queues becomes eligible for the next available license.

License Annotation

License annotation is data that is defined and included as part of the license information when a license is created. When the license is granted, the data is passed to the enabled application for its own use. Licenses of any type can be annotated.

A typical use of license annotation is to create licenses that correspond to different configurations of the same product. Consider an application that has several optional priced features, all delivered as part of the product package. The vendor can create license annotations to define which options the customer has bought and, therefore, which features are accessible to the end user.

Custom Configuration

Vendors who want to offer selected combinations of products, tailored more precisely to the needs of users, can define custom configurations by adding functions and products to a base configuration.

You specify the required content of a custom configuration when you order the configuration. You can order a custom configuration for one seat or for a block of any number of identical seats. If you order a configuration for a block of seats, the quantity of each add-on function or product must equal the number of seats in the block.

Each custom configuration, whether for a single seat or for a block of two or more seats, is assigned a separate custom configuration license. A custom configuration license is a special case of either a concurrent network license or a simple nodelocked license that contains a unique serial number identifying that custom configuration. The single serial number and license for a block configuration helps you to manage your installed licenses more easily.

After initial installation of a custom configuration, you can better manage the evolution and growth of your configurations, by ordering additional “add-on” functions and products, as necessary. To retain a single serial number and license, however, any changes made to the custom configuration must be applied to all seats under that serial number.

Customer-Controlled Policies

The customer can exercise the *hard stop/soft stop, user access restriction*, and *per-server/per-seat switch* license policies.

Hard Stop/Soft Stop Policy

The vendor can enable a product so that you can choose the behavior of the product when the end user starts it and no licenses are available.

If no license is available, one of two things can happen:

The product does not start, and there is no way for the end user to go on (*hard stop policy*).

The product starts (*soft stop policy*).

Only applications with customer-managed use licenses can be enabled for the hard stop/soft stop policy. Vendor-managed use licenses are always hard stop licenses.

When you enroll a product enabled for hard stop/soft stop, the default is soft stop. You can use the Basic License Tool to change the policy to hard stop and back again at any time.

License Use Runtime keeps track of the soft stop licenses in use, so that you can get meaningful information about license use.

When the soft stop policy is set, License Use Runtime keeps track of the *high-water mark.* The high-water mark is the maximum number of licenses ever granted for a

given product beyond the number of licenses that are enrolled for that product. You can see this number through the Basic License Tool, and you can reset it to 0. Use this number to help you decide the number of additional licenses you need. When the hard stop policy is selected, the number of in-use licenses cannot exceed the number of enrolled licenses, so the high-water mark is not maintained.

User Access Restriction

You can use the *user file* to control which users have access to licenses for specific products. The user file is a flat ASCII file that you create using a text editor. For each product in the file, there is a list of users. It lists either those who are allowed to use the product (in which case no one else can use it) or those who are not allowed to use it (in which case anyone else can use it).

See “Example 5: Restricting User Access” on page 77 for details.

Switching from Per-Server to Per-Seat Licenses

Vendors of client/server applications who choose per-server/per-seat licensing provide you with two enrollment certificates:

The per-server certificate, containing a per-server password.
The per-seat certificate, containing a per-seat password.

You have the option to start in per-server mode, and switch at any time to per-seat mode, or start directly in per-seat mode. Once the per-seat mode has been activated, it is not possible to go back to per-server mode.

See “Per-Server Licenses” on page 9 and “Per-Seat Licenses” on page 11 for information to help you decide between per-server and per-seat. See “Example 3: Switching from Per-Server to Per-Seat Licenses” on page 76 for information about how to perform the switch.

License-Enabling Models

To summarize, the product vendor can create license-enabled products that use nodelocked or network licenses. The enablement of nodelocked licenses can be either *non-runtime-based* or *runtime-based*.

If the vendor chose non-runtime-based enabling (nodelocked licenses only), the product does not make use of License Use Runtime on the machine where the product runs. Following the vendor's installation instructions, you may be required to store the password for such a product in a vendor-selected nodelock file. When you start the application, it checks the nodelock file to ensure you have a valid license. It is not necessary for the nodelocked license server to be running for the license to be granted. Information about use of the product is not logged. You cannot use the Basic License Tool to view information or get reports about the product and its usage.

If the vendor chose runtime-based enabling for a product with nodelocked licenses, the product makes use of License Use Runtime on the machine where the product runs. It does not require configuration unless the end user has special requirements. The

password for such a product is stored in the nodelocked license database. When you start the application, it contacts the nodelocked license server, which checks its database to ensure you have a valid license. Information about use of the product is logged. You can use the Basic License Tool to view information or get reports about the product and its usage.

A network license-enabled product makes use of License Use Runtime on the machine where the product runs and requires some limited configuration on that machine. The licenses are stored on one or more network license servers. When the user at a client starts a licensed program, License Use Runtime at the license server determines whether or not a license is available.

Table 2 summarizes the license-enabling models, license types, and license policies.

Table 2. License-Enabling Models, License Types, and License Policies

License-Enabling Model	License Types Available	License Policies Available
Nodelocked* License-Enabled Products (Non-Runtime-Based Enabling)	Simple Nodelocked	Try-and-Buy License Annotation Custom Configuration
Nodelocked* License-Enabled Products (Runtime-Based Enabling)	Simple Nodelocked	Try-and-Buy License Annotation
	Use-Once Nodelocked	User Access Restriction License Annotation
	Concurrent Nodelocked, Per-Server	Hard Stop/Soft Stop Multiuse Rules User Access Restriction License Annotation
Network License-Enabled Products	Concurrent	Hard Stop/Soft Stop Multiuse Rules User Access Restriction Product Wait Queues* License Annotation Custom Configuration
	Reservable	Hard Stop/Soft Stop (when unreserved) User Access Restriction License Annotation
	Use-Once	User Access Restriction License Annotation
	Per-Seat	Hard Stop/Soft Stop User Access Restriction Per-Server/Per-Seat Switch* License Annotation

Note: * This applies only to C-language applications.

License Creation Tool

Scalable Installation and Configuration

License Use Runtime consists of separate installable components, so that you can install exactly what you need on each workstation.

For example, in AIX 4.3.3, all required License Use Runtime components are automatically installed on every AIX workstation as part of AIX 4.3.3 installation. You can install optional components, such as the graphical user interface, either when you install AIX or later.

On OS/2 and Windows, you can select the appropriate components depending on the role the workstation is to play in your licensing environment. On OS/2, Windows NT (x86), Windows NT Alpha, Windows Terminal Server (x86), Windows Terminal Server Alpha, Windows 95, and Windows 98, there are components for runtime, communications, and online documentation. On OS/2, there is also a component for namespace binding support. (See “Namespace Binding” on page 38.) Vendors can, optionally, incorporate the communications component into the installation images of their license-enabled products at the minimum level of installation and configuration the products require. Alternatively, they can specify that you should download and install License Use Runtime.

When you configure License Use Runtime, the configuration tool recognizes which components are installed and presents only the options consistent with the installed component.

The configuration tool, for configuring License Use Runtime license servers and clients, has a command-line interface on all platforms, a graphical user interface on AIX, Windows, and OS/2, and an interactive script interface on all UNIX platforms.

License Creation Tool

License Use Runtime includes a tool that creates product licenses for the use of vendors who create license-enabled products. Two uses of the license creation tool are:

The tool enables vendors to create these kinds of passwords:

- Test passwords, for use in testing while enabling a product.
- Production passwords, to deliver to customers.

  To create production passwords, vendors must acquire the license for this tool from IBM or from Isogon Corp. The address of Isogon Corp. is:

  Isogon Corporation
  330 Seventh Avenue
  New York, New York 10001
  U.S.A.
  Tel: (+1) 212-376-3200
  Fax: (+1) 212-376-3280

Distribution of production passwords to customers depends on the use control level of the license-enabled products:

For customer-managed use control products, the customer receives the license password together with the product package.

For vendor-managed use control products, for IBM license-enabled products, the customer requests the license password from the IBM country software password distribution center. For non-IBM license-enabled products, the customer requests the license password from the vendor software password distribution center.

The tool is also useful for vendor sales representatives, who can be provided by the vendor with a production compound password for a vendor-managed use product. The compound password contains many licenses, from which the sales representative extracts licenses for individual customers.

For details about how to use this tool, see “i4lct - License Creation Tool” on page 130.

License Administration Tool

License Use Runtime includes a license administration tool, which is called the *Basic License Tool*.

The Basic License Tool has a command-line interface on all platforms, and a graphical user interface on AIX, OS/2, Windows NT (x86), Windows NT Server Terminal Server Edition (Windows Terminal Server) (x86), Windows 95, and Windows 98. It enables you to:

Manage all types of licenses

The administrator can use the Basic License Tool to manage nodelocked and network licenses.

Add, update, or delete licenses

Add licenses to or delete licenses from the network license server or nodelocked license server database; update information about existing customer-managed use licenses.

Display information

Display a notebook of information about the licenses that are installed for each product.

Distribute licenses

Extract licenses from a compound password and distribute them among the network license servers available on the network.

Reserve licenses

Manage the reservation of reservable licenses for the exclusive use of certain users.

License Administration Tool

Manage multiple network and nodelocked license servers

From any properly configured machine, you can view and manage licenses that are installed on any network license server and on any nodelocked license server in the network. Working at a single administration site, you can manage all kinds of licenses on all machines. The capability to manage licenses on nodelocked license servers is particularly useful for per-server and concurrent nodelocked licenses.

Generate reports

Standard Event Report. Displays detailed information about significant events that occur on the license servers that you specify.

License Request by Product Report. Displays statistical information about the use of the licenses of a product in the time interval that you specify. For each product, it reports the licenses that were requested, the licenses that were granted, and the percentage of rejections.

License Request by User Report. Displays statistical information about the use of products by users in the time interval that you specify. For each user, it reports the licenses that were requested, the licenses that were granted, and the percentage of rejections.

License Use by Product Report. Displays statistical information about the use of the licenses of a product in a specified time interval. For each product, it reports:

- The maximum number of nodes that used licenses for the product at the same time
- The maximum number of users that used licenses for the product at the same time
- The average time the licenses were in use

License Use by User Report. Displays statistical information about the use of the licenses by each user in a specified time interval. For each user, it reports the times the user requested licenses and the length of time the user kept the licenses in use.

Customer-Managed Use Audit. Reports the following information for customer-managed use product transactions:

- Vendor name
- Product name
- Product version
- Administrator information
- Time stamp
- Number of licenses
- Transaction type (for example, product enrolled, license distributed, license deleted, license updated, per-server/per-seat license migrated)

- Signature stamp (user, group, and node)
- Signature information

Use the high-water mark

When the soft stop policy is in effect, the high-water mark is recorded in the licensing database. The high-water mark is the maximum number of licenses ever granted for a given product beyond the number of licenses that are enrolled for that product. You can see this number through the Basic License Tool and can reset it to 0. This number assists you in deciding how many additional licenses you need. When the hard stop policy is selected, the number of in-use licenses cannot exceed the number of enrolled licenses, so the high-water mark is not maintained.

Set the threshold

You can set a threshold percentage of licenses. If more than the threshold percentage of licenses for a product are in use, messages about the level of usage are logged. There is a single threshold that applies to all vendor-managed products and, by default, to customer-managed products. You can change that threshold, and you can also set a separate threshold for each customer-managed product.

Exercise customer-controlled policies

You can switch between hard stop and soft stop, switch from per-server to per-seat, and manage the identifiers of application clients using per-seat licenses.

High-Availability Licensing

High-availability licensing enables you to set up an environment in which there is a very high degree of certainty that concurrent licenses will be available, even if a network license server goes down.

When you use this option, you create a *cluster* of network license servers. A cluster is a group of from 3 to 12 network license servers that jointly serve vendor-managed concurrent licenses that are enrolled on the cluster rather than on an individual server.

You can create and administer a cluster, and administer high-availability licenses, from any machine. However, only AIX, HP-UX, IRIX, Solaris, Windows NT, Windows NT Alpha, and Windows Terminal Server network license servers can be members of a cluster: no OS/2, Windows 95, or Windows 98 network license server can be a member of a cluster.

While some servers in the cluster are serving licenses, one or more servers remain in reserve, ready to take over should an active server fail.

Each active server serves an equal share of the licenses enrolled on the cluster. When a server becomes unavailable and another server takes its place, responsibility for the licenses is automatically redistributed among active servers.

For high-availability licensing to work for a particular product, the product vendor must supply a password tied to a cluster rather than to an individual target server.

Working with Licensed Products

High-availability licensing works only with the IP protocol and does not support the product wait queue policy. Before you decide to use high-availability licensing for a product, make sure such a password is available from the product's vendor.

High-availability licensing is recommended only for users who are already experienced with managing individual license servers and who already have a stable licensing environment working.

See “Planning Clusters” on page 43 for planning information, and “Example 6: Creating and Administering a Cluster” on page 78 for an example.

Backup Procedure

On license servers, there is an automatic backup procedure for License Use Runtime databases and files.

Working with Licensed Products

This section explains what happens when a user starts a licensed product, depending on how the product is enabled.

In the figures and text in this section, references to the *enabled application* or *enabled applet* refer to the application or applet itself (which contains API calls) plus the embedded code that services the API calls (which carries out the described steps).

This section assumes that, if required:

License Use Runtime is installed properly.
License Use Runtime is configured properly.
A Web server machine is set up properly for Java applications and applets.
The network is running properly.

If not, what happens depends on how the vendor enabled the product. See the product documentation for details.

Central Registry License Server

Some of the scenarios in this section show the use of a special server, which is called the *central registry* license server. The central registry is a repository of information that all the other network license servers can use. If you plan to install customer-managed use products with network licenses, or products with reservable licenses, you must identify one (and only one) central registry. Otherwise, the central registry license server is not required.

Some of the uses of the central registry are:

All the per-seat licenses in the licensing environment are installed on the central registry.

The list of application clients to which per-seat licenses have been granted is maintained in the central registry.

Soft stop licenses are tracked in the central registry.

The high-water mark is recorded in the central registry.

Reserved licenses that have not yet been granted to a user are kept in the central registry.

Working with Nodelocked Licenses (Non-Runtime-Based Enabling)

Figure 2 shows what happens when an end user invokes an application with nodelocked licenses for which the vendor chose non-runtime-based enabling. The licenses must be simple nodelocked licenses.

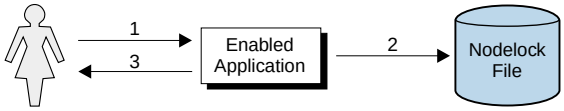

Figure 2. Using a Nodelocked License (Non-Runtime-Based Enabling)

1 The end user invokes the application.

2 The application checks the nodelock file to ensure a license is stored on the local system.

3 If there is a valid license in the nodelock file, the application runs. If not, depending on how the vendor enabled the application, it may return information to the end user, or it may run even with no license available.

For information about how to edit a nodelock file, see Appendix B, “Using the Nodelock File” on page 181.

Working with Licensed Products

Working with Nodelocked Licenses (Runtime-Based Enabling)

Figure 3 shows what happens when an end user invokes an application with nodelocked licenses for which the vendor chose runtime-based enabling. The licenses can be simple nodelocked, use-once nodelocked, or concurrent nodelocked licenses.

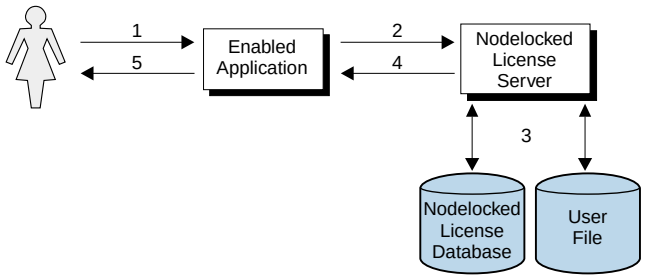

Figure 3. Using a Nodelocked License (Runtime-Based Enabling)

1 The end user invokes the application.

2 The application requests a license from the nodelocked license server on the local system.

3 The nodelocked license server checks that there is a valid license on the machine and that this user is authorized to use it.

If there is no nodelocked license but the application uses concurrent nodelocked licenses and implements the soft stop policy, the nodelocked license server checks for a soft stop license and checks the user file for authorization.

4 The nodelocked license server returns the status of the license request to the application.

5 If a license was found and granted, or if a soft stop license was granted, the application runs. If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Working with Use-Once Licenses

Figure 4 shows what happens when an end user invokes a C-language application with use-once licenses.

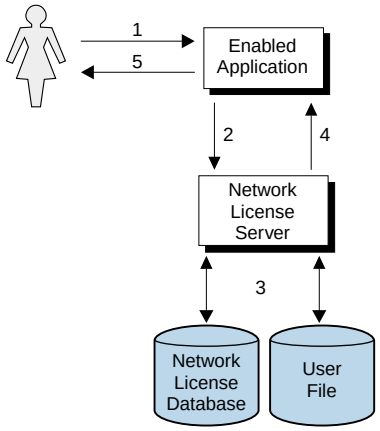

Figure 4. Using a Use-Once License for a C-Language Program

1 The user invokes the application.

2 The application requests a license from the network license server.

3 The network license server checks its license database for an available license and the user file for authorization.

4 The network license server returns the status of the request to the application. If a license was found and granted, the application runs, and one license is subtracted from the number of available use-once licenses.

5 If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Figure 5 on page 24 shows what happens when an end user invokes a Java application or applet with use-once licenses. The primary difference between usage of use-once licenses for C and Java programs is that in the Java case, the Web server machine, rather than the end user's machine, serves as the network license client.

Working with Licensed Products

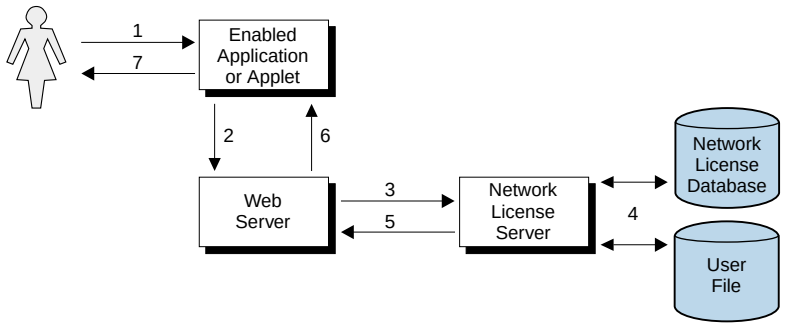

Figure 5. Using a Use-Once License for a Java Application or Applet

1 The user invokes the application or downloads the applet through a Web browser.

2 The application or applet sends a license request to the Web server using the http protocol.

3 The Web server requests a license for the application or applet from the network license server.

4 The network license server checks its license database for an available license and the user file for authorization.

5 The network license server returns the status of the request to the Web server. If a license was found and granted, one license is subtracted from the number of available use-once licenses.

6 The Web server returns the status of the request to the application or applet, using the http protocol. If the status is OK, the application or applet runs.

7 If no license can be granted, depending on how the vendor enabled the application or applet, it may return information to the end user, or it may run even with no license available.

Working with Concurrent Licenses

Figure 6 shows what happens when an end user invokes a C-language application with concurrent licenses.

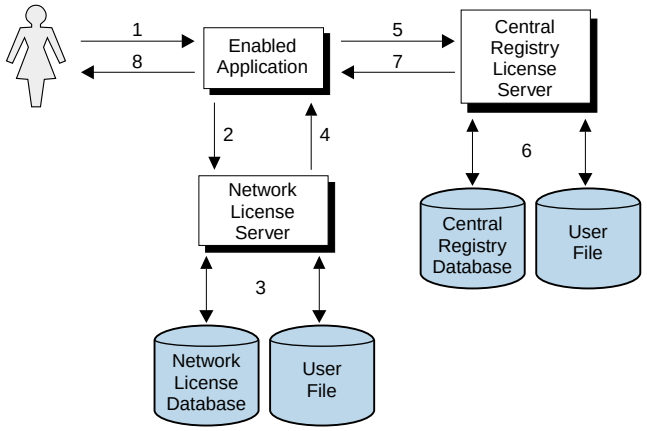

Figure 6. Using a Concurrent License for a C-Language Program

1 The user invokes the application.

2 The application requests a license from the network license server.

3 The network license server checks its license database for an available license and the user file for authorization.

4 The network license server returns the status of the request to the application. If a license was found and granted, the application runs.

5 If a network license was not found, and the application implements the soft stop policy, the application requests a soft stop license from the central registry license server.

6 The central registry license server checks its database for a soft stop license and the user file for authorization.

7 The central registry license server returns the status of the request to the application. If a soft stop license was granted, the application runs.

8 If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Figure 7 on page 26 shows what happens when an end user invokes a Java application or applet that has concurrent licenses. The primary difference between usage of concurrent licenses for C and Java programs is that in the Java case, the Web server machine, rather than the end user's machine, serves as the network license client.

Working with Licensed Products

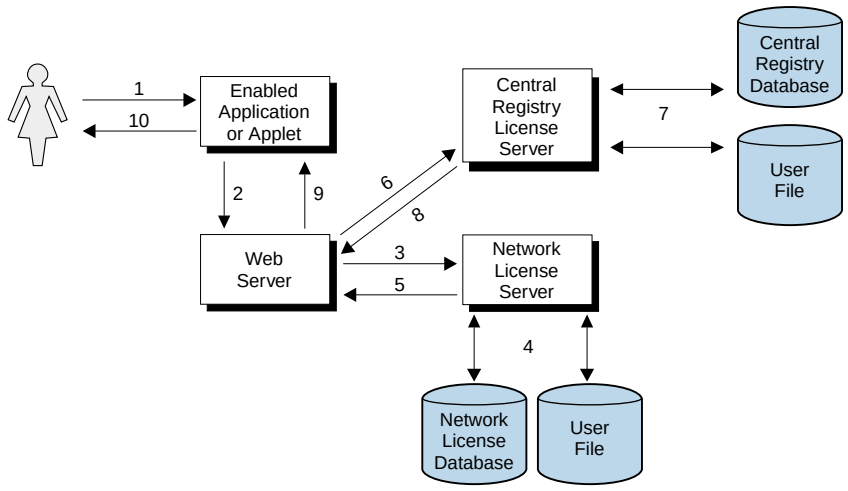

Figure 7. Using a Concurrent License for a Java Application or Applet

1 The user invokes the application or downloads the applet through a Web browser.

2 The application or applet sends a license request to the Web server using the http protocol.

3 The Web server requests a license for the application or applet from the network license server.

4 The network license server checks its license database for an available license and the user file for authorization.

5 The network license server returns the status of the request to the Web server. If a license was found and granted, the Web server returns a positive status to the application or applet, and it runs.

6 If no concurrent license was found, the Web server requests a soft-stop license from the central registry license server.

7 The central registry license server checks its database for a soft-stop license and the user file for authorization.

8 The central registry license server returns the status of the request to the Web server.

9 The Web server returns the status of the request to the application or applet, using the http protocol. If the status is OK, the application or applet runs.

10 If no license can be granted, depending on how the vendor enabled the application or applet, it may return information to the end user, or it may run even with no license available.

Working with Reservable Licenses

Figure 8 shows what happens when an end user invokes a C-language application with reservable licenses.

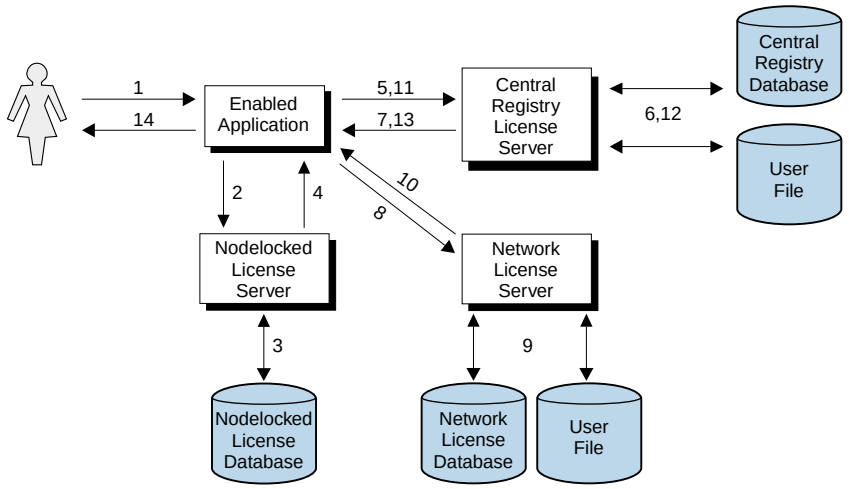

Figure 8. Using a Reservable License for a C-Language Program

1 The user invokes the application.

2 The application requests a reserved license from the nodelocked license server.

3 The nodelocked license server checks its database for a reserved license. This is a license that you reserved for the user. It was granted to the user and stored on the local machine, in response to a previous request.

4 The nodelocked license server returns the status of the request to the application. If a license was found, the application runs.

5 If the nodelocked license server does not find a license, the application requests a reserved license from the central registry license server. This is a license that you have reserved for this user, group, or workstation.

6 The central registry license server checks its database for a reserved license and the user file for authorization.

7 The central registry license server returns the status of the request to the application. If a reserved license was found and granted, it is stored in the nodelocked license server's database, and the application runs.

8 If a reserved license was not found, the application requests a reservable license from the network license server. This is a reservable license that you have not reserved for anyone.

9 The network license server checks its license database for a reservable license and the user file for authorization.

10 The network license server returns the status of the request to the application. If a reservable license was found and granted, the application runs.

11 If a reservable license was not found, and the application implements the soft stop policy, it requests a soft stop reservable license from the central registry license server.

12 The central registry license server checks its database for a soft stop reservable license and the user file for authorization.

13 The central registry license server returns the status of the request to the application. If a soft stop license was granted, the application runs.

14 If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Figure 9 shows what happens when an end user invokes a Java application or applet with reservable licenses. The primary differences between usage of reservable licenses for C and Java programs are that in the Java case:

The Web server machine, rather than the end user's machine, serves as the network license client.

Reserved licenses, when granted, are not moved to the nodelocked license server.

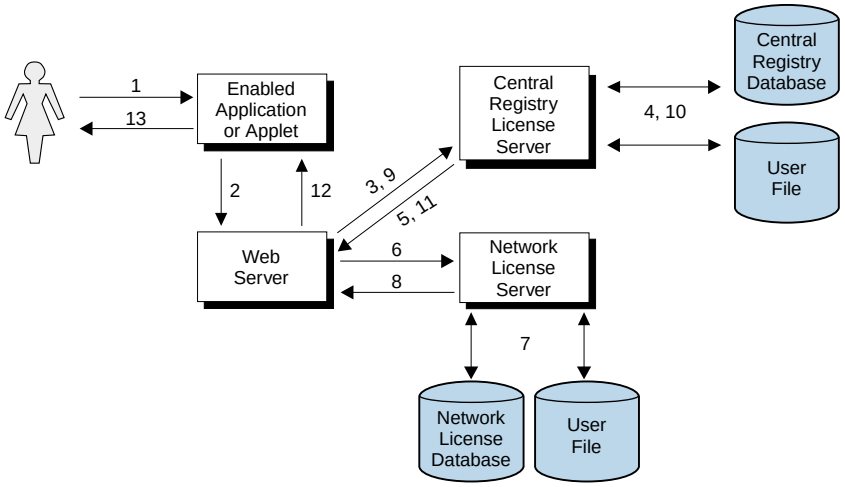

Figure 9. Using a Reservable License for a Java Application or Applet

1 The user invokes the application or downloads the applet through a Web browser.

2 The application or applet requests a reserved license from the Web server using the http protocol.

3. The Web server requests a reserved license for the application or applet from the central registry license server. This is a license that you have reserved for this user, group, or workstation.
4. The central registry license server checks its database for a reserved license and the user file for authorization.
5. The central registry license server returns the status of the request to the Web server. If a reserved license was found and granted, the Web server returns a positive status to the application or applet, and it runs.
6. If a reserved license was not found, the Web server requests a reservable license from the network license server. This is a reservable license that has not been reserved for anyone.
7. The network license server checks its database for a reservable license and the user file for authorization.
8. The network license server returns the status of the request to the Web server.
9. If no reservable license was found, the Web server requests a soft-stop license from the central registry license server.
10. The central registry license server checks its database for a soft-stop license and the user file for authorization.
11. The central registry license server returns the status of the request to the Web server.
12. The Web server returns the status of the request to the application or applet, using the http protocol. If the status is OK, the application or applet runs.
13. If no license can be granted, depending on how the vendor enabled the application or applet, it may return information to the end user, or it may run even with no license available.

Working with Licensed Products

Working with Per-Server Licenses

Figure 10 shows what happens when an end user invokes an application with per-server licenses when per-seat has not been enabled. In the figure, the application server is license-enabled.

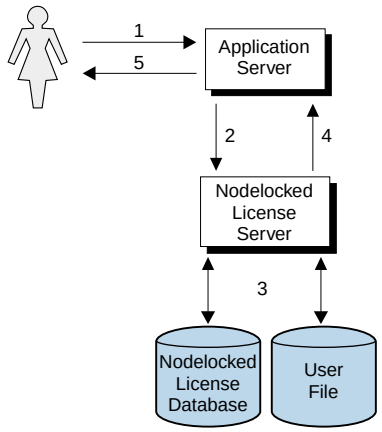

Figure 10. Using a Per-Server License

1 The application client user invokes the application.

2 The application server requests a per-server license from the nodelocked license server. This is a license that you have stored on the nodelocked license server.

3 The nodelocked license server checks the nodelocked license database for such a license and the user file for authorization.

If no license is found, but the application implements the soft stop policy, the nodelocked license server checks for a soft stop license.

4 The nodelocked license server returns the status of the request to the application server. If a license was found, or if a soft stop license was granted, the application runs.

5 If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Working with Per-Seat Licenses

Figure 11 shows what happens when an end user invokes a C-language application with per-server/per-seat licenses when per-seat has been enabled. In the figure, the application server is license-enabled.

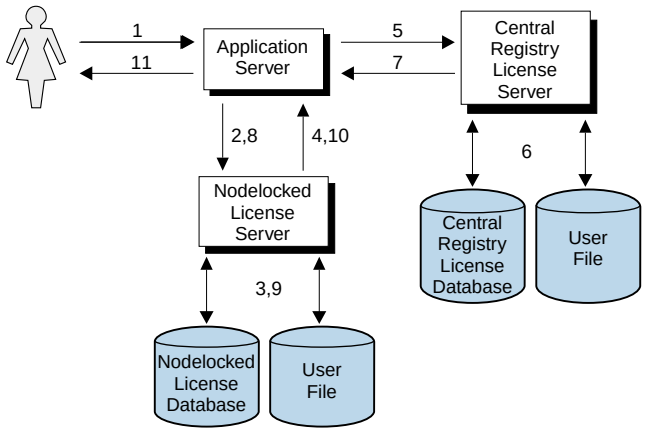

Figure 11. Using a Per-Seat License for a C-Language Program

1 The application client user invokes the application.

2 The application server requests a per-seat license from the nodelocked license server. This is a license that has already been granted to the user on a previous request and stored on the local machine.

3 The nodelocked license server checks the nodelocked license database for such a license.

4 The nodelocked license server returns the status of the request to the application server. If a per-seat license was found, the application runs.

5 If no per-seat license was found on the nodelocked license server, the application server requests a per-seat license from the central registry license server.

6 The central registry license server checks whether a license is already being used by the requesting application client, possibly granted through another application server. In such a case the application can start without having a new license granted.

Otherwise, the central registry license server checks whether a per-seat license is available. If so, it grants the license and records the application client identifier.

If no per-seat license is found, but the application implements the soft stop policy, the central registry license server checks for a soft stop license.

7 The central registry license server returns the status of the request to the application server.

8 If a per-seat or soft stop license was granted, the application sends a shadow copy of the granted per-seat license to the nodelocked license server.

9 The nodelocked license server adds the shadow copy to the nodelocked license database.

10 The nodelocked license server returns the status of the request to the application server, and the application runs.

11 If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

Figure 12 shows what happens when an end user invokes a Java application or applet with per-server/per-seat licenses when per-seat has been enabled. The primary differences between usage of per-seat licenses for C and Java programs are that in the Java case:

The Web server machine, rather than the end user's machine, serves as the network license client

When a per-seat license is granted no shadow copy is stored on the nodelocked license server.

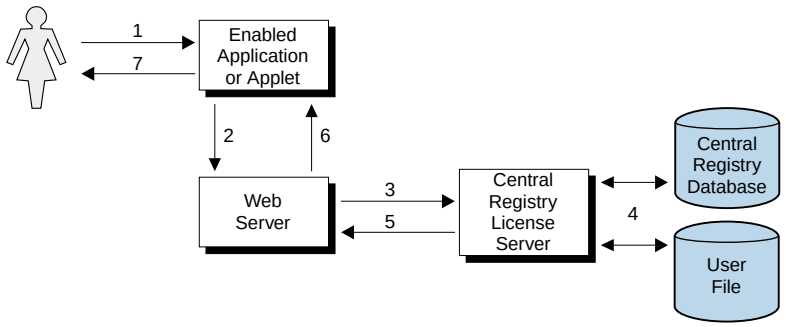

Figure 12. Using a Per-Seat License for a Java Application or Applet

1 The user invokes the application or downloads the applet through a Web browser.

2 The application or applet sends a license request to the Web server. It requests a per-seat license from the Web server, using the http protocol.

3 The Web server requests a license for the application or applet from the central registry license server.

4 The central registry license server checks whether a license is already being used by the requesting application client, possibly granted through another application server. In such a case the application can start without having a new license granted.

Otherwise, the central registry license server checks whether a per-seat license is available. If so, it grants the license and records the application client identifier.

If no per-seat license is found, but the application implements the soft-stop policy, the central registry license server checks for a soft-stop license. If a soft-stop license is available, it is granted.

5. The central registry license server returns the status of the request to the Web server.
6. The Web server returns the status of the request to the application or applet, using the http protocol. If the status is OK, the application or applet runs.
7. If no license can be granted, depending on how the vendor enabled the application or applet, it may return information to the end user, or it may run even with no license available.

Chapter 2. Planning Your Network Licensing Environment

This chapter is for the administrator who is setting up an environment to allow multiple client machines to share licenses stored on one or more network license servers. Skip this chapter if:

- You are using only applications with nodelocked licenses, and
- You do not intend to use the Basic License Tool from one machine to administer licenses on other machines

Designing the network licensing environment that provides the best solution for your business requires careful and thoughtful planning. This chapter will assist you, as the system administrator, to plan the environment for network licensed products and to decide on the configuration options you need.

It is important that you allow enough time for planning, especially when using License Use Runtime in large networks or across subnetworks.

The decisions you need to make include:

- How many network license servers you will set up
- Which machines, if any, will be the network license servers
- How you will distribute product licenses among the network license servers
- Which clients will have access to which servers
- How clients will locate the servers
- Whether all servers will serve all clients, or whether you will set up independent groups of servers and clients
- Which machine will be the central registry (if required)
- Whether you will make use of License Use Runtime high-availability licensing; if so, which network license servers will be part of a cluster
- Whether and how you will make use of remote administration to administer from one machine the network and nodelocked licenses on other machines
- Which machines will be used as Web servers for Java applications and applets

Selecting Your Servers

The license server system depends on a stable network. If name resolution and routing in a network are not running properly, then the network license servers, network license clients, and central registry license server may be unable to communicate properly.

Selecting Your Servers

In designating machines to be network license servers or the central registry license server, keep the following criteria in mind:

- If any other license management product is installed in your environment, it is recommended that you do not install a License Use Runtime server and a server of the other product on the same machine.
- A license server should be a computer that stays on at all times. Machines that are frequently unavailable or unreliable, such as those that are brought down often for testing or maintenance, are not good candidates.
- It is important to keep license servers for production environments separate from those for test environments.
- A computer that is already acting as a file server may well be a good choice, because it satisfies these criteria.
- If you have multiple subnetworks, then ideally, the servers should be on the same subnetwork as the majority of clients that will run the licensed products. Accessing in another subnetwork, across a bridge or router, may not be quite as fast.

If the network spans subnetworks, you need to spread the licenses out among network license servers. Also, inside the same network, each client request for a license generates network traffic. Therefore, it can be useful to spread the application licenses across more than one network license server, and across multiple platforms. When a computer is down, the licenses assigned to the network license server on that system are unavailable, but licenses assigned to other network license servers remain available. Having several license servers on the network will help to prevent bottlenecks that result when many clients communicate with a single network license server.

The number of network license servers in the network should be proportional to the frequency of license requests rather than to the number of users. For example, suppose that a compiler and a word processor are both license-enabled. A single user running many short compilations will place a heavier load on the license server than many users each starting the word processor once.

Computers that function as network license servers or as the central registry can also run the license-enabled products. The license server software does not have a noticeable effect on the performance of products.

When you have identified the network license servers, and before you configure them, you must organize the servers into one or more groups. The servers in a group form an independent licensing environment and serve a common set of clients. You must also identify a central registry license server for the group, if it requires one. “Selecting a Type of Network Binding” on page 37 will help you to group your servers.

Network Computing System (NCS)

The network computing system (NCS) is a set of tools for distributed computing, some of which are included in the License Use Runtime components.

NCS includes:

Remote Procedure Call (RPC) Runtime Library
: The backbone of the network computing system. It provides the calls that enable local programs to execute procedures on remote hosts. These calls transfer requests and responses between clients (the programs that call the procedures), and servers (the programs that execute the procedures). The RPC that is embedded in all license servers and in enabled products provides a common mechanism to support the request and acquisition of licenses.

Location Broker
: The location broker processes, as discussed under “Namespace Binding” on page 38, and tools to administer them.

After configuration, these mechanisms are transparent to the end user of the software product.

Installing on a Machine where NCS is Already Installed

When you install License Use Runtime, the NCS shipped as part of License Use Runtime does not replace any other NCS that is already installed on the machine. Instead, it is installed in parallel directories.

At any given time, you can have only one set of NCS processes active. It is strongly recommended that you select as License Use Runtime servers machines that are not servers for any other license management product. If you configure a machine as a License Use Runtime server, whenever an NCS process is started, any corresponding process that is already running is stopped.

A License Use Runtime server can provide licenses only for products that are enabled using the License Use Management Application Developer's Toolkit. It cannot provide licenses that require any other license management product, such as iFOR/LS or NetLS, that may be installed on the system.

On client machines, you can install products that were enabled using License Use Management, iFOR/LS, or NetLS. The client part of iFOR/LS or NetLS can coexist with the client part of License Use Management on the same system. You should configure License Use Runtime clients to work only with License Use Runtime servers.

Selecting a Type of Network Binding

License Use Runtime provides two types of network configuration to enable clients to locate (or *bind* to) the network license servers and the central registry. They are *direct binding* and *namespace binding*.

Binding

Direct Binding

Direct binding is the simpler binding mechanism, suitable for small networks and for networks that do not change frequently.

With direct binding, you make a list of your network license servers and the central registry. (The list is called the *direct binding servers list* in this chapter and in the configuration information in “Configuring Direct Binding” on page 69).

During configuration of servers and clients, you specify the network addresses of all the servers on the list. They are stored on every server and every client in a local text file, called the *configuration file*.

All network license servers, and the central registry license server, listen for incoming communications on well-known ports

All network license servers, and the central registry license server, listen for incoming communications on well-known ports (1515 and 10999). The network license client uses these port numbers, together with the network addresses of the server systems that are specified in the configuration file, to locate and connect to the servers.

For environments, with one or two network license servers, direct binding provides a simple, effective licensing environment.

In addition to enabling clients to locate license servers, the direct binding mechanism makes it possible, from any license server, to use the Basic License Tool remotely administer licenses on all the servers in the direct binding list. By adding nodelocked license servers to the direct binding servers list, you can administer licenses that are on remote nodelocked license servers.

Nodelocked license servers that are configured for remote administration listen on port 12999. The Basic License Tool uses this port number, together with the network addresses of the nodelocked license servers, to locate and connect to the servers for remote administration.

In the same way, you can enable remote administration from a machine configured only as a nodelocked license server. When you configure a nodelocked license server, you can create a direct binding servers list that contains all the license servers (both network and nodelocked) whose licenses you want to administer remotely.

Namespace Binding

As the licensing environment increases, keeping the direct binding environment up-to-date becomes more complex, and namespace binding becomes the better way to manage the license use management environment. Namespace binding is a powerful method of administering large client/server networks and networks that change frequently.

With namespace binding, one or more network license servers must run a subsystem called the *global location broker.* All the network license servers register themselves with the global location broker. The global location broker maintains a database of all

the network license servers and the license-enabled products for which they have licenses. When a client requests a license, the global location broker locates a server for the client.

The client machine does not need to have a list of all the network license servers. It needs only the address of a server on which the global location broker runs.

The global location broker dynamically updates network location information for each license server. If you configure new license servers, or move existing license servers to new locations on the network, licensed applications will always be able to find them.

You may want to set up your namespace binding environment so that some of the servers serve only some of the clients. Such a grouping of clients and servers is called an *NCS cell*, or just a *cell*.

When a network license client requests a license, only a license server in the same cell as the client can satisfy the request.

If multiple servers in the cell have licenses for the product, the servers are checked for an available license in random sequence. This automatically balancing the workload of the servers.

The cell is analogous to the direct binding servers list (see “Direct Binding” on page 38). You should configure each server as part of a direct binding servers list or a namespace binding cell, but not both.

Network license servers configured using namespace binding cells can support clients that locate the server through either namespace binding or direct binding.

In addition to enabling clients to locate license servers, the namespace binding mechanism makes it possible to use the Basic License Tool to do remote administration of licenses on all the servers in the cell.

Planning Direct Binding

Before you begin configuring machines to use direct binding, be sure you have identified all the servers in your direct binding servers list. Then go to the configuration scenarios in “Setting Up Your Servers and Clients” on page 63. As you configure direct binding at each server and at each client, be sure you enter exactly the same list of servers.

If any other license management product is installed in your environment, your direct binding servers list should include *only* License Use Runtime servers, *not* servers of the other product.

Performance Notes:

It is important that the direct binding servers list include all the servers, and that it exclude machines that will not actually function as servers. If there are any extra machines in the list, there will be a noticeable effect on performance.

Planning Namespace Binding

When a network license client configured for direct binding requests a license, and multiple servers have licenses for the product, the servers are checked for an available license in the sequence they were entered into the direct binding servers list during configuration of the client. Therefore, if you know how frequently specific network license clients will request licenses for specific products, you may be able to balance the workload of the servers by varying the sequence in which servers are defined at different clients.

If you are certain that all the licenses requested by a particular network license client will be supplied by a subset of the servers, when you configure the client you may configure direct binding with just those servers, rather than all servers in the direct binding servers list, to improve performance. If you configure the client in this way, make sure that it is configured to communicate with the necessary servers; otherwise, it will not be able to get licenses.

Planning Namespace Binding

In setting up namespace binding, you need to decide:

How you will group the servers and clients
Within each group, which servers will run the location brokers and other NCS tools

Planning Cells

In namespace binding, all nodes belonging to a cell are identified by the same name, called a universal unique identifier (UUID). The UUID is a 36-byte string that identifies the host on which it was created and the time at which it was created.

A node cannot be in more than one cell.

You can, optionally, place servers in the *default cell*, which has a default UUID. If a network license client is configured for namespace binding and is not configured as part of another cell, it joins the default cell.

You can create alternate cells to isolate individual departments or other groups of users. Be careful that different NCS users at your location do not inadvertently create two ore more default cells. Because the cells would have the same UUID, they would not be isolated from one another, and results would be unpredictable.

Important!

If any other license management product is installed in your environment, keep your License Use Runtime servers in separate cells from the other product's servers. Do not mix License Use Runtime servers and servers of any other license management product in the same cell.

Because cells cannot overlap, it is important to understand who should have access to which licenses before you configure your servers. In a production environment, you may want to configure all your license servers to run in the default cell. This simplifies the task of managing servers and allows a central administrator to control all the license

servers. However, if some licenses are to be restricted to a certain group of users, you may choose to install those licenses on servers running in an alternate cell, and make the clients that use the licenses part of that cell.

You should establish alternate cells for test environments, because the unstable nature of the test environment could negatively affect regular production users.

When you have decided which servers and clients will belong to each cell, go to the configuration scenarios in “Setting Up Your Servers and Clients” on page 63. As you configure namespace binding at each server and at each client, place it in the selected cell.

If you are setting up License Use Runtime on a machine that is not on a network, but you plan to use license-enabled products with network licenses, you need to have a network license server running on this machine. In this case, it is best to configure NCS to be in its own alternate cell.

Selecting the Location Brokers

If the network is small to medium in size with high-speed connections throughout, one global location broker is probably sufficient. Choose one of the network license servers to run the global location broker. If the network is large, it may be best to set up one server that runs the global location broker on each LAN.

When you are deciding which machines should run the global location broker, keep in mind that the process runs continuously in the background, waiting for a request for the function it provides. The function it provides is called infrequently. It is usually in wait state and has little effect on system performance.

In a namespace binding environment, each network license server and the central registry license server, including systems that run the global location broker, runs a process called the *local location broker*. The local location broker handles communication with the global location broker. When you configure a network license server, you specify whether it is to run just the local location broker or also the global location broker.

Running the Location Brokers

See “Setting Up Your Servers and Clients” on page 63 to configure your network and start the location broker processes. License Use Runtime also provides tools to administer the location brokers. To use them, see “Using NCS Tools” on page 42, and Chapter 5, “License Use Runtime Commands” on page 81.

Running the Global Location Broker Database Cleaner

The global location broker database cleaner is a process that should always be active; it automatically and periodically cleans up the global location broker databases.

When a license server starts up, it registers itself to the global location broker, to notify the global location broker of its network location information. When the server stops, it deletes itself from the global location broker database. Should the server accidentally

go down without being able to deregister itself, invalid entries remain in the global location broker database. The global location broker database cleaner deletes them.

Using NCS Tools

NCS provides tools that you can optionally use to administer your namespace binding environment.

Table 3. NCS Tools

Tool	***Description***
Local Broker Administration (lb_admin)	Administers the registration of the servers in global location broker or local location broker databases. It can be used to look up information, add new entries, and delete existing entries in a specified database.
GLBD Replicas Administration (drm_admin)	Monitors and modifies the list of the replicated versions of the global location broker databases. It can be used to modify, or merge databases to force convergence among replicas, to stop servers, and to delete replicas.
GLBDs List (lb_find)	Lists the servers running the global location broker in the network.
Universal Unique Identifier generator (uuid_gen)	Generates the UUID for an NCS cell.

For a detailed explanation of how to use these tools, see Chapter 5, “License Use Runtime Commands” on page 81.

Reaching a Global Location Broker in a Different Subnetwork

Normally, products on network license clients contact a global location broker by broadcasting on the local network. If your system does not support broadcasting, or if the global location broker is running on a license server in a separate subnetwork, you need to set up an alternate mechanism to enable the machine to locate a global location broker. The mechanism is the file *glb_site.txt*, which you create on the machine that needs to reach a global location broker.

The glb_site.txt file lists the network addresses of servers where a global location broker may be running. A machine that has a glb_site.txt file tries these addresses in order. Once it locates a server that is running the global location broker, it can locate network license servers. If it does not locate a server that is running the global location broker, the machine does not broadcast.

See “Configuring to Reach a Global Location Broker in a Different Subnetwork” on page 72 for information about how to create the glb_site.txt file.

Planning the Central Registry

The *central registry license server* subsystem provides a mechanism for storing licensing information in a database common to all the servers. It is used for the administration of customer-managed use products and products with reservable

licenses. There must be one and only one central registry subsystem running in a cell, in the case of namespace binding, and one and only one running in a direct binding servers list, in the case of direct binding. This ensures that the data is accurate and complete.

1. You use the configuration tool to specify where to start the central registry.
2. Select the node where you will run the central registry carefully. After you place the central registry on a node, it cannot be moved.
3. The machine running the central registry must be up and running in order to perform administration tasks on network customer-managed products or on products using reservable licenses.

Planning for Java Applications and Applets

If you plan to use license-enabled Java applications or applets, you will need to set up one or more Web server machines. License-enabled Java applications and applets request licenses from the Web server. The Web server machine, in turn, serves as the network license client.

On the Web server machine, you must:

Run one of the following operating systems:

| AIX 4.3.1, 4.3.2, or 4.3.3
| Windows NT 4.0 (x86 processor)
OS/2 Warp 4.0
| Solaris 2.6 (with the native threads package) or 2.7

Install a Web server and IBM WebSphere. For details about software requirements, see “Installing LUM Java Client Support on Solaris” on page 61.

Install License Use Runtime Version 4.5.5 and LUM Java Client Support.

Configure the machine as a network license client, to communicate with network license servers where the licenses for the Java applications and applets are stored.

On each machine where license-enabled Java applications run, in the Java home directory, you must create a file named LicenseClient.properties. The contents of this file must be `url=http://hostname`, where `hostname` is the TCP/IP hostname of the Web server machine. This file identifies the Web server to which license requests are to be directed.

Planning Clusters

To take advantage of License Use Runtime high-availability licensing, you set up *clusters* of network license servers connected through TCP/IP. For concurrent licenses

with vendor-managed use control only, the software vendor generates passwords that are bound to the cluster rather than to a single server. Some of the servers in a cluster serve licenses, while others wait in reserve to take over in case a serving server goes down. The servers that are serving at any time share equally the responsibility for the licenses that are bound to the cluster, and keep one another informed about the status of the licenses.

You can create and administer a cluster, and administer high-availability licenses, from any machine. However, only AIX, HP-UX, IRIX, Solaris, Windows NT, Windows NT Alpha, and Windows Terminal Server network license servers can be members of a cluster: no OS/2, Windows 95, or Windows 98 network license server can be a member of a cluster.

A network license server that is a member of a cluster can serve licenses that are bound to the server and participate as a member of a cluster at the same time.

High-availability licensing is recommended only for users who are already experienced with managing individual license servers and who already have a stable licensing environment working.

Restrictions on Cluster Size and Composition

For security reasons, it is necessary to impose strict rules on the size and composition of clusters. Be very careful when you decide how many and which servers to put in a cluster. You will not be able to change your decisions after the fact, and they will affect the size and composition of the cluster for as long as it exists.

When you create a cluster (using the GUI or the command-line interface; see “Example 6: Creating and Administering a Cluster” on page 78), you specify the initial number of servers in the cluster, and which servers they are. The initial number must be in the range 3 through 10. The first server assigned to the cluster is automatically *activated*; that is, it is available to participate in serving licenses as part of the cluster. You must explicitly activate the other members.

The initial number of servers dictates two important attributes of the cluster:

The minimum number of servers that must be activated in the cluster for the cluster to work

The maximum number of servers that can be added to the cluster in addition to the initial number

If you want to replace a server machine that is one of the initial minimum number, to upgrade the hardware or to replace failing hardware, add a new server to the cluster. The number of new servers you can add, even to replace other servers, is limited.

Attention: The initial minimum number of servers must always be in the cluster; they must not be deactivated. If any is deactivated, the cluster ceases to serve licenses.

Planning Clusters

Passwords that are bound to a cluster are usable on only that cluster. If you find it necessary to delete a cluster and create a new one, or to create additional clusters, you will not be able to use existing passwords on the new cluster.

To delete a cluster, deactivate all its members. When you deactivate a server, it must be up and running.

After a cluster has been created and its members have been activated, the number of activated members determines how many servers must be up and running for the cluster to function.

The relationships between these cluster attributes is shown in Table 4 on page 46.

The minimum number of servers up and running, as shown in the table, is the number of servers that serve licenses. All servers beyond that number are in reserve, waiting to take over if a serving server goes down.

Table 4. Number of Servers in a Cluster

Initial Number	Min No. Activated for Cluster to Work	Max No. Added after Cluster Creation	Actual No. Activated	Min No. Up & Running for Cluster to Work
3	2	1	2 3 4	2 2 3
4	4	2	4 5 6	3 3 4
5	4	1	4 5 6	3 3 4
6	6	2	6 7 8	4 4 5
7	6	1	6 7 8	4 4 5
8	8	2	8 9 10	5 5 6
9	8	1	8 9 10	5 5 6
10	10	2	10 11 12	6 6 7

Examples of Cluster Size Rules

Example 1: Initial number of servers is 3

The following rules apply:

During the life of the cluster, you can add only one server to the cluster. This means you can add a new server with upgraded hardware, and deactivate one of the original three servers, only once during the life of the cluster. This scenario has the effect of replacing a server with an upgraded machine.

Alternatively, you can add a fourth server to the cluster without deactivating any of the original three, thus increasing the cluster size to four servers. Again, you can add a server only once.

No matter whether the cluster has three or four members, at least two members must be activated for the cluster to work.

More than half of the activated servers must be up and running for the cluster to work.

Table 5 on page 47 shows how the servers are deployed, depending on how many are activated and how many are up and running.

Table 5. Example - Cluster with Three Initial Members

Number of Activated Members	Number of Members Up and Running	Number of Members Serving Licenses	Number of Servers In Reserve
2	2	2	0*
3	2	2	0*
	3	2	1
4	3	3	0*
	4	3	1

Note: * When the number of servers in reserve is 0, there is no high-availability advantage.

Example 2: Initial number of servers is 6

The following rules apply:

During the life of the cluster, you can add two servers to the cluster. This means you can add two new servers with upgraded hardware, in effect replacing two servers with upgraded machines.

Alternatively, you can add one or two servers to the cluster without deactivating any of the original six, thus increasing the cluster size to seven or eight servers.

Whether the cluster has six, seven, or eight members, at least six members must be activated for the cluster to work.

More than half of the activated servers must be up and running for the cluster to work.

Table 6 on page 48 shows how the servers are deployed, depending on how many are activated and how many are up and running.

Network Examples

Table 6. Example - Cluster with Six Initial Members

Number of Activated Members	Number of Members Up and Running	Number of Members Serving Licenses	Number of Servers In Reserve
6	4	4	0*
	5	4	1
	6	4	2
7	4	4	0*
	5	4	1
	6	4	2
	7	4	3
8	5	5	0*
	6	5	1
	7	5	2
	8	5	3

Note: * When the number of servers in reserve is 0, there is no high-availability advantage.

Cluster Membership Considerations

If you use direct binding, be sure all network license clients that will use licenses bound to the cluster, and all servers that are members of the cluster, have all the servers of the cluster in their direct binding server list in order to exploit fully the high availability of licenses.

If you use namespace binding, all the servers in a cluster and all their network clients must be in the same cell in order to exploit fully the high availability of licenses.

A server can be activated in only one cluster at any time. If you assign a server to a cluster and never activate it, or explicitly deactivate it, it can join a second cluster and be activated there. But in this case, the server cannot be activated in its original cluster, and no other server can be substituted in the original cluster. To reactivate the server in its original cluster, you must first deactivate it in the second cluster.

Do not disable remote administration on a server that is part of a cluster. If you do, you may have problems enrolling and removing licenses bound to the cluster.

Verifying Network Connections

License Use Runtime provides the i4tv tool to verify that license servers are running properly. For a detailed explanation of how to start the tool and how to use it, see Chapter 5, “License Use Runtime Commands” on page 81.

Network Examples

This section shows some of the possible network configurations you can have in your environment. The examples, for simplicity, show an environment with at most five network license clients, three network license servers, and two nodelocked license servers.

Network Examples

Figure 13 on page 49 shows a configuration where all the required NCS subsystems run on the same server.

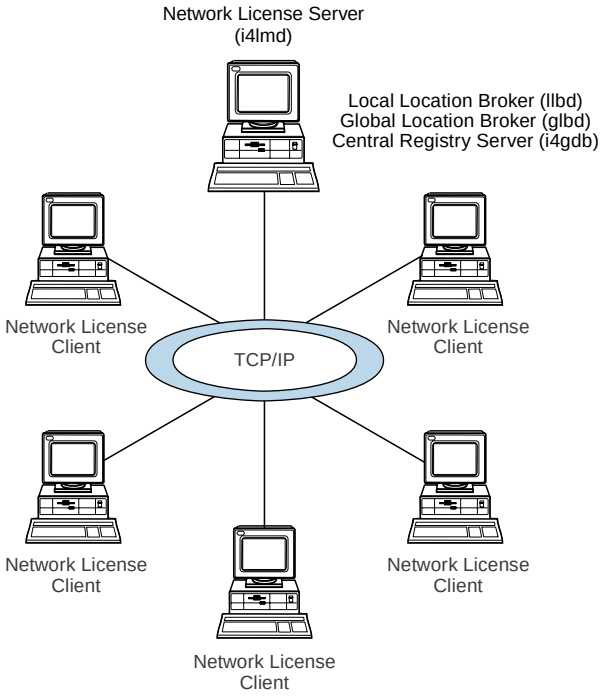

Figure 13. NCS Cell with All the Subsystems on the Same Server

Network Examples

Figure 14 shows a network with two network license servers (A and D) and two nodelocked license servers (B and C). One network license server, and both nodelocked license servers, run only the local location broker, which is mandatory on all servers. One network license server also runs the central registry and the global location broker. From any of the license servers, it is possible to administer licenses on all the license servers.

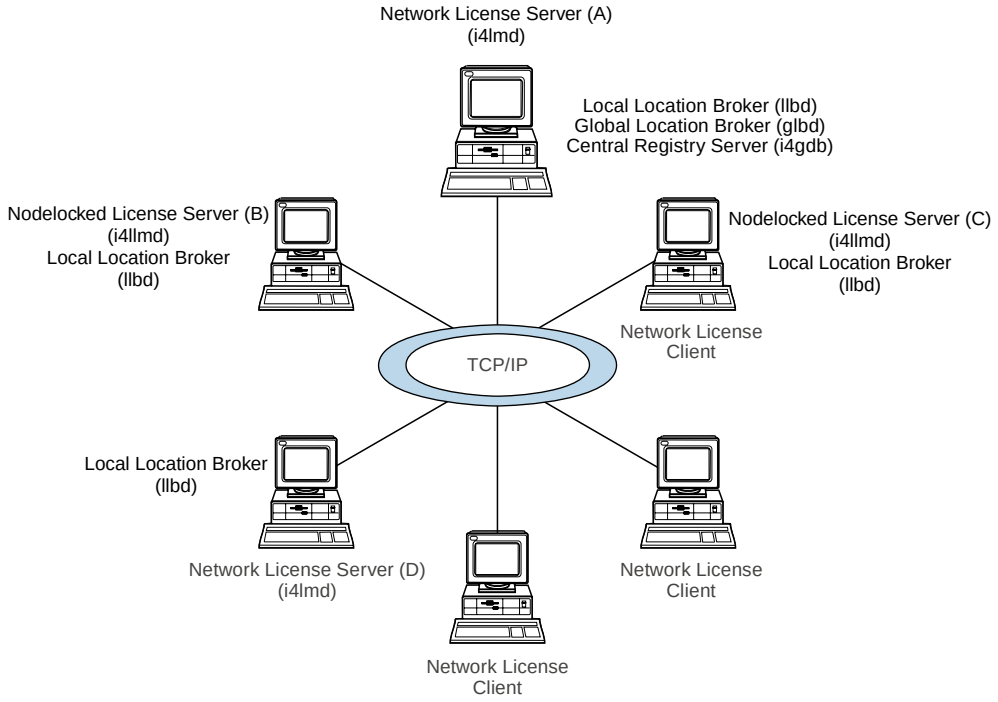

Figure 14. NCS Cell with Network License Servers and Nodelocked License Servers

Network Examples

Figure 15 shows a network with three network license servers. This example shows that more than one license server in the network can run the global location broker but with only one central registry. Server C runs a global location broker that is a replica of the first one that was started on server B.

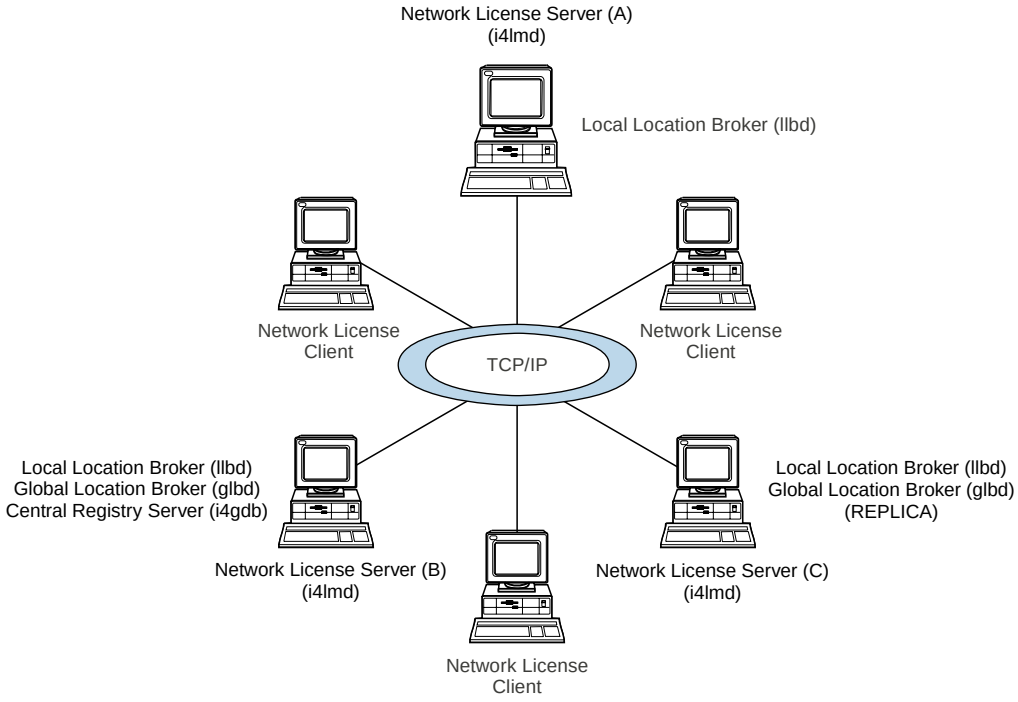

Figure 15. NCS Cell with Three Network License Servers and Three Clients

Network Examples

Figure 16 shows an example of a network configuration that uses direct binding. The example shows a network license server and two nodelocked license servers in a network. From any of the license servers, it is possible to administer licenses on all the license servers.

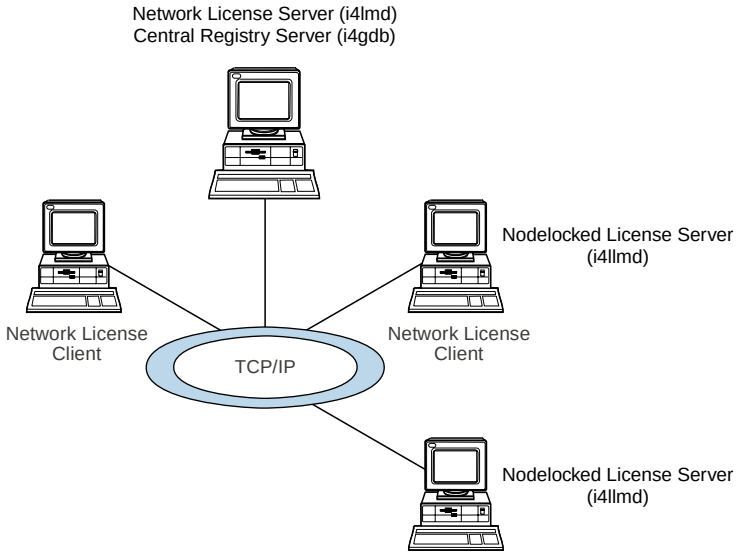

Figure 16. Direct Binding with Network License Servers and Nodelocked License Servers

Network Examples

Figure 17 shows an example of a Java configuration. The example shows two network license servers, a network license client, and a network application client in a Java-enabled network. Server A communicates with the Java-enabled applet or application by means of the Hypertext Transfer Protocol (HTTP), if necessary being protected by a firewall.

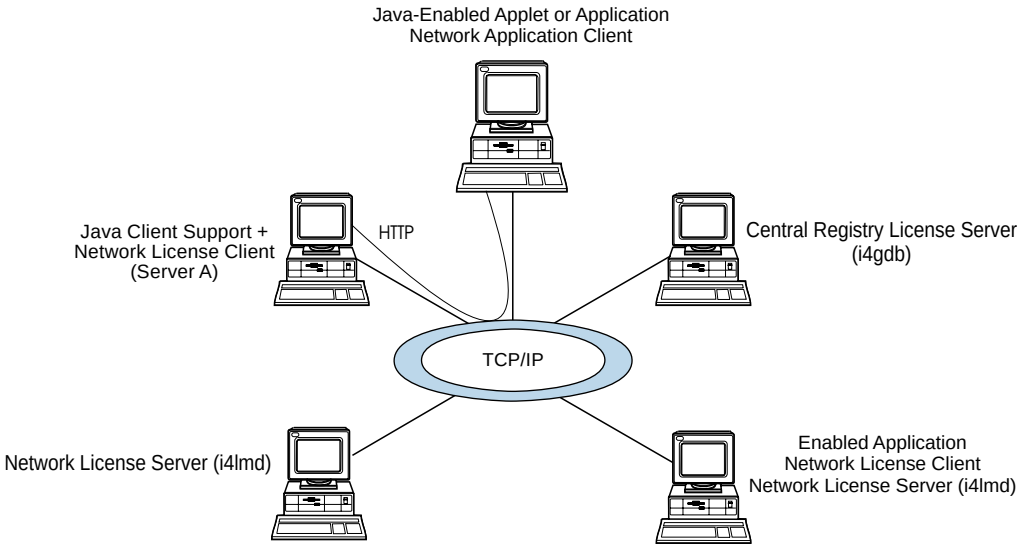

Figure 17. Direct Binding with Java Client Support

Chapter 3. Installing License Use Runtime

This chapter is intended for administrators using license-enabled products and software developers who are license-enabling products.

This chapter includes information about:

- What to do before installing License Use Runtime
- Installing License Use Runtime
- What to do after installing License Use Runtime
- Uninstalling License Use Runtime

Important!

If any other license management product is installed on the machines in your environment, you must be aware of certain requirements before you install. Be sure you read Chapter 2, “Planning Your Network Licensing Environment” on page 35 carefully. Considerations for coexistence with other license management products are in the sections “Selecting Your Servers” on page 35, “Installing on a Machine where NCS is Already Installed” on page 37, “Planning Direct Binding” on page 39, and “Planning Cells” on page 40.

Installing License Use Runtime on HP-UX

License Use Runtime Version 4.5.5 contains one package, named LUM - License Use Runtime Base Code.

Before Installing License Use Runtime

To determine whether License Use Runtime Version 4.5.5 is already installed on your machine, refer to the file:

`/var/lum/VERSION`

Before you install License Use Runtime:

- Allocate at least 20 MB of disk space in the /opt file system.
- Delete any directory (not a symbolic link to a directory) named `/var/lum`.

Hardware and Software Requirements

License Use Runtime requires no hardware other than a system that can run HP-UX 10.20 or 11.0.

TCP/IP must be installed.

A Web browser is required to view the .HTM documentation.

Obtaining the License Use Runtime Code

You can get the License Use Runtime code:

From the product CD-ROM.

By downloading License Use Runtime from the Web. Download the TAR file from http://www.software.ibm.com/is/lum. Use the command `tar -xvf` to decompress the file so that you have the License Use Runtime package ready for installation.

With an enabled application, if the vendor chose to redistribute the License Use Runtime code.

Installing the License Use Runtime Package

Follow these steps:

1. Login with root authority.

Use SAM to install License Use Runtime, as follows:

2. Click on the **Software Management** icon.
3. Select **Install Software to Local Host** icon.
4. In the Specify Source window, set **Source Depot Type** to **Local CD-ROM** or **Local Directory**. Set **Source Depot Path** to the file system where the CD-ROM is mounted or the directory where you have downloaded the code. Click on **OK**.
5. Select **License Use Management Runtime Base Code** from the list of programs.
6. From **Actions**, select **Mark for Install**.
7. From **Actions**, select **Install (Analysis)**.
8. When the analysis is complete, click on **OK**, then **Yes** to confirm installation, and then **Done**.

If you do not have a graphical terminal, use the `swinstall` command.

For more information about installation in general, refer to the `swinstall` MAN pages.

After Installing License Use Runtime

After installation, before starting License Use Runtime, login with root authority and add the following path to the `.profile` in the PATH environmental variable:

`/var/lum`

Then exit and login again to allow changes to take effect.

Uninstalling License Use Runtime

To remove License Use Runtime from your machine:

1. Login with root authority.

Use SAM to uninstall License Use Runtime, as follows:

2. Click on the **Software Management** icon.
3. Select **Remove Software** and then **Remove Local Host Software**.
4. Select **License Use Management Runtime Base Code** from the list of programs.
5. From **Actions**, select **Mark for Remove**.
6. From **Actions**, select **Remove (Analysis)**.
7. When the analysis is complete, click on **OK**, then **Yes** to confirm uninstallation, and then **Done**.

If you do not have a graphical terminal, use the `swremove` command.

Installing License Use Runtime on IRIX

License Use Runtime Version 4.5.5 contains one package, named LUM - License Use Runtime Base Code.

Before Installing License Use Runtime

To determine whether License Use Runtime Version 4.5.5 is already installed on your machine, refer to the file:

`/var/lum/VERSION`

Before you install License Use Runtime, allocate at least 20 MB of disk space in the /opt file system.

Hardware and Software Requirements

License Use Runtime requires no hardware other than a system that can run IRIX 6.3, 6.4, or 6.5.

IRIX 6.3, 6.4, or 6.5 with the POSIX Patch Set or its successor must be installed.

TCP/IP must be installed.

A Web browser is required to view the .HTM documentation.

Installing License Use Runtime

Obtaining the License Use Runtime Code

You can get the License Use Runtime code:

From the product CD-ROM.

By downloading License Use Runtime from the Web. Download the License Use Runtime package from http://www.software.ibm.com/is/lum.

With an enabled application, if the vendor chose to redistribute the License Use Runtime code.

Installing the License Use Runtime Package

Follow these steps:

1. Login with root authority.
2. Do either of the following:

   Enter the command `swmgr`
   From the desktop, select **System**, then **Software Manager**.
3. Select **Default Installation**.
4. In the **Available Software** field, enter the file system where the CD-ROM is mounted or the directory in which you have downloaded the code.
5. Select **Start** to begin installation.
6. When installation ends, select **OK**.

If you do not have a graphical terminal, use the `inst` utility.

For more information about installation in general, refer to the `inst` MAN pages.

After Installing License Use Runtime

After installation, before starting License Use Runtime, login with root authority and add the following path to the `.profile` in the PATH environmental variable:

`/var/lum`

Then exit and login again to allow changes to take effect.

Uninstalling License Use Runtime

To remove License Use Runtime from your machine,

1. Login with root authority.
2. Do either of the following:

   Enter the command `swmgr`
   From the desktop, select **System**, then **Software Manager**.
3. Select **Manage Installed Software**, to display a list of installed products.

4. From the list of products, select **License Use Management Runtime 4.5.5**, then check **Remove**.
5. Select **Start** to begin uninstallation.
6. When uninstallation ends, select **OK**.

If you do not have a graphical terminal, use the `inst` utility.

Installing License Use Runtime on Solaris

License Use Runtime Version 4.5.5 contains one package, named LUM - License Use Runtime Base Code.

Before Installing License Use Runtime

To determine whether License Use Runtime Version 4.5.5 is already installed on your machine, refer to the file:

`/var/lum/VERSION`

Before installing License Use Runtime, allocate at least 20 MB of disk space in the /opt file system.

Hardware and Software Requirements

License Use Runtime requires no hardware other than a system that can run Solaris 2.6 or 2.7.

TCP/IP must be installed.

A Web browser is required to view the .HTM documentation.

Obtaining the License Use Runtime Code

You can get the License Use Runtime code:

From the product CD-ROM.

By downloading License Use Runtime from the Web. Download the License Use Runtime package from http://www.software.ibm.com/is/lum.

With an enabled application, if the vendor chose to redistribute the License Use Runtime code.

Installing the License Use Runtime Package

Follow these steps:

1. Login with root authority.

Use `admintool` to install License Use Runtime as follows:

2. From **Browse**, select **Software**.

Installing on Solaris

Perform the following step only if you have already installed License Use Runtime Version 4.0 and you are now installing a later release:

3. Select **Properties --> Package Administration**, and then change the value of **Existing Packages** to **Overwrite**.
4. From **Edit**, select **Add**.
5. In **Software Location**, select **Hard Disk** or **CD-ROM,** and fill in the file system where the CD-ROM is mounted or the directory where you have downloaded the code.
6. Select **License Use Management Runtime Base Code**, and click on **Add**.

If you do not have a graphical terminal, use the `pkgadd` command:

```
pkgadd -d
```

or

```
pkgadd -d /CDROM
```

where /CDROM is the name of the file system where the CD-ROM is mounted.

If you have already installed License Use Runtime Version 4.0 and you are now installing a later release, the pkgadd command must also contain the parameter:

```
-a lumadmin
```

For more information about installation in general, refer to the *Solaris System Administration Answer Book.*

After Installing License Use Runtime

After installation, before starting License Use Runtime, login with root authority and add the following path to the `.profile` in the PATH environmental variable:

```
/var/lum
```

Then exit and login again to allow changes to take effect.

Uninstalling License Use Runtime

To remove License Use Runtime from your machine:

1. Login with root authority.

Use `admintool` to uninstall License Use Runtime as follows:

2. From **Browse**, select **Software**.
3. Select **License Use Management Runtime Base Code**.
4. From **Edit**, select **Delete**.
5. Select **Delete** to confirm, and **Y** when asked if you want to remove this package.

If you do not have a graphical terminal, use the `pkgrm` command.

Installing LUM Java Client Support on Solaris

LUM Java Client Support contains one package, named LUM Java Client Support.

Before Installing LUM Java Client Support

Disk Space Requirements

LUM Java Client Support requires approximately 200 KB of disk space.

Software Requirements

The following software is required:

| Solaris 2.6 with the Native Threads Package, or 2.7.

If you are using Lotus Domino Go Webserver 4.6.2.5 or Netscape Enterprise Server Version 3, the SunOS kernel update patch 104283-04 is required.

One of the following Web servers:

- Lotus Domino Go Webserver 4.6.*x*
- Netscape Enterprise Server 3.0.1 or 3.5.1; 3.5.1 is recommended
- Netscape FastTrack Server 2.0.1 or 3.0.1

Either Java Development Kit (JDK) 1.1.4 or 1.1.6 with the Native Threads Package, or Java Runtime Environment (JRE) 1.1.4 or 1.1.6 with the Native Threads Package JDK 1.1.6 is shipped with the IBM WebSphere Application Server.

| IBM WebSphere Application Server 1.1 or 2.0

The License Use Runtime Version 4.5.5 runtime code

Obtaining LUM Java Client Support Code

You can get the LUM Java Client Support code:

From the product CD-ROM. LUM Java Client Support is in the `servlet` directory.

By downloading the package from the Web. Download the TAR file from http://www.software.ibm.com/is/lum. Use the command `tar -xvf` to decompress the file so that you have the LUM Java Client Support package ready for installation.

With an enabled application, if the vendor chose to redistribute the LUM Java Client Support code.

Installing on Solaris

Installing the LUM Java Client Support Package

Follow these steps:

1. Login with root authority.

Use `admintool` to install LUM Java Client Support as follows:

2. From **Browse**, select **Software**.
3. From **Edit**, select **Add**.
4. In **Software Location**, select **Hard Disk** and fill in the directory where you have downloaded the code.
5. Select **LUM Java Client Support,** and click on **Add**.
6. LUM Java Client Support is installed in the directory `/usr/lpp/IBMWebAS/classes/com/ibm/licUseMgmt`. One file is installed in `/usr/opt/lum/ls/os/solaris/dll`.

If you do not have a graphical terminal, use the `pkgadd` command:

```
pkgadd -d
```

or

```
pkgadd -d /CDROM
```

where /CDROM is the name of the file system where the CD-ROM is mounted.

For more information about installation in general, refer to the *Solaris System Administration Answer Book.*

Uninstalling LUM Java Client Support

To remove LUM Java Client Support from your machine,

1. Login with root authority.

Use `admintool` to uninstall LUM Java Client Support as follows:

2. From **Browse**, select **Software**.
3. Select **LUM Java Client Support.**
4. From **Edit**, select **Delete**.
5. Select **Delete** to confirm, and Y when asked if you want to remove this package.

If you do not have a graphical terminal, use the `pkgrm` command.

Chapter 4. Getting Started with License Use Runtime

This chapter describes how to set up your License Use Runtime environment and how to manage both nodelocked and network licensed products.

Setting Up Your Servers and Clients

After you install License Use Runtime, you must configure the program on each machine.

You configure using a configuration tool. This tool has a script interface and a command line interface. The information you provide is saved in a configuration file. When you start License Use Runtime, it uses the information in this file to direct the behavior of the local system in the licensing environment. See Appendix A, “License Use Runtime Configuration File” on page 173 for reference information on the configuration file. The configuration details depend on the role the machine will play in your licensing environment and the types of licenses you need to handle.

Important!

If any other license management product is installed on the machines in your environment, you must be aware of certain requirements before you begin configuring servers and clients. Be sure you read Chapter 2, “Planning Your Network Licensing Environment” on page 35 carefully. Considerations for coexistence with other license management products are in the sections “Selecting Your Servers” on page 35, “Installing on a Machine where NCS is Already Installed” on page 37, “Planning Direct Binding” on page 39, and “Planning Cells” on page 40.

Configuring to Handle Nodelocked Licenses

To handle only products with nodelocked licenses, you need only configure the machine as a nodelocked license server (“Using the Configuration Tool Command-Line Interface” on page 70) and have the nodelocked license server up and running. Every machine is automatically configured as a nodelocked license server when License Use Runtime is installed. You do not have to do any configuration unless you want to change the default configuration.

If a machine configured as a nodelocked license server is in a network, instances of the Basic License Tool on other machines can administer licenses on the nodelocked license server. With a little additional configuration (“Using the Configuration Tool Command-Line Interface” on page 70), you can run the Basic License Tool on the nodelocked license server machine and administer licenses stored on remote network license servers, nodelocked license servers, and the central registry.

Configuration

Configuring to Handle Network Licenses

To handle products with network licenses, you must configure at least one network license server and configure each client as a network license client If you use products with customer-managed or reservable licenses, you must also configure one server as the central See “Using the Configuration Tool Command-Line Interface” on page 70 for information about how to configure.

Determining the Configuration Required

You can configure a machine to play more than one role in your licensing environment. For example, if you configure the same machine as a nodelocked license server, a network license server, and the central registry license server, that machine can handle all types of licensed products.

If you know the types of licenses used by the license-enabled products in your environment, Table 7 on page 65 will help you to determine how to configure:

The license servers for the application
The machines that will request licenses for the application

Consult the documentation of the license-enabled products for the license types and other information about the enabling that might affect your configuration. In any case, the enrollment certificate file that you receive from the vendor shows the type of license.

Table 7. Configuration Required to Support All Types of Licenses

License Type	License Requester	License Server
Simple Nodelocked (Non-Runtime-Based Enabling)[1]	License-Enabled Application	None
Simple Nodelocked (Runtime-Based Enabling)[1]	License-Enabled Application	Nodelocked License Server on the Same Machine as the Application
Use-Once Nodelocked	License-Enabled Application	Nodelocked License Server on the Same Machine as the Application
Concurrent Nodelocked	License-Enabled Application	Nodelocked License Server on the Same Machine as the Application
Per-Server	License-Enabled Application	Nodelocked License Server on the Same Machine as the Application
Use-Once	Network License Client	Network License Server[2] + Central Registry License Server[3]
Concurrent	Network License Client	Network License Server[2] + Central Registry License Server[3]
Per-Seat	Network License Client + Nodelocked License Server	Central Registry License Server[3]
Reservable	Network License Client + Nodelocked License Server[4]	Network License Server[2] + Central Registry License Server[3]

Notes:

1. If the vendor enabled the product with simple nodelocked licenses and delivered the licenses to you in a compound network password, you must also:

   Configure a network license server, where you will install the compound password, and

   Configure the local machine (where the nodelocked license is to be installed) as a network license client of that server.

2. You can configure one or more network license servers.
3. You can configure only one central registry license server. For customer-managed use products, the central registry is required. It enables you to update the count of product licences, implement the hard stop or soft stop policy, or track the high-water mark. Note that because per-seat licenses are always customer-managed, they require the central registry. For reservable licenses, the central registry is required if you want to be able to reserve licenses for specific users.
4. For reservable licenses, the nodelocked license server is required for the end user to get a reserved license.

Configuration

Before You Configure

Before you begin the configuration process, for every machine you are going to configure, you need to decide which roles it will play and how you plan to set up direct binding or namespace binding. You might also decide to override some of the configuration defaults. The actions you can take during configuration are summarized in Table 8 on page 67. Check the table for all the roles your machine will play, and make all the indicated decisions before you start configuration.

Configuration

Table 8. Configuration Options

Configuration Options	Nodelocked License Server	Network License Client	Network License Server	Central Registry License Server
Customize selection of information logged or accept default? (“Customizing Log Information” on page 68)				
Change log path or accept default? (“Customizing Log Information” on page 68)				
Start license servers at system startup (default is no)? (“Automatically Starting License Servers” on page 68)				
Disable remote administration of network license servers (default is no)? (“Disabling Remote Administration” on page 68)				
Disable remote administration of nodelocked license servers (default is no)? (“Disabling Remote Administration” on page 68)				
Set up direct binding (“Configuring Direct Binding” on page 69)				
Prepare a list of nodelocked license servers for remote administration	1			
Have your direct binding servers list ready	1			
Know which machine is the central registry	1			
Change default direct binding ports?	1			
Set up namespace binding (“Configuring Namespace Binding” on page 69)				
Join an existing alternate cell?	1,2			
Know the name of a server already there	1,2			
Run a replica GLB?	1,2			
Join the default cell?	1,2			
Run a replica GLB?	1,2			
Start a new cell?	1,2			

Notes:

1. Applicable if you plan to run the Basic License Tool and administer licenses on other machines from this nodelocked license server.
2. Applicable if you plan to run the Basic License Tool and administer licenses on other machines from this nodelocked license server, *or* if you want instances of the Basic License Tool on other machines to be able to administer licenses on his machine.

Configuration

Customizing Log Information

For any license server, you can customize the selection of events that are logged, and the location of the log file. Note that if you want to change the location of the log from the default to a path you choose, the directory you specify must already exist. Otherwise, you will lose the logging function.

The following events can be logged:

All events
: Includes all the events in the list.

Errors
: Describes server errors that do not stop the server, but return a status code and a message. This is logged by default.

License timeout
: Tells you that the server has canceled the request for a license because the check period expired. This is not logged by default.

License wait
: Tells you when a license request cannot be satisfied because no licenses are available, and the user is added to a queue. This is not logged by default.

License checkin
: Tells you when a licensed product has sent a check-in call to the server to notify that the product is running. This is not logged by default.

License grant/release
: Tells you when a license was granted or released. This is not logged by default.

Vendor added/deleted
: Tells you when a product of a new vendor was registered or deleted. This is logged by default.

Vendor messages
: Provides the log messages the vendor inserted in the enabled product. This is logged by default.

Product added/deleted
: Tells you when a new product was registered or deleted. This is logged by default.

Server start/stop
: Logs the successful start or stop of the license server. This is not logged by default.

Automatically Starting License Servers

During configuration of any license server, you can specify that license servers should start automatically when you start the machine. Otherwise, you must remember to start the services manually after configuration and before using the Basic License Tool or any enabled applications.

Disabling Remote Administration

When you configure a network or nodelocked license server, you can specify that licenses stored on that server cannot be administered from any other license server.

Configuring Direct Binding

When you configure a network license server, network license client, or central registry license server that is to be part of a direct binding environment, you must have your direct binding servers list ready. (See “Direct Binding” on page 38.) If you configure in this way, clients will be able to locate the server only through direct binding.

When you configure a nodelocked license server, network license server, or central registry license server that is to be part of a direct binding environment, you must also have ready a list of nodelocked license servers whose licenses you want to administer remotely from this machine.

You will enter the hostnames or network addresses of all the servers in the list (other than the nodelocked license server on the local machine, which is added to the list automatically). You will also designate which server, if any, is the central registry.

When you configure the servers in the direct binding servers list, and the clients that will use them, be sure you define exactly the same set of servers on each.

You can change the default port numbers for nodelocked license servers, network license servers, and the central registry license server. Do not change the defaults unless they are already in use by other applications.

If you are running License Use Management Java Client Support on the same machine and want to change the direct binding list:

1. Stop License Use Management and the Web server.
2. Change the direct binding list.
3. Restart License Use Management and the Web server.

Configuring Namespace Binding

When you configure a nodelocked license server, network license server, network license client, or central registry license server to be part of a namespace binding environment, clients will be able to locate the server, and the Basic License Tool will be able to locate remote servers, through either namespace binding or direct binding.

You must know which cell this machine is to be part of (see “Planning Cells” on page 40 and “Selecting the Location Brokers” on page 41). If the machine is to join an existing cell, other than the default cell, you must be able to identify a server that is already in the cell.

If there are other users of NCS at your location who might create a default cell, it is safer to configure only alternate cells. Since the default cell always has the same UUID, results would be unpredictable.

Configuration

In the case of a server joining an existing cell, you must decide whether the server is to run a replica of the global location broker.

If your machine is on a subnetwork different from the one of the server that starts the global location broker, or if your system does not support broadcasting, further configuration steps are needed after you do the basic configuration (see "Configuring to Reach a Global Location Broker in a Different Subnetwork" on page 72).

Using the Configuration Tools

To configure License Use Runtime, you can use one of the following tools:

- Configuration Tool script
- Configuration Tool command line interface

Using the Configuration Tool Script

License Use Runtime provides a configuration script that takes you through a guided configuration of your machine.

Login with root authority and issue the following command:

```
i4cfg -script
```

You are then offered the choice of four configuration scenarios:

If the machine you are configuring is the central registry license server, select **4**. During the question-and-answer session that follows, the machine will automatically be configured as a network license client, and you can optionally configure as a nodelocked license server or a network license server, or both.

If the machine you are configuring is not the central registry license server, but is a network license server, select **3**. During the question-and-answer session that follows, the machine will automatically be configured as a network license client, and you can optionally configure as a nodelocked license server.

If the machine you are configuring is a nodelocked license server and is neither the central registry license server nor a network license server, select **2**. During the question-and-answer session that follows, you can optionally configure as a network license client.

If the machine is a network license client only, select **1**.

If you run the script more than once, the latest information entered takes effect.

Using the Configuration Tool Command-Line Interface

You can use the `i4cfg` command to accomplish the same configuration tasks you can accomplish through the configuration script. The command is explained in detail in "i4cfg - Configuration Tool" on page 107. These are some examples:

Configuration

1. Configure a standalone nodelocked license server without setting up any network connections to other License Use Runtime servers.

   The nodelocked license server is to be started at system startup, all loggable events are to be logged, and the log is to be in the directory /home/baratti.

   ```
   i4cfg -a n -S a -e a -l /home/baratti
   ```

2. Configure a nodelocked license server in a network so that it is possible to use the Basic License Tool to administer licenses on other License Use Runtime servers in the network. It is also possible for remote servers to manage licenses on this nodelocked license server.

   The nodelocked license server is to be started at system startup. Events to be logged are errors; vendor added or deleted; vendor messages; product added or deleted; and server start or stop. The log is to be in the directory /home/baratti.

   With direct binding:

   ```
   i4cfg -a n -S a,n -e evmps -l /home/baratti -b
   "'network ip:thelma ip:louise'
   'nodelocked ip:louise ip:speedy' 'registry ip:thelma'"
   ```

   With namespace binding, joining an existing cell that has UUID 456b91c50000.0d.00.00.87.84.00.00.00:

   ```
   i4cfg -a n -S a,n -e evmps -l /home/baratti -n l
   -c 456b91c5    . d.  .  .87.84.  .  .
   ```

   Note that to achieve the same result as the direct binding example, *louise, speedy,* and *thelma* must join the same cell.

3. Configure a network license server.

   The network license server is to be started at system startup. Events to be logged are license checkin; errors; license granted or released; vendor added or deleted; vendor messages; and product added or deleted. The log is to be in the directory /home/baratti.

   With direct binding:

   ```
   i4cfg -a s -S a -e cegvmp -l /home/baratti -b "'network ip:thelma
   ip:louise' 'nodelocked ip:speedy' 'registry ip:thelma'" -n n
   ```

   With namespace binding, starting a new alternate cell:

   ```
   i4cfg -a s -S a -e cegvmp -l /home/baratti -b null -n g -r first
   ```

   Note that to achieve the same result as the direct binding example, *speedy* and *thelma* must join this new cell.

4. Configure a network license client.

   The network license client is to be started at system startup. All loggable events are to be logged. The log is to be in the directory /home/baratti.

Reaching a GLB in a Different Subnetwork

With direct binding:

```
i4cfg -a c -b "'network ip:thelma ip:louise'
'registry ip:thelma'" -n n
```

With namespace binding, joining an existing cell that has UUID 456b91c50000.0d.00.00.87.84.00.00.00:

```
i4cfg -a c -b null -n c -c 456b91c5    . d.  .  .87.84.  .  .
```

Note that to achieve the same result as the direct binding example, *louise* and *thelma* must join the same cell.

5 Configure the central registry license server.

The central registry license server is to be started at system startup.

With direct binding:

```
i4cfg -a r -S a -e cegvmp -l /home/baratti -b "'network ip:louise'
'nodelocked ip:speedy ip:louise' 'registry ip:thelma'" -n n
```

With namespace binding, joining an existing alternate cell and replicating the global location broker at the server lab68086:

```
i4cfg -a r -S a -e cegvmp -l /home/baratti -b null -n g
-r from:ip:lab68 86 -c 789b91c5    . d.  .  .87.84.  .  .
```

Note that to achieve the same result as the direct binding example, *louise* and *speedy* must join the same cell.

Configuring to Reach a Global Location Broker in a Different Subnetwork

If your system does not support broadcasting or if the global location broker is running on a machine in a different subnetwork, perform the following additional configuration steps on your network license servers, network license clients, and central registry license server to enable them to reach the global location broker:

1 Create a file called *glb_site.txt* in the directory:

```
/etc/lum/ncs
```

In the file, make one line for the address of each server that runs the global location broker that you want to enable this machine to reach. Each address has the following form:

ip:*host*

where *host* is the TCP/IP hostname or the ip address. In the latter case, use a leading # to indicate that the host is an address and is in the standard numeric form (for example, #192.9.8.7 or #515c.111g).

Blank lines and lines beginning with # are ignored.

This is a sample of a glb_site.txt file:

```
ip:charlie
ip:#192.9.8.7
```

2 If the machine belongs to an alternate cell, copy the file:

`/etc/lum/ncs/glb_obj.txt`

from the server running the global location broker into the `/etc/lum/ncs` directory of the machine being configured. Put the same value in the NCSCell tag of the configuration file (see Appendix A, “License Use Runtime Configuration File” on page 173).

Starting and Listing Your Subsystems

When you finish your configuration, issue the command:

```
i4cfg -start
```

to start the subsystems you have configured on your machine.

To verify that they are up and running, issue the command:

```
i4cfg -list
```

Verifying Connections to Servers

To verify that license servers are running properly, use the i4tv (test verification) tool, or use the `i4blt -ln` command to get a list of active servers (network license servers, nodelocked license servers, and the central registry license server). For more information about these commands, see Chapter 5, “License Use Runtime Commands” on page 81.

Administering License Use

The rest of this chapter consists of examples that illustrate how the administrator performs the daily activities of managing license-enabled products.

The examples assume that the administrator has configured the nodelocked license server, a network license server, and the central registry license server on the server named *louise*, and that they are all up and running. The Basic License Tool is run from the server named *louise*. The administrator is Luigi Ferretti of Infotech.

The examples use three sample license-enabled products from three fictitious IBM vendors:

SMARTJava Version 2.3, a product of the vendor IBM Software Group. SMARTJava has concurrent licenses, which the vendor delivers via a compound password. It is a customer-managed use product, and the vendor enabled it to allow the customer to exercise the hard stop/soft stop policy. Its enrollment certificate is named *smrtjava.lic*.

DataMaster Version 2.1a, a product of the vendor IBM Corporation. DataMaster is a vendor-managed use product with reservable licenses. Its enrollment certificate is named *datamst.lic*.

e-MailVision Version 1.2, a product of the vendor IBM Software Solutions. e-MailVision has per-server/per-seat licenses. The enrollment certificate for the per-server license is *emailvps.lic*, and for per-seat it is *emailvpt.lic*.

The administrator has placed the enrollment certificate files for these three products in the directory `/home/ferretti/certif/`.

Of course, when you perform the activities illustrated in the examples you must substitute your own values for variables such as server name, product name, vendor name, enrollment certificate name, product version, and user name.

Performing Basic Administration

The examples in this section demonstrate how to:

Enroll a licensed product (“Example 1: Managing a Licensed Product”).

Distribute licenses from a compound password (“Example 1: Managing a Licensed Product”).

Get a report on the use of licensed products (“Example 1: Managing a Licensed Product”).

Check the current users of licensed products (“Example 1: Managing a Licensed Product”).

Reserve reservable licenses for specific users and monitor the use of reservable licenses (“Example 2: Managing Reservable Licenses” on page 75).

Example 1: Managing a Licensed Product

In this example, the administrator enrolls and manages the SMARTJava product. This example shows you how to:

Enroll the SMARTJava product
Enroll 20 licenses for SMARTJava
Distribute five of the SMARTJava licenses to a network license server
Request a report on usage of SMARTJava licenses during a one-month period
Check the number of concurrent users of SMARTJava

To enroll the product on the central registry license server:

```
i4blt -a -n louise -f /home/ferretti/certif/smrtjava.lic -T 2
-R "'Luigi Ferretti' Infotech '73 Fifth Avenue New York'"
-I "'First installed by Luigi'"
```

If you choose to enroll a product using the command line interface, check the top of the enrollment certificate file; the vendor, while generating the password, may have generated the command to be used.

To distribute five licenses to network license server *louise*:

```
i4blt -E -n louise -v "'IBM Software Group'" -p "SMARTJava 2.3" -A 5
-w louise -I "'Luigi using root'"
```

To generate a report of requests for SMARTJava from July 2 to August 2, 1998:

```
i4blt -r2 -p "SMARTJava" -b  7/ 2/1998 -g  8/ 2/1998
```

To display information about concurrent users of SMARTJava:

```
i4blt -s -lc -p "SMARTJava"
```

Example 2: Managing Reservable Licenses

In this example, the administrator manages licenses of the DataMaster product. This example shows you how to:

- Reserve some reservable licenses for the exclusive use of a specified user
- Monitor usage of reserved licenses by the users for which they were reserved
- Monitor use of unreserved reservable licenses by other users

In this example, 100 reservable licenses for DataMaster have already been enrolled. The enrollment process is the same as for concurrent licenses of a customer-managed use product.

The enrollment certificate file for DataMaster is shown as an example in “Checking License Details” on page 153.

To get the timestamp of the licenses to be reserved:

```
i4blt -lp -i -v "'IBM Corporation'" -p "DataMaster"
```

To reserve ten DataMaster licenses for the user *Paula* **in group** *staff* **on node** *louise.rome.tivoli.com* **using the license password identified by timestamp** *899460562*:

```
i4blt -R r -v "'IBM Corporation'" -p "DataMaster 2.1a"
-t 89946 562 -A 1  -g 11/ 2/1998 -H 18:   -u "Paula staff louise.rome.tivoli.com"
```

To display information about the users of reserved licenses:

```
i4blt -s -lrr -v "'IBM Corporation'" -p "DataMaster"
```

To display information about the users of unreserved licenses:

```
i4blt -s -lru -v "'IBM Corporation'" -p "DataMaster"
```

To display detailed information about the product, including the number of reserved and unreserved licenses:

```
i4blt -lp -i -v "'IBM Corporation'" -p "DataMaster"
```

Exercising Customer-Controlled Policies

The examples in this section explain how to exercise the customer-controlled policies outlined in “Customer-Controlled Policies” on page 13. The examples show how to:

- Switch from per-server to per-seat licenses (“Example 3: Switching from Per-Server to Per-Seat Licenses”).
- Use the hard stop/soft stop policy (“Example 4: Using the Hard Stop/Soft Stop Policy”).
- Update the number of licenses of a customer-managed use product (“Example 4: Using the Hard Stop/Soft Stop Policy”).
- Control the set of users who are permitted to use a specific application (“Example 5: Restricting User Access” on page 77).

Example 3: Switching from Per-Server to Per-Seat Licenses

In this example, the administrator switches the per-server/per-seat policy for the product e-MailVision from per-server to per-seat. The per-server license has already been enrolled. The enrollment process is the same as for customer-managed concurrent licenses.

To enroll the per-seat licenses for e-MailVision:

```
i4blt -a -f /home/ferretti/certif/emailvpt.lic -T 2 -R "'Luigi Ferretti'"
```

To switch e-MailVision from per-server to per-seat licensing:

```
i4blt -U -v "'IBM Software Solutions'" -p "e-MailVision 1.2" -S yes
```

Example 4: Using the Hard Stop/Soft Stop Policy

In “Example 1: Managing a Licensed Product” on page 74, the administrator enrolled the SMARTJava product and distributed five licenses from a network compound password. Now the administrator has distributed the remaining 15 enrolled licenses. Because the vendor enabled this product to use the hard stop/soft stop policy, and the administrator is running it with soft stop set, it is possible that more than 20 licenses are being used at any given time.

In this example, the administrator:

- Checks the current number of licenses in use and the maximum number of licenses that have been granted beyond the 20 enrolled (the *high-water mark*)
- Decides to acquire ten more licenses
- Updates the number of enrolled licenses to 30
- Resets the high-water mark to 0
- Distributes ten more licenses from the compound password

To display information about usage of soft stop licenses of SMARTJava:

```
i4blt -lp -p "SMARTJava" -i
```

To update the enrollment, enrolling ten more licences on the central registry license server:

```
i4blt -U -v "'IBM Software Group'" -p "SMARTJava 2.3" -T 3  -I "'Luigi using root'"
```

To reset the high-water mark to 0:

```
i4blt -U -v "'IBM Software Group'" -p "SMARTJava 2.3" -M
```

To distribute the ten licenses to network license server *louise*:

```
i4blt -E -n louise -v "'IBM Software Group'" -p "SMARTJava 2.3" -A 1
-w louise -I "'Luigi using root'"
```

Example 5: Restricting User Access

This example explains how to create a user file to designate that certain users may or may not use certain products. You could also use a previously created user file as a base. To create a user file, follow these steps:

1. Using a text editor, open a file named *user_file*.
2. Within the file, to restrict access to a product, use the **vendor** keyword, followed by the name of the vendor, followed by either **all** (meaning all products of this vendor) or the name of a product. Enclose vendor names and product names in quotation marks if they contain embedded blanks.

   For example:

   ```
   vendor "IBM Software Group" SMARTJava
   vendor Grafix,Inc. all
   ```

   You need one **vendor** statement for each product of the same vendor, unless **all** is sufficient for your purposes.
3. After each **vendor** statement, code either an **allow** or a **disallow** statement:

   allow
   Specifies that the user names that follow this keyword are allowed to use the product. If no user names follow this keyword, no users can use the product.

   The user name is the login user name.

   For example:

   ```
   vendor "IBM Software Group" SMARTJava
   allow fritz harry monique penny
   ```

   This specifies that only four users can use the *SMARTJava* product: Fritz, Harry, Monique, and Penny.

   allow and **disallow** are mutually exclusive.

   disallow
   Specifies that the user names that follow this keyword are not allowed to use the product. If no user names follow this keyword, all users can use the product.

The user name is the login user name.

For example:

```
vendor Grafix,Inc. all
disallow heather jason
```

This specifies that all users **except** Heather and Jason can use all *Grafix,Inc* software products.

allow and **disallow** are mutually exclusive.

4. Store the file in the `/var/lum` directory of the machine where the licenses to be restricted are installed.

   In this example, the complete user file is:

   ```
   % This line is a comment
   %
   vendor "IBM Software Group" SMARTJava
   allow fritz harry monique penny
   %
   vendor Grafix,Inc. all
   disallow heather jason
   ```

5. For a consistent user authorization policy, store the same use file on all network license servers and nodelocked license servers in your environment, including the central registry license server.

6. When adding a new product, remember to update user files at all the license servers accordingly.

Administering High-Availability Licensing

The example in this section shows how to set up and manage a cluster of network license servers to ensure high availability of concurrent licenses.

Note that when you create a cluster, License Use Runtime generates the cluster ID. For a software vendor to create passwords that are bound to a cluster rather than to a single server, you must provide the cluster ID to the vendor. Therefore, you must create the cluster before you can request licenses bound to the cluster from a software vendor.

Example 6: Creating and Administering a Cluster

In this example, the administrator:

Creates a cluster consisting of four network license servers (members)
Activates all the servers in the cluster
Adds a fifth server to the cluster
Deactivates a cluster member

To create a cluster named Peanut that has members *moon, hydra, speedy,* **and** *louise***:**

```
i4blt -H c -N Peanut -T 4 -n "moon hydra speedy louise"
```

To activate the servers *hydra, speedy,* **and** *louise***:**

```
i4blt -H a -N Peanut -n hydra
i4blt -H a -N Peanut -n speedy
i4blt -H a -N Peanut -n louise
```

The server *moon,* which is the first in the list, is automatically activated.

To add the server *thelma* **to the cluster:**

```
i4blt -H a -N Peanut -n thelma
```

Note that only AIX, HP-UX, IRIX, Solaris, Windows NT (x86), Windows NT Alpha, Windows Terminal Server (x86), and Windows Terminal Server Alpha network license servers can be members of a cluster.

To deactivate the server *thelma***:**

```
i4blt -H d -N Peanut -n thelma
```

To get an overall report of cluster status:

```
i4blt -H s -N Peanut
```

To get a report of cluster status from the perspective of one of the activated servers in the cluster:

```
i4blt -H s -n moon
```

Upgrading a Custom Configuration

The scenario in this section shows you how to upgrade a custom configuration by adding a product to a current custom configuration.

To ensure that the products used are up to date, functionally suitable, and competitive, you will occasionally need to add new product components, increase the number of licenses, or extend the license period. To do this, you request from the vendor a new custom configuration password and supply the serial number of the current license. This serial number identifies your current custom configuration. You then pay for only the difference between the cost of the current license and that of the new license. Next, you install the upgraded license as shown in the following procedure.

Upgrading a Custom Configuration

To upgrade a custom configuration license, using the license certificate file `m2update.lic`**, on server** *louise*:

```
i4blt -a -f m2update.lic -n louise
```

where:

`m2update.lic` Is the name of the file that contains the upgraded license certificate.

`louise` Is the name of the server on which the initial key is installed.

Chapter 5. License Use Runtime Commands

This chapter describes how to use the License Use Runtime command line interface.

In the .HTM version of this Command Reference, changes made since Version 4.0 are shown in purple.

In the descriptions of command syntax, the following conventions are used:

Code items shown in **bold** type exactly as shown.
Replace items shown in *italic* type with your own values.
Parameters shown in brackets ([ ]) are optional.
Choose one from a list of parameters shown in braces ({ }).

The following commands are available:

i4blt	Basic License Tool
i4cfg	Configuration Tool
lb_admin	Local Broker Administration
drm_admin	GLBD Replicas Administration
lb_find	GLBs List
uuid_gen	ID Generator
i4tv	Test Verification Tool
i4target	Target View Tool
llbd	Local Location Broker Subsystem
glbd	Global Location Broker Subsystem
i4lmd	Network License Server Subsystem
i4llmd	Nodelocked License Server Subsystem
i4gdb	Central Registry Subsystem
i4glbcd	Global Location Broker Data Cleaner Subsystem
i4lct	License Creation Tool

i4blt - Basic License Tool

General Rules for the i4blt Command

1. The parameters within any of the following name specifications are positional:

   vendor_information (*vendor_name vendor_id vendor_password*)

   product_information (*product_name product_version license_password license_annotation*)

   administrator_information (*administrator_name company_name address additional_info*)

   user_information (*user_id user_group user_node*)

2. All the following name specifications must be enclosed within double quotation marks (for example: `"vendor_name vendor_id vendor_password"`).

   vendor_information (*vendor_name vendor_id vendor_password*)

   product_information (*product_name product_version license_password license_annotation*)

   administrator_information (*administrator_name company_name address additional_info*)

   user_information (*user_id user_group user_node*)

   signature_information

3. When a list of values (such as server names, vendor names, product names, or user names) is entered as a parameter, the list must be enclosed in double quotation marks. For example:

   ```
   i4blt -r3 -u "katie dustin emily adam"
   ```

4. A name that contains character spaces must additionally be enclosed within single quotation marks. If multiple blanks within the name must be preserved, each must be preceded by a backslash. For example:

   ```
   -v "'IBM Corporation'"
   -p "'Corel\ \ - System' 1.1"
   ```

5. The parameters you specify in any of the command options (for example, server names, vendor names, and product names) are case-sensitive.

6. You can display help on i4blt command options as follows:

   To get help on just the -a, -U, -E, -d, or -m option:

   i4blt -*option*

   To get help on just the -R, -l, -r, x, or H option:

   i4blt -*option***h**

Examples

Display the i4blt -E syntax:

```
i4blt -E
```

Display the i4blt -r syntax:

```
i4blt -rh
```

Primary Command Options

The i4blt command has the following primary command options:

-a (Enroll a Product)
Add products to a license database

-U (Update a Product)
Update the number of licenses you enrolled, update the hard stop/soft stop policy and high-water mark when enabled on the product, switch from per-server to per-seat licenses, and set the threshold value of a customer-managed product.

-E (Extract and Distribute Licenses)
Extract and distribute licenses from a network compound password of a given product to the servers.

-d (Delete a Product License or an Application Client Identifier)
Delete products from a license database, or Application Client Identifiers from the Central Registry of Application Clients.

-R (Reserve Licenses; Delete or Update Reserved Licenses)
Reserve reservable licenses for use by a specific user, group, or node; deletes reserved licenses; updates reservation status.

-C (Clean Up Stale Licenses)
Update the number of concurrent, reservable, per-server, and concurrent nodelocked licenses in use.

-l (Display a List)
List license database information about servers, vendors, products, and licenses.

-s (Display Product License Status)
Gather status information about product license usage.

-r (Generate a Report)
Report on information recorded in the log file of a license server.

-x (Delete Log Entries)
Delete license server and central registry log file entries.

-m (Monitor and Log Threshold Events)
Monitor and log the threshold messages.

-H (Administer High-Availability Licensing)
Create a cluster of network license servers; add servers to an existing cluster; display cluster status; activate and deactivate servers in a cluster.

i4blt - Basic License Tool

-h (Display Help)
Display command syntax and usage information about the Basic License Tool command-line interface.

-a Enroll a Product

This option adds a product to the license database on the license server that you specify. Use the **i4blt -a** command to add a new product and its initial licenses to a license server database. You can also use this command to add licenses for existing vendor-managed products.

You can add product license information to a server in two ways:

If you got the product license information in the form of an enrollment certificate file, you can install the product importing the enrollment certificate.

If you got the product license information in a format other than an enrollment certificate file, you must enter the product information manually.

Syntax

If you have the enrollment certificate file:

i4blt -a
[**-n** *server_name*]
-f *filename*
[**-R** *administrator_name* [*company_name address additional_info*]]
[**-T** *enrolled_licenses*]
[**-I** *signature_information*]

If you do not have the enrollment certificate file:

i4blt -a
[**-n** *server_name*]
-v *vendor_name vendor_id vendor_password*
-p *product_name product_version license_password* [*license_annotation*]
[**-R** *administrator_name* [*company_name address additional_info*]]
[**-T** *enrolled_licenses*]
[**-I** *signature_information*]
[**-S** *serial_number*]

Parameters

-n *server_name*
Specifies the name of the license server to which you intend to add the product.

If -n is omitted:

If the product is customer-managed, and the licenses are network licenses, they are enrolled on the central registry.

If the product is customer-managed, and the license is nodelocked, it is enrolled on the local machine.

If the product is vendor-managed, and the enrollment certificate file specifies a target ID, the licenses are enrolled on that machine.

If the product is vendor-managed, and the enrollment certificate file does not specify a target ID, the licenses are enrolled on the local machine.

-f *filename* (Only if you have the enrollment certificate)
The complete path and file name of the enrollment certificate file containing the product license information that you intend to add.

-v *vendor_name* (If you do not have the enrollment certificate)
The name of the vendor that manufactured the product that you intend to add.

vendor_id
The unique vendor ID string for the vendor that you specify in the ***vendor_name***.

vendor_password
The unique vendor password string for the vendor that you specify in the ***vendor_name*** argument.

-p *product_info* (If you do not have the enrollment certificate)
The information on the licensed product that you intend to install.

product_name
The name of the product that you want to install.

product_version
The version of the product that you specified in the *product_name* parameter.

license_password
The unique license password string associated with the product.

license_annotation
The license annotation information (if any) provided by the vendor.

-R *administrator_info* (for customer-managed use products only)
The information on the administrator who enrolls the product.

administrator_name
The name of the administrator who performs the operation. This parameter is required.

company_name
The name of your company.

address
The address of your company.

additional_info
Comments, notices to future users, or other information about the initial enrollment of this product.

-T ***enrolled_licenses*** **(for customer-managed use products only)**
The number of licenses you have acquired from the software supplier. This parameter is required.

-I ***signature_information*** **(for customer-managed use products only)**
Information about the user issuing the command, to be stored with the signature stamp.

-S ***serial_number***
The serial number of a custom configuration license. The serial number is a string of up to 31 alphanumeric characters that uniquely identifies a custom configuration.

Examples

Add a customer-managed use product:

```
i4blt -a
-v "Venus 4ca fd5cf   . d.  . 2.1a.9a.  .  .   kz5esmu69hzyw"
-p "timer 1.1 wzx3ewdfrvu4v64d53bbrkzhheaaaaa"
-R "Alex IBM Rome" -T 1
-I "'Alex using root userid'"
```

Add a vendor-managed use product:

```
i4blt -a -n thelma
-p "scena 1.  suf fpeixfi5v78a22xxrkzhheaaaaa"
-v "Operatix 7gp4ac8jj   . d.  . 2.1a.9a.  .  .   lb7usud93jdna"
```

-U Update a Product

This option is valid only for customer-managed use products.

It is issued for the following purposes:

To update the number of licenses you enrolled. Use it when you acquire new licenses for an already enrolled customer-managed use product, to update the total number of licenses you are entitled to use. In the case of a network compound password, the licenses must be distributed after the update to make them available to end users.

For a product with per-server/per-seat licenses, to switch from per-server to per-seat licenses.

For a product with the hard stop/soft stop policy enabled, to change the hard stop/soft stop policy and to reset the high-water mark.

To update the threshold value of a product.

Syntax

i4blt -U
-v *vendor_name*
-p *product_name product_version*
[**-n** *server_name*]
[**-T** *enrolled_licenses*]
[**-S** *enable_switch* [**yes**]]

[**-H** *hard_soft_mode* [**yes** | **no**]]
[**-M** *hwm_reset*]
[**-t** *threshold* [**1...100**]]
[**-I** *signature_information*]

Parameters

-v *vendor_name*
The name of the vendor that manufactured the product that you intend to update.

-p *product_info*
The information on the licensed product that you intend to update.

product_name
The name of the product for which you have acquired the new licenses.

product_version
The version of the product that you specified in the *product_name* parameter.

-n *server_name*
Name of the license server on which you want to update product information.

This parameter is required if the product has nodelocked licenses and you are updating the product on a remote nodelocked license server. It is the name of the nodelocked license server. If you are updating the product on the local nodelocked license server, omit the **-n** parameter. If the product has network licenses, this parameter need not be specified, because the server is the central registry license server.

-T *enrolled_licenses*
The total number of licenses you have for the specified product; that is, the number of licenses you had, plus the new ones.

-S *enable_switch*
Use this parameter to migrate the license from per-server to per-seat. To use the per-seat license remember also to enroll the per-seat certificate.

The only allowed value for **-S** is **yes**. When the licenses have been changed to per-seat, you cannot go back to per-server licenses.

-H *hard_soft_mode*
Use this parameter to switch the product behavior from hard stop to soft stop and vice versa. You can do it only on products the vendor has enabled to allow hard stop/soft stop switching.

Allowed values for **-H** are:

no Set the soft stop
yes Set the hard stop

-M *hwm_reset*
Use this parameter to reset the high-water mark to 0. You can do it only on products the vendor has enabled to soft stop.

-t *threshold*
Use this parameter to set a specific value for the threshold value of a customer-managed product. Allowed values are 1 to 100.

-I *signature_information*
Information about the user issuing the command, to be stored with the signature stamp. Use this parameter along with the **-T** parameter.

Examples

Update the number of licenses for the **Test Compiler** product, Version **1.1** of vendor **Psychosync** to **50**. The product has network licenses.

```
i4blt -U -v "Psychosync" -p "'Test Compiler' 1.1"
-T 5  -I "'Paula using root userid'"
```

Set the soft stop policy and reset the high-water mark of the **Test Compiler** product, Version **1.1** of vendor **Psychosync**. The product has network licenses.

```
i4blt -U -v "Psychosync" -p "'Test Compiler' 1.1" -H no -M
```

Update to **5** the number of nodelocked licenses for the **ScreenPic** product, Version **2** of vendor **ArtTools** on nodelocked license server **Virginia**:

```
i4blt -U -n Virginia -v "ArtTools" -p "ScreenPic 2" -T 5
```

-E Extract and Distribute Licenses

Use the **i4blt -E** command to extract licenses from an installed network compound password and distribute them to the network license servers.

Syntax

i4blt -E
-n *origin_server_name*
-v *vendor_name*
-p *product_name product_version*
-A *license_number_per_server*
-w *target_server_names*
[**-I** *signature_information*]

Parameters

-n *origin_server_name*
The name of the server where the network compound password is enrolled.

-v *vendor_name*
The name of the vendor that manufactured the product whose licenses you want to distribute.

-p *product_info*
The information on the licensed product whose licenses you intend to distribute.

product_name
The name of the product whose licenses you want to distribute.

product_version
The version of the product that you specified in the *product_name* parameter.

-A *license_number_per_server*
The number of licenses for the specified product you want to distribute on each of the servers specified after the -w parameter.

-w *target_server_names*
The servers on which you want to distribute the licenses.

-I *signature_information* (For customer-managed use products only)
Information about the user issuing the command, to be stored with the signature stamp.

Examples

Extract and distribute **10** licenses to each of the servers **Louise** and **Hall**, for the **Test Compiler** product, Version **1.1** of vendor **Psychosync**, installed on server **Thelma**:

```
i4blt -E -n "Thelma" -v "Psychosync" -p "'Test Compiler' 1.1"
-A 1  -w "Louise Hall" -I "'Paula using root userid'"
```

-d Delete a Product License

This option deletes a product license from the license database on the license server that you specify, or an Application Client Identifier from the central registry.

Syntax

i4blt -d
-n *server_name*
-v *vendor_name*
-p *product_name product_version*
{ **-t** *timestamp* | **-A** *ACID* }
[**-I** *signature_information*]

Parameters

-n *server_name*
| Either of the following:

| The name of the license server from which you intend to delete the product
| license.
| To delete a high-availability license, the name of one of the servers in the
| cluster on which the license is enrolled.

To delete a high-availability license, issue the command:

```
i4blt -d -n server_name -v vendor_name -p product_name product_version -t timestamp
```

where `server_name` identifies one of the servers in the cluster on which the license is enrolled.

-v *vendor_name*
Name of the vendor whose product license you intend to delete.

-p *product_info*
The information on the licensed product whose licenses you intend to delete.

product_name
Name of the product whose license you intend to delete.

product_version
Version of the product whose license you intend to delete.

-t *timestamp*
Unique timestamp of the product license that you intend to delete. To get the timestamp, issue the following command:

```
i4blt -lp -p "product_info" -i
```

Do not specify the timestamp when you delete an Application Client Identifier.

-A *ACID*
Unique identifier of the Application Client Identifier of an application client you want to delete from the central registry. After deletion the application client no longer has the license to use the specified product. To get the Application Client Identifier, issue the following command:

```
i4blt -s -lpt -v "vendor_name" -p "product_info"
```

Do not specify **-A** when you delete a product.

-l *signature_information* (for customer-managed use products only)
Information about the user issuing the command, to be stored with the signature stamp. Use this parameter when deleting a product license.

When the last license for the only remaining product of a vendor is deleted, the vendor is automatically deleted from the license database. Vendor-managed compound passwords and use-once licenses cannot be deleted until they expire.

Examples

Delete an expired license to use a **VectorComp Corporation** product called **EZ-Vectors** Version **1.0**. The unique timestamp of the license to be deleted from the database on server **saturn** is **781401788**:

```
i4blt -d -n saturn -v "'VectorComp Corporation'" -p "EZ-Vectors 1. " -t 7814 1788
```

Delete an application client whose Application Client Identifier is **thelma** from the central registry. After this command the application client will no longer have licenses for the product **EZ-Vectors** Version **1.0** of **VectorComp Corporation** vendor:

```
i4blt -d -v "'VectorComp Corporation'" -p "EZ-Vectors 1. " -A thelma
```

-R Reserve Licenses; Delete or Update Reserved Licenses

Use **i4blt -R** to reserve reservable licenses and to delete or update the reservation status of reserved licenses.

Syntax

i4blt -R *action_type* [**r | d | u**]
-n *server_name*
-v *vendor_name*
-p *product_info*
[**-t** *timestamp*]
[**-A** *license_number*]
[**-g** *end_date*]
[**-H** *end_time*]
[**-u** *user_id user_group user_node*]

Parameters

action_type
To reserve licenses, **r**; to delete licenses, **d**; to update an existing reservation, **u**.

-n *server_name*
The name of the server where the product license is enrolled.

-v *vendor_name*
The name of the vendor that manufactured the product.

-p *product_info*
The information on the licensed product whose licenses you intend to reserve, delete, or update.

product_name
The name of the product.

product_version
The version of the product.

-t *timestamp*
Unique timestamp of the product license from which you intend to reserve, or that you intend to delete or update. To get the timestamp, issue the following command:

```
i4blt -lp -p "product_info" -i
```

If you are reserving licenses (option **-R r**), the timestamp is optional. If it is omitted, the first usable reservable license is used.

-A *license_number*
The number of licenses you intend to reserve. If you are updating a reservation (**-R -u**) or deleting licenses (**-R -d**), do not specify **-A**.

-g ***end date***
The end date of the new or updated reservation (*mm/dd/yyyy*). The latest allowed expiration date of a reservation is 12/31/2037. If you are deleting licenses (**-R -d**), do not specify **-g**.

-H ***end time***
The end time of the new or updated reservation (*hh:mm*). If you are deleting licenses (**-R -d**), do not specify **-H**.

-u ***user_id user_group user_node***
The identification of the user, group, and node for which a license is being reserved or a reservation is being changed. Any of these values may be *, meaning “any”. If you are deleting licenses (**-R -d**), do not specify **-u**.

Examples

Reserve three licenses for **Test Compiler** product, taken from the reservable license identified by the timestamp **389588975**, Version **1.1** of vendor **Psychosync** for any member of the **testers** group. They expire March 2, 1998, at 11:00.

```
i4blt -R r -v "Psychosync" -p "'Test Compiler' 1.1"
-t 389588975 -A 3 -g  3/ 2/1998 -H 11:   -u "  testers  "
```

-C Clean Up Stale Licenses

Use **i4blt -C** to update the number of in-use concurrent, reservable, per-server, and concurrent nodelocked licenses.

When you issue this command, License Use Runtime polls all the license servers that have granted licenses of these types and verifies that the licenses are still in use. If any stale licenses are found, they are removed from the number of in use licenses.

Syntax

i4blt -C
[**-F** *server_type* { **l** | **w** | **a** }]
[**-n** *server_names*]
[**-v** *vendor_names*]
[**-p** *product_names*]

Parameters

-F ***server_type***
A filter on the type of server to be searched. Specify **l** for nodelocked license servers, **w** for network license servers, or **a** (the default) for both network license servers and nodelocked license servers.

-n ***server_names***
The names of the servers where the products are enrolled.

-v ***vendor_names***
The name of the vendors that manufactured the products whose licenses are in use.

-p *product_names*
The names of the products whose stale licenses you want to clean up.

Examples

Clean up stale licenses for the **Graphics** product of vendor **Alpha** on servers **Thelma**, **Hall**, and **Louise**:

```
i4blt -C -n "Thelma Hall Louise" -v "Alpha" -p "Graphics"
```

Clean up stale licenses for the **Graphics** product of vendor **Alpha** on all nodelocked license servers in the network.

```
i4blt -C -F l -v "Alpha" -p "Graphics"
```

-l Display a List

You can use this option to display a list of servers, vendors, products, or licenses. You can also use it to display details about individual products or individual licenses.

Syntax

i4blt -l *list_type* [{ **n** | **s** } | **v** | **p** [**-i**] | **l**]
[**-F** *server_type* { **l** | **w** | **a** }]
[**-n** *server_names*]
[**-v** *vendor_names*]
[**-p** { *product_name*[**?***product_version*] } ...]
[**-u** *user_names*]
[**-t** *timestamp*]

Parameters

list_type Indicates the type of information that you want to list.

You can specify any one of the following list types:

-ln or -ls
To display a list of active license servers.

Filters:

You can use the **-F** filter option to display a list of active network license servers or nodelocked license servers.

Do not specify the **-n, -v, -p,** or **-u** filter option together with this parameter.

-lv
To create a vendor list.

Filters:

To list vendor information gathered from a specific type of license server, use the **-F** filter option to specify nodelocked license servers or network license servers.

To list vendor information gathered from servers that you specify, use the **-n** filter option followed by one or more server names.

Do not specify the **-v, -p,** or **-u** filter option together with this parameter.

-lp [-i]

To create a product list.

Filters:

To list product information gathered from a specific type of license server, use the **-F** filter option to specify nodelocked license servers or network license servers.

To list product information gathered from servers that you specify, use the **-n** filter option followed by one or more server names.

To list product information on products from particular vendors, use the **-v** filter option, followed by one or more vendor names.

To list product information on particular products, use the **-p** filter option, followed by one or more product names.

To list information on users who are currently using the products that you specify, use the **-u** filter option, followed by one or more user names.

Specify the **-i** option to display detailed information about each product in a product list.

-ll

To create a list of individual licenses. The output includes all the information you get by specifying **lp** with the **-i** option, plus, for products with concurrent licenses that are administered in a high-availability environment, information about the cluster and servers within the cluster.

Filters:

To list license information gathered from a specific type of license server, use the **-F** filter option to specify nodelocked license servers or network license servers.

To list license information gathered from servers that you specify, use the **-n** filter option followed by one or more server names.

To list license information on products from particular vendors, use the **-v** filter option, followed by one or more vendor names.

To list license information on particular products, use the **-p** filter option, followed by one or more product names.

To list information on users who are currently using the licenses that you specify, use the **-u** filter option, followed by one or more user names.

To list information on a specific license, use the **-t** filter option, followed by the timestamp of the license.

High-Availability Output:

Cluster name

For each server in the cluster:

- Server name
- Server status:

  Serving Running, serving licenses

  Waiting Server is ready, but the cluster is in incomplete or inactive state

  Unavailable Not started

  Reserve In reserve in case a serving server becomes unavailable

  Not activated Defined as a member of the cluster but the administrator has not yet activated the server or has deactivated the server
- Percentage of licenses being served by this server
- Target ID
- Number of licenses served by this server
- Number of in-use licenses served by this server

-F *server_type*

A filter on the type of server to be searched. Specify **l** for nodelocked license servers, **w** for network license servers, or **a** (the default) for both network license servers and nodelocked license servers.

-n *server_names*

Names of the servers about which you want to display information in a vendor or product list.

-v *vendor_names*

Names of the vendors about whose products you want to display information in a product list.

-p { *product_name*[?*product_version*] } ...

Names of the products and, optionally, theiT gTj /F5 1 Tf 4.T]332Fdi-2F4 10.778 0 Tcense servers and no

-i Specify the **-i** option in conjunction with a product list (`i4blt -lp`) to display the following detailed license usage information about an individual licensed product in a product list:

Vendor name
Vendor ID
Product name
Product version
Product ID
Licenses (total on all the selected servers)
In-use licenses (total on all the selected servers)

For each license instance:
- Number of licenses
- License type
- Server on which the license is installed
- License annotation (if any)
- Serial number (if any)
- Start date
- Expiration date
- Time stamp
- Password use control level

For products with customer-managed use control and per-server, per-seat, or concurrent nodelocked licenses, the following information is also displayed:
- High-water mark licenses
- Threshold value
- Soft stop
- Soft stop enabled

For products with customer-managed use control and use-once nodelocked licenses, the following information is also displayed:
- Threshold

For per-seat licenses, the following information is also displayed:
- Enablement flag

For reservable licenses, the following information is also displayed:
- Number of reserved licenses
- Number of unreserved licenses

For reserved licenses, the following information is also displayed:
- User for whom licenses are reserved
- Group for which licenses are reserved
- Node for which licenses are reserved

For concurrent, concurrent nodelocked, and per-server licenses, the following information is also displayed:
- Multiuse rules (if any)

For try-and-buy licenses, the following information is also displayed:
- Try-and-buy flag

For compound passwords, the following information is also displayed:
- Derived license type
- Aggregate duration
- Derived start type
- Derived expiration date
- ***For products with customer-managed use control and concurrent or reservable licenses,*** the following information is also displayed:
  - Enrolled licenses
  - Distributed licenses
  - To be distributed licenses
  - High-water mark licenses
  - Threshold value
  - Soft stop
  - Soft stop enabled
- ***For products with customer-managed use control and use-once licenses,*** the following information is also displayed:
  - Enrolled licenses
  - Distributed licenses
  - To be distributed licenses
  - Threshold value

Examples

List all servers:

```
i4blt -ln
```

List all vendors on all servers:

```
i4blt -lv
```

List all vendors on all network license servers:

```
i4blt -lv -F w
```

List all vendors on server **Hall**:

```
i4blt -lv -n Hall
```

List all products on server **Hall**:

```
i4blt -lp -n Hall
```

List all products on server **mercury** provided by vendors **Opticon, Inc.** and **Cybertronics Ltd.**:

```
i4blt -lp -n mercury -v "'Opticon, Inc.' 'Cybertronics Ltd.'"
```

List detailed information for the product **PsychoSynch** on server **venus**:

```
i4blt -lp -n venus -p PsychoSynch -i
```

i4blt - Basic License Tool

List all of the products on the server **neptune** that are currently being used by the user **Alex**:

```
i4blt -lp -n "neptune" -u Alex
```

-s Display Product License Status

This option displays information about current product usage on the license servers that you specify.

Syntax

i4blt -s
[**-l** *list_type* [**c** | **pt** | **ps** | **ru** | **rr** | **cn**]]
[**-n** *server_names*]
[**-v** *vendor_names*]
[**-p** { *product_name*[**?***product_version*] } ...]
[**-u** *user_names*]

Parameters

-l *list_type*

Indicates the type of license usage you want to list.

You can specify one of the following list types:

-c To display information related to concurrent users of concurrent licenses.

-pt To display information related to application clients that use per-seat licenses.

-ps To display information related to users of per-server licenses.

-ru To display information related to users of unreserved reservable licenses.

-rr To display information related to users of reserved licenses.

-cn To display information related to users of concurrent nodelocked licenses.

If you omit **-l**, its default value is **c**.

-n *server_names*

The name of each of the license servers for which you want to display product usage information.

If you omit the **-n** parameter, the display defaults to all servers in your cell. This parameter is not used if you use **pt, ps, rr,** or **cn** as the list type.

-v *vendor_names*

The name of the vendor (or vendors) about whose products you want to display information.

-p { *product_name*[*?product_version*] } ...

Names of the products and, optionally, their versions about which you want to display information.

-u *user_names*
Use the optional **-u *user_names*** argument to display product usage information for the specified products that are currently in use by the named users.

This command displays the following information for the servers, vendors, products, and users that you specify:

Vendor name
Product name
Product version
Total number of installed licenses
Number of licenses currently in use
Number of soft stop licenses currently in use
Number of licenses not in use
Number of queued users

For each user who currently holds a license, the following information is displayed:

User name
Node name
Group name
Number of licenses the user has been granted
Check-out date for each granted license

Examples

Display current license availability and usage information for concurrent licenses of the **Monolith Inc.** product **Megamail/2** on server **uranus**:

```
i4blt -s -lc -n "uranus" -v "'Monolith Inc.'" -p "Megamail/2"
```

-r Generate a Report

This option lists server, event, vendor, product, and user information on the license servers that you specify.

Syntax

i4blt -r *report_type* [**1** | **2** | **3** | **4** | **5** | **6**]
[**-b** *start_date*]
[**-g** *end_date*]
[**-e** *event_filter*]
[**-F** *server_type* { **l** | **w** | **a** }]
[**-n** *server_names*]
[**-v** *vendor_names*]
[**-p** { *product_name*[**?***product_version*] } ...]
[**-u** *user_names*]

Parameters

-r ***report_type***

Specifies the type of report to generate. The following report types are available:

1 - Standard Event Report. Displays detailed information about significant events occurring on the license servers that you specify. Available for all license types.

2 - License Request by Product Report. Displays statistical information about the use of the licenses of a product in the time interval you specify. For each product, it reports the licenses requested, the licenses granted, and the percentage of rejections. Not available for simple nodelocked or use-once (nodelocked or network) licenses.

3 - License Request by User Report. Displays statistical information about the use of products by users in the time interval you specify. For each user, it reports the licenses requested, the licenses granted, and the percentage of rejections for each product the person is using. Not available for simple nodelocked or use-once (nodelocked or network) licenses.

4 - License Use by Product Report. Displays statistical information about the use of the licenses of a product in the time interval you specify. For each product, it lists the maximum number of concurrent nodes that used the product, the maximum number of concurrent users, and the average time of use of the product. Not available for per-seat, simple nodelocked, or use-once (nodelocked or network) licenses.

5 - License Use by User Report. Displays statistical information about the use of the licenses of a product in the time interval you specify. For each user, it lists the number of times each product was invoked, and the average time the user used each product. Not available for per-seat, simple nodelocked, or use-once (nodelocked or network) licenses.

6 - Customer-Managed Use Audit. Reports the following information for customer-managed use product transactions:

Vendor name
Product name
Product version
Administrator information
Time stamp of the event
Number of licenses involved in the transaction
Event list (product enrolled, license distributed, license deleted, license updated, per-server/per-seat license migrated)
Signature stamp (user, group, and node)
Signature information

Available for all license types.

-b ***start_date***

Specifies the start date of a report. Be sure to express the date using the ***mm/dd/yyyy*** format. If you specify a start date and do *not* specify an end date, the report will include all information logged from the specified start date until the present.

-g ***end_date***
Specifies the end date of a report. Be sure to express the date using the ***mm/dd/yyyy*** format. If you specify an end date and do *not* specify a start date, the report will include all information logged prior to (and including) the specified end date.

-e ***event_filter***
You can use the *event_filter* argument to generate a **Standard Event** report on the following types of events which you specify, by number, on the command line. (Separate multiple event type arguments with a comma.)

1 All events
2 License-related events
3 Vendor messages
4 License database modifications
5 Error events
6 Server start and stop events
7 Fatal errors

-F ***server_type***
A filter on the type of server to be searched. Specify **l** for nodelocked license servers, **w** for network license servers, or **a** (the default) for both network license servers and nodelocked license servers.

-n ***server_names***
Names of the servers about which you want to display information.

-v ***vendor_names***
Names of the vendors about whose products you want to display information.

-p **{** ***product_name*****[?*****product_version*** **] } ...**
Names of the products and, optionally, their versions about which you want to display information.

-u ***user_names***
Names of users about whom you want to display license usage information.

Examples

Standard Event Report:

Report on license-related events (2) and server start and stop events (6) that were logged on server **neptune** since May 21, 1998:

```
i4blt -r1 -n "neptune" -b  5/21/1998 -e 2,6
```

License Use by Product Report:

Report current license usage information on server **saturn** for the products **NetLS Test Product**, **Compiler**, **PsychoSynch**, **Megamail/2**, **EZ-Vectors**, and **DataVision**:

```
i4blt -r4 -n "saturn" -p "'NetLS Test Product' 'Compiler'
PsychoSynch Megamail/2 EZ-Vectors DataVision"
```

License Request by User Report:

Report current license usage information on server **mercury** for users **alex**, **ann**, **mary**, **christine**, **paul**, and **alby**:

```
i4blt -r3 -n "mercury" -u "alex ann mary christine paul alby"
```

Customer-Managed Use Audit Report:

Report information about customer-managed use product transactions on all nodelocked license servers from May 1, 1999 to July 31, 1999:

```
i4blt -r6 -F l -b  5/ 1/1999 -g  7/31/1999
```

-x Delete Server Log Entries

This option deletes all entries before a specified delete date from the log file of the license servers that you specify. If one of the specified license servers has the central registry, the central registry log entries are also deleted. If the specified license server is the local node, the nodelocked license server log entries are also deleted.

Syntax

i4blt -x *delete_date*
[**-F** *server_type* { **l** | **w** | **a** }]
[**-n** *server_names*]

Parameters

-x *delete_date*
Specifies an end date for the delete operation. All log entries recorded before the delete date are removed from the log file. You must specify a delete date in the ***mm/dd/yyyy*** format. If you do not specify a ***delete_date***, all entries in the log file are deleted.

-F *server_type*
A filter on the type of server to be searched. Specify **l** for nodelocked license servers, **w** for network license servers, or **a** (the default) for both network license servers and nodelocked license servers.

-n *server_names*
Specifies the license servers from whose log file you want to delete the entries.

Examples

Delete all the log file entries recorded on server **neptune** before August 25, 1998:

```
i4blt -x  8/25/1998 -n neptune
```

Delete all the log file entries recorded on all nodelocked license servers before August 25, 1998:

```
i4blt -x  8/25/1998 -F l
```

-m Monitor and Log Threshold Events

This option displays the threshold messages and logs them if the threshold logging option is specified.

Syntax

i4blt -m
[**-T** *percentage* [**1...100**]]
[**-A** *periodic_mode* [**yes** | **no**]]
[**-X** *frequency* [**1...1440**]]
[**-l** *log* [**yes** | **no**]]

Parameters

-T ***percentage***

Specifies the level of threshold value. It can be any number between 1 and 100. The default is 80.

This is the percentage over which you want to log the level of usage of each product installed on all the servers.

For instance, if you have 100 licenses of the product *Icon Editor* Version 1.5, and you set the level of threshold to 10, a message appears in the vendor messages report only if more than 10 licenses are in use.

If 20 licenses are being used, the message will say:

```
The 2 % of licenses of Icon Editor 1.5 is in use.
```

Note that a customer-managed product may have its own threshold value, set with the i4blt -U command. Such a threshold value overrides the **-T** value.

-A ***periodic_mode***

Specifies whether to check the license usage of the products only once, or periodically. Its values can be:

no To check the threshold conditions on the products once, immediately. If **-A** is omitted, this is the default.

yes To check the threshold conditions on the products periodically, with the frequency specified with the **-X** parameter.

-X ***frequency***
Specifies the number of minutes between one license usage check and the next. Enter a value between 1 and 1440. It is mandatory if you set the **A** parameter to **yes**.

-l ***log***
Specifies whether or not the threshold messages must be logged on the license server to be reviewed with the report function.

Examples

Set the threshold percentage to 50% and set the check on the products' usage to every 4 hours:

```
i4blt -m -T 5  -A yes -X 24
```

-H Administer High-Availability Licensing

This option creates a cluster of network license servers; adds servers to an existing cluster; displays cluster status; and activates and deactivates servers in a cluster.

Syntax

i4blt -H *action_type* { **c** | **a** | **d** | **s** }
[**-N** *cluster_name*]
[**-T** *initial_number_of_servers*]
[**-n** *server_names*]

Parameters

action_type
Specifies the action to be taken:

c To create a cluster. With action type **c,** the **-N, -T,** and **-n** parameters are all required.

a To add a server to a cluster, or to activate a server in a cluster. With action type **a,** the **-N** and **-n** parameters are required.

d To deactivate a server in a cluster. With action type **d,** the **-N** and **-n** parameters are required.

s To request cluster status. With action type **s,** the **-N** parameter is recommended to give you an overall view of cluster status. If you are having problems with cluster operation and you want to see a view of the cluster from the perspective of an individual server, use action type **s** with the **-n** parameter.

In either case, the cluster status display includes the following information about the cluster:

Cluster name

Cluster ID (available only after the cluster switches to **Active** status for the first time)

Cluster status:

Active	Running, serving licenses
Change Pending	Waiting for a change in the status of a server, or of the cluster, to be propagated to all the servers
Inactive	Not enough servers up and running
Incomplete	Not enough servers activated

Initial number of servers

Minimum number of members, maximum number of members, and minimum up and running for the cluster to work

For each server in the cluster:

- Server name
- Server status:

Serving	Running, serving licenses
Waiting	Server is ready, but cluster is in incomplete or inactive state
Unavailable	Not started
Reserve	In reserve in case a serving server becomes unavailable
Not activated	Defined as a member of the cluster but administrator has not yet activated the server or has deactivated the server

- Percentage of licenses being served
- Target ID

-N *cluster_name*
The name of the cluster to which the command is directed. The **-N** parameter is required if *action_type* is **c, a,** or **d**, and it is recommended when *action_type* is **s**.

-T *initial_number_of_servers*
The initial number of servers in the cluster that you are creating. The **-T** parameter is required if *action_type* is **c**, and is not valid if *action_type* is not **c**. The **-n** parameter must specify a number of servers equal to the value of **-T**.

-n *server_names*
The names of the servers to which the command is directed. The **-n** parameter is required if *action_type* is **c, a,** or **d**, and it can be used when *action_type* is **s**.

A cluster cannot contain an OS/2, Windows 95, or Windows 98 machine.

If *action_type* is **c**, this is the list of initial members of the cluster. You must specify a number of servers equal to the value of **-T**. After this command has been processed, the first server in the list is automatically activated. Issue `i4blt -H` again, using *action_type* **a**, to activate each additional server.

At the time the `i4blt -H c` command is processed, all the servers specified must be up and running. If not, the command fails.

If *action_type* is **a** or **d**, this is the name of the server to be activated or deactivated. You must specify exactly one server. If *action_type* is **a**, the server must be up and running when the command is processed. If not, the command fails.

If *action_type* is **s**, this is the name of any server that is currently activated in the cluster. The command returns cluster status from the perspective of this server.

Note: When the cluster is in **Change pending** status, different servers may return different data.

Examples

Create a cluster named ruth that has three members: anthony, germaine, and costanza:

```
i4blt -H c -N ruth -T 3 -n "anthony germaine costanza"
```

Activate the servers germaine and costanza:

```
i4blt -H a -N ruth -n germaine
i4blt -H a -N ruth -n costanza
```

Add the server sandra to the cluster:

```
i4blt -H a -N ruth -n sandra
```

Deactivate the server germaine:

```
i4blt -H d -N ruth -n germaine
```

Get an overall report of the status of the cluster nobel:

```
i4blt -H s -N nobel
```

Get a report of the status of the cluster nobel from the perspective of one of its activated members, pirandello (recommended only for troubleshooting purposes):

```
i4blt -H s -n pirandello
```

-h Display Help

This option displays general syntax information for the Basic License Tool command line interface.

Syntax

i4blt -h

Example

Display the i4blt syntax:

```
i4blt -h
```

i4cfg - Configuration Tool

Use the `i4cfg` command as an alternative to the Configuration Tool script to configure your machine to perform various roles in the licensing environment. Before coding the `i4cfg` command, see “Before You Configure” on page 66 to plan your configuration requirements.

Syntax

i4cfg
[**-a** { { **c,n,s,r** } | { **C,N,S,R** } }]
[**-e** { **a** | { **e,t,w,c,g,v,m,p,s** } }]
[**-l** *logfile_path*]
[**-S** { **a,n,s** }]
[**-R** { **a,n,s** }]
[**-b** { "*binding_list*" | **null** }]
[**-t** "*transport_list*"]
[**-n** { **c** | **l** | **g** | **n** }]
[**-c** { **d** | **a** | *cell_uuid* }]
[**-r** { **first** | **from:ip:***host_name* }]
[**-G** { "*site_list*" | **null** }]
[**-d** { *option_string* | **all** }]
[**-script**]
[**-start**]
[**-stop**]
[**-list**]
[**-h**]

Parameters

-a

The roles the machine is to play in your licensing environment. Code any combination of these values, optionally separated by commas:

- **c** Reset the current role of the machine to network license client.
- **n** Reset the current role of the machine to nodelocked license server.
- **s** Reset the current role of the machine to network license server.
- **r** Reset the current role of the machine to central registry license server.
- **C** Update the current role of the machine to include network license client.
- **N** Update the current role of the machine to include nodelocked license server.
- **S** Update the current role of the machine to include network license server.
- **R** Update the current role of the machine to include central registry license server.

-b "*binding_list*"

The complete list of servers (network license servers, nodelocked license servers, and central registry license server) with which this machine will communicate in a direct binding environment. Enclose the complete list in double quotes.

Specify the network license servers, nodelocked license servers, and central registry license server as follows:

'**network ip:***network_address1* [*port_number1*] **ip:***network_address2* [*port_number2*] ... '

'**nodelocked ip:***network_address1* [*port_number1*] **ip:***network_address2* [*port_number2*] ... '

'**registry ip:***network_address* [*port_number*]'

Code **-b null** to delete all previously specified entries from the binding list.

-c

The NCS cell the machine is to join. This parameter is meaningful only if namespace binding support is enabled (see the **-n** parameter). Code one of the following:

d The default cell.

a A new alternate cell. The Configuration Tool creates the UUID. You can retrieve the UUID from the glb_obj.txt file.

cell_uuid An alternate cell with the specified UUID.

If you are configuring as a GLB replica (**-r from**), code this parameter to specify which cell this server is to join.

-d *option_string*

Display the current configuration settings for the **i4cfg** options specified in *option_string*. Code **all** to see the current settings of all the options.

For example, `i4cfg -d elS` requests a display of which events are being logged, the path to the log databases, and a list of startup options showing which are enabled and which are disabled.

-e

The list of events you want to be logged. Code **a** to log all events, or any combination of these values, optionally separated by commas:

e - Errors
Describes server errors that do not stop the server, but return a status code and a message. This is logged by default.

t - License timeout
Tells you that the server has canceled the request for a license because the check period expired. This is not logged by default.

w - License wait
Tells you when a license request cannot be satisfied because no licenses are available, and the user is added to a queue. This is not logged by default.

c - License checkin
Tells you when a licensed product has sent a check-in call to the server to notify that the product is running. This is not logged by default.

g - License grant/release
Tells you when a license was granted or released. This is not logged by default.

v - Vendor added/deleted
Tells you when a product of a new vendor was registered or deleted. This is logged by default.

m - Vendor messages
Provides the log messages the vendor inserted in the enabled product. This is logged by default.

p - Product added/deleted
Tells you when a new product was registered or deleted. This is logged by default.

s - Server start/stop
Logs the successful start or stop of a license server. This is not logged by default.

-G "*site_list*"
This parameter is meaningful only if namespace binding support is enabled (see the **-n** parameter). If your system does not support broadcasting or if the global location broker is running on a machine in a different subnetwork, use this parameter to set the list of hosts running the global location broker. Clients can contact the servers using the *site_list*. List each server that runs the global location broker, in the form:

ip:*network_address*

Separate the entries with spaces, and enclose the entire list in double quotes.

Code **-G null** to delete a previously-specified site list. In this case, clients must locate global location brokers by broadcasting. Before configuring a machine to join an existing cell, check that there is no glb_site.txt file, or, if the file exists, that it includes a server that is in the cell being joined. Otherwise, use **-G null** to delete the existing site list.

-h
Displays command syntax and usage information about the Configuration Tool command-line interface.

-l *logfile_path*
The path in which you want log files to be stored.

-list
Displays a list of active subsystems.

i4cfg - Configuration Tool

-n

Specifies namespace binding support. Code one of the following:

c Namespace binding support as a network license client only.

l This machine is to run the local location broker but not the global location broker.

g This machine is to run the global location broker and the local location broker.

n No namespace binding support (direct binding only).

-R

Startup options that you want to disable. Code any combination of these values, optionally separated by commas:

a Automatic startup of subsystems at system startup (disabled by default)
n Remote administration of nodelocked license server (disabled by default)
s Remote administration of network license server (enabled by default)

Note: This parameter is not valid for network clients.

-r

This parameter is meaningful only if namespace binding support is enabled and this machine is to run the global location broker (see the **-n** parameter). Code **first** if this is to be the first global location broker in a cell. Code **from:ip:***host_name* to replicate the global location broker that already exists on *host_name*.

If you code **-r from,** you must also code the **-c** parameter to specify which cell this server is to join.

-S

Startup options that you want to enable. Code any combination of these values, optionally separated by commas:

a Automatic startup of subsystems at system startup (disabled by default)
n Remote administration of nodelocked license server (disabled by default)
s Remote administration of network license server (enabled by default)

Note: This parameter is not valid for network clients.

-script

Starts the interactive script to configure your machine using a guided step-by-step procedure.

-start

Starts all the subsystems you have configured to run on the machine.

-stop

Stops all the subsystems that are running on your machine.

-t "*transport_list*"

Use this parameter to change the default port numbers, as follows:

"**ip** '*netls_port,crls_port,nodls_port*'"

The three subparameters are positional. If you omit one, its value is reset to the default. For example:

```
"ip ',1 999,1215'"
```

Examples

1 Configure a standalone nodelocked license server, specifying automatic startup of the server and customizing the path to the log files and the selection of events logged:

```
i4cfg -a n -S a -e evmps -l /home/baratti
```

2 Configure a nodelocked license server in a network. Specify automatic startup of the server, make it possible to administer licenses on another nodelocked licensed server (*louise*) remotely, and customize the path to the log files and the selection of events logged.

With direct binding:

```
i4cfg -a n -S a,n -e evmps -l /home/baratti -b 'nodelocked ip:louise' -n n
```

With namespace binding, joining an existing cell that has UUID 456b91c50000.0d.00.00.87.84.00.00.00:

```
i4cfg -a n -S a,n -e evmps -l /home/baratti -b null -n l
-c 456b91c5    . d.  .  .87.84.  .  .
```

Note that the nodelocked license server *louise* must belong to the same cell.

3 Configure a network license server (*thelma*). Specify automatic startup of the server, and customize the path to the log files and the selection of events logged. Configure to communicate with:

Network license server *louise*
Nodelocked license server *louise*
Nodelocked license server *speedy*
Central registry license server *speedy*

With direct binding:

```
i4cfg -a s -S a,s -e cegvp -l /home/baratti -b "'network ip:thelma
ip:louise' 'nodelocked ip:speedy ip:louise' 'registry ip:speedy'"
-n n
```

With namespace binding, starting a new alternate cell:

```
i4cfg -a s -S a,s -e cegvp -l /home/baratti -b null -n g -r first
```

Note that *speedy* and *louise* must join this new cell.

4 Configure a network license client that will communicate with a machine named *thelma* that is configured as both a network license server and the central registry license server.

With direct binding:

```
i4cfg -a c -b "'network ip:thelma' 'registry ip:thelma'" -n n
```

With namespace binding, joining an existing cell that has UUID 456b91c50000.0d.00.00.87.84.00.00.00:

```
i4cfg -a c -b null -n c -c 456b91c5    . d.  .  .87.84.  .  .
```

Note that *thelma* must belong to the same cell.

5 Configure a machine named *thelma* as the central registry license server and a network license server. Configure to communicate with a network license server named *hydra*. Specify automatic startup of the servers.

With direct binding:

```
i4cfg -a s,r -S a,s -b "'network ip:thelma ip:hydra' 'registry ip:thelma'" -n n
```

With namespace binding, joining an existing alternate cell that has UUID 789b91c50000.0d.00.00.87.84.00.00.00 and replicating the global location broker at the server hydra:

```
i4cfg -a s,r -S a,s -b null -n g -r from:ip:hydra
-c 789b91c5    . d.  .  .87.84.  .  .
```

6 Cancel all entries previously made in the direct binding servers list:

```
i4cfg  -b null
```

7 Display the command syntax and usage:

```
i4cfg  -h
```

License Use Runtime and NCS Tools

This section contains information on the following License Use Runtime and NCS tools:

Local Broker Administration (lb_admin)
Administers the registration of the servers in global location broker or local location broker databases. It can be used to look up information, add new entries, and delete existing entries in a specified database.

GLBD Replicas Administration (drm_admin)
Monitors and modifies the list of the replicated versions of the global location broker databases. It can be used to modify, or merge databases to force convergence among replicas, to stop servers, and to delete replicas.

GLBs List (lb_find)
Lists the servers running the global location broker in the network.

UUID Generator (uuid_gen)
Generates the UUID for an NCS cell.

Test Verification Tool (i4tv)
Verifies that license servers are running properly.

Target View Tool (i4target)

Displays the target ID of your machine. The vendor of a licensed product may ask you to provide the target ID of the machine on which the license is to be installed.

1. Use the NCS tools only on servers that are configured in namespace binding mode, since the direct binding configuration does not use NCS location broker services.
2. To prevent conflicts with other NCS daemons that may be running on your machine, use the NCS tools from the `/etc/lum/ncs` directory. To be sure the version used is the one in that directory, enter the commands in the form:

   ./lb_admin
   ./drm_admin
   ./lb_find
   ./uuid_gen

lb_admin - Local Broker Administration

The Local Broker Administration tool (lb_admin) administers the registrations of NCS-based servers in global location broker (GLB) or local location broker (LLB) databases. A server registers universal unique identifiers (UUIDs) specifying an object, a type, and an interface, along with a socket address specifying its location. A client can locate servers by issuing lookup requests to GLBs and LLBs.

Use the Local Broker Administration tool (lb_admin) to look up information, add new entries, and delete existing entries in a specified database.

The Local Broker Administration tool is useful for inspecting the contents of location broker databases and for correcting database errors. For example, if a server terminates abnormally without unregistering itself, use Local Broker Administration (lb_admin) to manually remove its entry from the GLB database.

When accepting input or displaying output, Local Broker Administration (lb_admin) uses either character strings or descriptive textual names to identify objects, types, and interfaces. A character string directly represents the data in a UUID in the format:

`xxxxxxxxxxxx.xx.xx.xx.xx.xx.xx.xx.xx`

where each x is a hexadecimal digit.

Local Broker Administration (lb_admin) will examine or modify only one database at a time. This is referred to as the current database. The use_broker command selects the type of location broker database, GLB or LLB. The set_broker command selects the host whose GLB or LLB database is to be accessed. Of course, if one replica of a replicated GLB database is modified, the modifications will be propagated to the other replicas of that database.

License Use Runtime and NCS Tools

Syntax

lb_admin [**-nq**] [**-version**]

Parameters

-nq
Do not query for verification of wildcard expansions in unregister operations.

-version
Display the version of NCS that this lb_admin belongs to, but do not start the tool.

Commands

When you type:

```
lb_admin
```

you are prompted with the following line, where you can enter the lb_admin commands:

```
lb_admin:
```

In lookup, register, and unregister commands, the object, type, and interface arguments can be either character strings representing UUIDs or textual names corresponding to UUIDs, as described earlier.

a[dd]
Synonym for register.

c[lean]
Find and delete obsolete entries in the current database.

When issuing this command, lb_admin attempts to contact each server registered in the database. If the server responds, the entry for its registration is left intact in the database. If the server does not respond, lb_admin looks up its registration in the LLB database at the host where the server is located, tells the result of this lookup, and asks if the entry is to be deleted. If a server responds, but its UUIDs do not match the entry in the database, lb_admin tells this result and asks if the entry is to be deleted.

Entries that meet either of these conditions are probably safe to delete:

The server does not respond. The lb_admin succeeds in contacting the LLB at the host where the server is located, but the server is not registered with that LLB. The server is probably no longer running.

A server responds, but its UUIDs do not match the entry in the database. The server that responds is not the one that registered the entry.

Entries that meet either of these conditions are probably safe to delete.

In other situations, it is best not to delete the entry unless it can be verified directly that the server is not running (for example, by listing the processes running on its host).

When lb_admin asks to delete an entry, you can respond in four ways:

A y[es] response deletes the entry.

A n[o] response leaves the entry intact in the database. After a yes or a no, lb_admin proceeds to check the next entry in the current database.

A g[o] response invokes automatic deletion, in which all eligible entries are deleted and all ineligible entries are left intact, without the user being queried, until all entries have been checked.

A q[uit] response terminates the clean operation.

d[elete]

Synonym for unregister.

h[elp] [command] or ? [command]

Display a description of the specified command or, if none is specified, list all of the lb_admin commands.

l[ookup] object type interface

Look up and display all entries with matching object, type, and interface fields in the current database. You can use an asterisk as a wildcard for any of the parameters. If all the parameters are wildcards, lookup displays the entire database.

q[uit]

Exit the lb_admin session.

r[egister] object type interface location annotation [flag]

Add the specified entry to the current database. Use an asterisk to represent the null UUID in the object, type, and interface fields.

The location is a string in the format family:host[port], where *family* is an address family, *host* is a host name, and *port* is a port number. A leading # can be used to indicate that a host name is in the standard numeric form.

The following are sample location specifiers:

```
ip:vienna[1756]
ip:#192.5.5.5[1791]
```

The annotation is a string of up to 64 characters annotating the entry. Use double quotation marks to enclose a string that contains a space or contains no characters. To embed a double quotation mark in the string, precede it with a backslash.

The flag is either local (the default) or global, indicating whether the entry should be marked for local registration only or for registration in both the LLB and GLB databases. The flag is a field that is stored with the entry but does not affect where the entry is registered. The set_broker and use_broker commands select the particular LLB or GLB database for registration.

s[et_broker] [broker_switch] host

Set the host for the current LLB or GLB. If specifying global as the broker_switch, set_broker sets the current GLB, otherwise it sets the current LLB. The host is a string in the format family:host, where family is an address family and host is a host name. Use a leading # to indicate that a host name is in the standard numeric form. The following are sample location specifiers:

```
ip:prague
ip:#192.5.5.5
```

Issue use_broker, not this command, to determine if subsequent operations will access the LLB or the GLB.

set_t[imeout] [short | long]

Set the timeout period used by lb_admin Administration for all of its operations. With an argument of short or long, set_timeout sets the timeout accordingly. With no argument, it displays the current time-out value.

u[nregister] object type interface location

Delete the specified entry from the current database.

The location is a string in the format family:host[port], where *family* is an address family, *host* is a host name, and *port* is a port number. Use a leading # to indicate that a host name is in the standard numeric form. The following are sample location specifiers:

```
ip:vienna[1756]
ip:#192.5.5.5[1791]
```

You can use an asterisk as a wildcard in the object, type, and interface fields to match any value for the field. Unless queries have been suppressed by invoking lb_admin with the -nq option, unregister allows deletion of each matching entry.

A y[es] response deletes the entry.

A n[o] response leaves the entry in the database.

A g[o] response deletes all remaining database entries that match, without querying.

A q[uit] response terminates the unregister operation, without deleting any additional entries.

us[e_broker] [broker_switch]

Select the type of database that subsequent operations will access, GLB or LLB. The broker_switch is either global or local. If a broker_switch is not supplied, use_broker determines if the current database is global or local.

Use set_broker to select the host whose GLB or LLB is to be accessed.

Example

1. Set the global location broker as the default database.

   ```
   lb_admin
   lb_admin: use global
   ```

2. Find and delete obsolete entries in the global location broker database.

   ```
   lb_admin: clean
   ```

   This is the output, if there are no entries to be cleaned:

   ```
          Entries deleted of 8 processed
   ```

3. Exit the tool:

   ```
   lb_admin: Quit
   ```

drm_admin - GLBD Replicas Administration

The GLBD Replicas Administration tool (drm_admin) administers servers based on the NCS location brokers such as glbd, the replicated version of the global location broker. With the GLBD Replicas Administration tool (drm_admin), the replica lists can be inspected or modified, databases can be merged to force convergence among replicas, servers can be stopped, and replicas can be deleted.

The role of the GLBD Replicas Administration tool (drm_admin) is to administer the databases, not to change the data they contain. For instance, you can use GLBD Replicas Administration (drm_admin) to merge two replicas of the global location broker database, but the Local Broker Administration (lb_admin) must be used to add a new entry to the database.

Also, although GLBD Replicas Administration (drm_admin) can stop or delete a global location broker replica, glbd must be invoked directly to start or create a replica. After you start it, GLBD Replicas Administration (drm_admin) enters an interactive mode in which it accepts the following commands.

Syntax

drm_admin [**-version**]

Parameters

-version

Displays the version of NCS that this GLBD Replicas Administration (drm_admin) belongs to, but does not start the tool.

Commands

When you type:

```
drm_admin
```

you are prompted with this line:

```
drm_admin:
```

where you can enter the drm_admin commands.

Most drm_admin commands operate on a default object (default_obj) at a default host (default_host). Together, default_obj and default_host specify a default replica. Defaults are established by the set command and are remembered until changed by another set. Currently, the only known object is glb. Some drm_admin commands operate on a host other than the default. Identify this host as other_host. The host name supplied as a default_host or an other_host takes the form family:host, where the host can be specified either by its name or by its network address. The following are examples of acceptable host names:

```
ip:bertie
ip:#192.5.5.5
```

addrep other_host
: Add other_host to the replica list at default_host. The replica at default_host will propagate other_host to all other replica lists for default_obj.

chrep -from other_host -to new_other_host
: Change the network address for other_host in the replica list at default_host to new_other_host. The replica at default_host will propagate this change to all other replica lists for default_obj. The chrep command will fail if a replica of default_obj is running at other_host or if other_host is not on the replica list at default_host.

delrep other_host
: Delete the replica of default_obj at other_host. The delrep command tells the replica at other_host to do the following:

  1. Propagate all of the entries in its propagation queue.
  2. Propagate a delete request to all other replicas, causing other_host to be deleted from all other replica lists for default_obj.
  3. Delete its copy of default_obj.
  4. Stop running.

  The **delrep** command returns you immediately to the GLBD Replicas Administration prompt, but the actual deletion of the replica can take a long time for configurations that are not stable and intact. Check to see if the daemon for the deleted replica has stopped by listing the processes running on its host.

info

Get status information about the replica for default_obj at default_host.

lrep [-d] [-clocks] [-na]

List replicas for default_obj as stored in the replica list at default_host.

The -d option lists deleted as well as existing replicas.

The -clocks option shows the current time on each host and indicates the time difference between the replicas.

The -na option lists the network address of each host.

merge { -from | -to } other_host

The **merge** command copies entries in the default_obj database and replica list from one replica to another. It copies an entry if no corresponding entry exists in the destination database or if the corresponding entry in the destination database bears an earlier time stamp. A merge does not cause entries to be propagated. The database and replica list at the origination are not changed.

The **-from** parameter copies entries from the default_obj database and replica list at other_host to the default_obj database and replica list at default_host.

The **-to** parameter copies entries from the database and replica list at default_host to the database and replica list at other_host.

A **merge -from** followed by a **merge -to** causes the replicas at the two hosts to converge.

merge_all

The **merge_all** command uses default_host as the hub for a global merge of all replicas for default_obj. For each host on the replica list at default_host, a **merge_all** first runs a **merge -from**, then runs a **merge -to**. All replicas of default_obj are thereby forced into a consistent state. The **merge_all** operation does not cause any entries to be propagated. You should run a **merge_all** when:

A replica is purged
A replica is reset
A replica has been not communicating for two weeks or more
A replica *stops* (for example, when its database is destroyed by a disk failure)

monitor [-r n]

This command causes drm_admin to read the clock of each replica of default_obj every n minutes and to report any clock skews or non-answering replicas. If **-r** is not specified, the period is 15 minutes.

purgerep *other_host*

The **purgerep** command purges other_host from the replica list at default_host. The replica at default_host then propagates a delete request to the replicas at the hosts remaining on its list, thereby removing other_host from all other replica lists for default_obj. The delete request is

not sent to other_host. A **purgerep** can cause data to be lost and should only be used when a replica has "stopped." It is strongly recommended that a **merge_all** operation be run after the **purgerep** to prevent the remaining replicas of the default_obj database from becoming inconsistent. If the purged replica is still running, it should be reset. It is recommended that you use **chrep** (rather than **addrep** and **purgerep**) to change entries on the replica list.

quit
Quit the drm_admin session.

reset other_host
Reset the replica of default_obj at other_host. The reset command tells the replica at other_host to delete its copy of default_obj and to stop running. It does not cause other_host to be deleted from any other replica lists. This command can cause data to be lost unless a successful merge_all is run first.

set [-o obj_name] -h host_name
Set the default object and host. All subsequent commands will operate on obj_name. Subsequent commands that do not specify a host will be sent to host_name. If the -o option is not specified, drm_admin keeps the current default_obj. If set is used with the -o option, drm_admin checks the clocks at all hosts with replicas of the object.

stop
Stop the server for default_obj that is running at default_host.

Example

The following example starts drm_admin, sets the default object to glb, and sets the default host to ip:mars:

```
drm_admin
drm_admin: set -o glb -h ip:mars
```

This is the output:

```
Default object: glb default host:
ip:mars state: in service
Checking clocks of glb replicas
ip:mars 1997/ 4/ 9.17: 9
ip:pluto 1997/ 4/ 9.17: 9
ip:mercury 1997/ 4/ 9.17: 7
```

lb_find - GLBs List

The GLBs List (lb_find) lists global location broker processes and their attributes. It sends out inquiries to the NCS location broker processes and gathers the responses.

If on the machine where you issue the command there are not the two files:

```
glb_site.txt
glb_obj.txt
```

lb_find finds all the GLB processes of the same subnet, or all the GLB processes it can reach via broadcast.

If you want to see GLB processes of a different subnetwork you must have those two files. In such a case lb_find finds all the GLB processes that run on the hosts whose addresses are in the *glb_site.txt*, situated in the cell specified in the *glb_obj.txt*. If the glb_obj.txt is not on the machine, the default cell is taken.

The results are analyzed to determine whether or not the global location broker can be replicated, and which cell each daemon serves. After 10 seconds, the results are summarized, showing the server host's network address, the port number, the global location broker type, a cell name of either default or alternate_n, where n is a number greater than or equal to 1, and the cell's UUID.

Syntax

lb_find [**-dl**] [**-f ip** | **-q**] [**-h**] [**-v**]

Parameters

-dl Turn on RPC remote procedure call (RPC) debugging while searching for GLB servers.

-f Query for the global location broker servers that communicate with the specified protocol in all the cells.

-q Query for a global location broker server using the standard RPC mechanism. At most, one global location broker server is printed, and only servers in the current machine's cell are searched. The program exits with a status of 0 if a global location broker server is found; otherwise, the status is nonzero.

-h Print out the help for the command.

-v Print out the NCS version string.

Example

A network contains two global location broker processes (glbd) in the default NCS cell.

```
lb_find
```

This is the output:

```
sent to broadcast address ip:#9.87.22 .255
waiting for replies
ip:server5(9.87.22 .5) 1 24 replicatable default
333b91c5    . d.  .  .87.84.  .  .
ip:server3(9.87.22 .3) 1 72 replicatable default
333b91c5    . d.  .  .
87.84.  .  .
```

License Use Runtime and NCS Tools

uuid_gen - UUID Generator

Use the uuid_gen tool to generate the UUID (universal unique identifier) for an NCS cell. The UUID is 28 hexadecimal characters string, and is contained in the *glb_obj.txt* file.

Syntax

uuid_gen

Example

To generate the UUID:

```
uuid_gen
```

This is an example of the output:

```
54c7874546ae. .2.81.87.92.34. . .  .
```

i4tv - Test Verification Tool

Use the i4tv tool after the license servers are started to verify that they are running properly. A message describing a completed license transaction and a list of all license servers will be displayed.

Syntax

i4tv { [**-n** *hostname*] [**-z**] [**-v**] | { **-h** | **-usage** | **-version** }
[**-p** *number_of_transactions*] }

Parameters

-n ***hostname***
Checks that the specified machine is running a network license server. It returns 0 if the hostname is running a network license server and 1 if the hostname is not running a network license server.

-z Turns on NCS remote procedure call (RPC) tracing messages, which can be used to diagnose problems.

-v Displays progress messages during the license request operation.

-h Displays command usage information (same as **-usage**). This parameter is valid only when issued without other parameters.

-usage
Displays command usage information (same as **-h**). This parameter is valid only when issued without other parameters.

-version
Displays command version information. This parameter is valid only when issued without other parameters.

-p [*number_of_transactions*]
Specifies the number of transactions to be completed before performance information is displayed. This information provides averages for the specified

period. It can be used for tuning the system and for troubleshooting performance problems. The default value is 1000.

Example

Run the i4tv test and verification tool:

```
i4tv
```

Check for the presence of the license server pluto:

```
i4tv -n pluto
```

i4target - Target View Tool

Use the i4target tool to display the target ID of your machine.

Syntax on IRIX and Solaris

i4target [**-O** | **-V** | **-o** | **-l** | **-h**] [**-v**]

Syntax on HP-UX

i4target [**-O** | **-V** | **-o** | **-c** | **-h**] [**-v**] [**-q**]

Parameters

-O Displays the target identifier of the machine on which you issue the command, in the form that the license creation tool accepts.

-V Displays command version information.

-o Displays the operating system name of your machine.

-l Displays all target IDs of the machine, starting with the most secure (the one based on the network adapter, if it is available). (IRIX and Solaris only.)

-c Displays multiple target IDs for machines that have more than one network connection, and enables you to change from the one that is currently being used to another. (HP-UX only.)

-h Displays command usage information.

-v Displays information in verbose mode.

-q Displays information in quiet mode (comments are not displayed before the output). (HP-UX only.)

License Use Runtime and NCS Subsystems

Read this section for reference information on License Use Runtime and NCS subsystems.

License Use Runtime and NCS Subsystems

To prevent conflicts with other NCS daemons that may be running on your machine, use the NCS-related subsystems from the `/etc/lum/ncs` directory. To be sure the version used is the one in that directory, enter the commands in the form:

```
./llbd
./glbd
./i4glbcd
```

Run the other commands in this section from the appropriate directory for your operating system:

```
/opt/lum/ls/os/hpux/bin
/opt/lum/ls/os/svr4.sgi/bin
/opt/lum/ls/os/solaris/bin
```

llbd - Local Location Broker Subsystem

The local location broker subsystem (llbd) is part of the network computing system (NCS). It manages the local location broker (LLB) database, which stores information about NCS based server programs running on the local host.

A host must run llbd if it is to support the location broker forwarding function or to allow remote access, for example by the Local Broker Administration (lb_admin) to the LLB database. In general, any host that runs NCS-based servers should run an llbd, and llbd should be running before any such servers are started. Additionally, any network or internet supporting NCS activity should have at least one host running a global location broker subsystem (glbd).

Syntax

startprc llbd [**-dl**]

Parameters

-dl

Prints debugging information.

glbd - Global Location Broker Subsystem

The global location broker (GLB) subsystem (glbd) helps clients to locate servers on a network or internet. The GLB database stores the locations (that is, the network addresses and port numbers) where server subsystems are running. A process maintains this database and provides access to it.

You can replicate the GLB database to increase its availability. Copies of the database can exist on several hosts, with glbd running on each of those hosts to maintain the consistency of the database replicas. (In an internet, at least one glbd must run in each network.) Each replica of the GLB keeps a list of all the other GLB replicas. The GLBDs Replicas Administration (drm_admin) tool administers the replication of the GLB database and of the replica list (see “License Use Runtime and NCS Tools” on page 112).

In an internet, all routing nodes must support the same family. If a set of global location broker replicas includes systems that support only HP-UX, IRIX, and Solaris, all replicas must use IP protocols to communicate with each other. A replica running on an HP-UX, IRIX, and Solaris system can communicate with other replicas using IP protocols, but still provide lookup and update services to its clients.

The glbd command writes diagnostic output to the file /etc/lum/ncs/glb_log.

Syntax

startprc glbd [**-create** { **-first** [**-family ip**] | **-from** *host_name* }] [**-debug**] [**-log_stdout**]

Parameters

-create
Creates a replica of the GLB. This option creates a GLB database in addition to starting a broker subsystem. It must be used with either **-first** or **-from**.

-first
Use this option only with **-create**. Use it to create the first replica (that is, the very first instance) of the GLB on the network or internet.

-family *family_name*
Use this option only with **-first**, to specify the address family that the first replica will use to identify itself on the replica list. Any subsequently created replicas must use this family to communicate with this replica. The family_name can only be **ip**.

-from *host_name*
Use this option only with **-create**, to create additional replicas of the global location broker. A replica of the global location broker must exist at host_name. The database and replica list for the new replica are initialized from those at host_name. The replica at host_name adds an entry for the new replica to its replica list and propagates the entry to the other global location broker replicas. A host_name takes the form family:host, where the host can be specified either by its name or by its network address.

The following are examples of acceptable host names:

```
ip:bertie
```

```
ip:#192.5.5.5
```

The new replica uses the same address family as host_name in identifying itself on the replica list. For example, if host_name is an IP address, the new replica is listed by its IP address on the replica list.

-debug
Prints debugging information.

-log_stdout
Redirects the log and debug printout to standard output instead of /etc/lum/ncs/glb_log.

Examples

Create and start for the first time the first replica of the GLB on the network or internet:

```
startprc glbd -create -first -family ip
```

Start for the first time a subsequent replica of the GLB, initializing its database from host //buddy:

```
startprc glbd -create -from ip:buddy
```

Restart an existing replica of the GLB:

```
startprc glbd
```

i4lmd - Network License Server Subsystem

The i4lmd subsystem starts the network license server on the local node. If the machine is not configured to run the network license server, i4lmd has no effect.

The parameters of i4lmd override the corresponding settings in the i4ls.ini file.

Syntax

startprc i4lmd [**-no** *event_list*] [**-v**] [**-z**] [**-l** *log_name*] [**-s**] [**-r**] [**-c**] [**-p**]

Parameters

-no ***event_list***

Turns off logging of the events specified in event_list. Any combination of events is valid, but items in the list of events must not be separated by spaces or other characters. Following are the event types that you can specify:

l Grant and release licenses.

c Check in licenses. (Licensed products usually check in with the license server at regular intervals while a user is using the product.)

w Waiting events: these include wait events (a user was waiting for a license), wait grant events (a user was waiting for and then was granted a license), and wait remove events (a user was waiting for a license and then asked to be removed from the queues before a license was granted.)

v Vendor events: a vendor was added, renamed or deleted.

p Product events: a product was added, renamed, or deleted.

e Errors.

m Messages.

s Starts and stops of this license server.

t License timeout events. (When a licensed product fails to check in with the license server, it may stop running after it times out. The vendor of the product sets the timeout interval, which is how long a product can run after it has lost contact with the license server.)

-v
License Use Runtime library verbose mode.

-z
Debugging flag. Prints RPC debugging information.

-l ***log_name***
Overrides the default name and location of the file used to store log information. This allows the I/O activity to the files used by the license server to be spread across multiple file systems that may become important for large installations.

-s
Instructs the license server to ignore attempts from administrators on remote systems to modify the license database. Records in the database remain readable by all instances of the License Use Runtime Administration Tool.

-r
Recovers files from the automatic backup version.

-c
Specifies that this is a cold start, meaning that the license server restarts from scratch, as if it had granted no licenses before stopping.

-p
Specifies that i4lmd is to display performance information at specified intervals. The default and maximum interval is 1000 calls received from clients. To change the frequency of reporting, change the environment variable I4_POLL_FREQ.

Examples

Start a license server and do not log checkin, vendor, product, timeout, or message events:

```
startprc i4lmd -no cvptm
```

Start a license server changing the default log-file:

```
startprc i4lmd -l /lum/ls/my_log
```

i4llmd - Nodelocked License Server Subsystem

The i4llmd subsystem starts the nodelocked license server on the local node. If the machine is not configured to run the nodelocked license server, i4llmd has no effect.

The parameters of i4llmd override the corresponding settings in the i4ls.ini file.

Syntax

startprc i4llmd -b [**-no** *event_list*] [**-v**] [**-l** *log_name*] [**-s**] [**-r**] [**-c**]

Parameters

no ***event_list***

Turns off logging of the events specified in event_list. Any combination of events is valid, but items in the list of events must not be separated by spaces or other characters. Following are the event types that you can specify:

- **l** Grant and release licenses.
- **v** Vendor events: a vendor was added, renamed or deleted.
- **p** Product events: a product was added, renamed, or deleted.
- **e** Errors.
- **m** Messages.
- **s** Starts and stops of this license server.
- **t** Time out.

-v

License Use Runtime library verbose mode.

-l ***log_name***

Overrides the default name and location of the file used to store log information. This allows the I/O activity to the files used by the license server to be spread across multiple file systems that may become important for large installations.

-s

Instructs the license server to ignore attempts from administrators on remote systems to modify the license database. Records in the database remain readable by all instances of the License Use Runtime Administration Tool.

-r

Recovers files from the automatic backup version.

-c

Specifies that this is a cold start, meaning that the license server restarts from scratch, as if it had granted no licenses before stopping.

Examples

Start a nodelocked license server and do not log checkin, vendor, product, or message events:

```
startprc i4llmd -b -no cvpm
```

Start a nodelocked license server changing the default log file:

```
startprc i4llmd -b -l /lum/ls/my_log
```

Start a nodelocked license server, disabling remote administration from instances of the Basic License Tool on other machines:

```
startprc i4llmd -b -s
```

i4gdb - Central Registry License Server Subsystem

The Central Registry is a License Use Runtime subsystem that provides a mechanism for storing data pertaining to licensing information. There must be one and only one

central registry license server running per cell. This ensures that the data is accurate and complete.

The Basic License Tool requires a central registry license server up and running to administer customer-managed use products.

In namespace binding, if more than one i4gdb is found in a given cell, the newly started i4gdb automatically shuts down.

In direct binding there is no such control, and you must double-check that you have started one and only one central registry license server in your licensing environment by issuing the following command on every License Use Runtime server:

```
i4cfg -list
```

If the machine is not configured to run the central registry license server, i4gdb has no effect.

The parameters of i4gdb override the corresponding settings in the i4ls.ini file.

Syntax

startprc i4gdb -a [**-no** *event_list*] [**-v**] [**-l** *log_name*] [**-r**] [**-c**] [**-z**]

Parameters

no ***event_list***

Turns off logging of the events specified in event_list. Any combination of events is valid, but items in the list of events must not be separated by spaces or other characters. Following are the event types that you can specify:

- **l** Grant and release licenses.
- **c** Check in licenses. (Licensed products usually check in with the license server at regular intervals while a user is using the product.)
- **v** Vendor events: a vendor was added, renamed or deleted.
- **p** Product events: a product was added, renamed, or deleted.
- **e** Errors.
- **m** Messages.
- **s** Starts and stops of this license server.
- **t** Time out.

-v

License Use Runtime library verbose mode.

-l ***log_name***

Overrides the default name and location of the file used to store log information. This allows the I/O activity to the files used by the license server to be spread across multiple file systems that may become important for large installations.

i4lct - License Creation Tool

-r
Recovers files from the automatic backup version.

-c
Specifies that this is a cold start, meaning that the license server restarts from scratch, as if it had granted no licenses before stopping.

-z
Debugging flag. Prints RPC debugging information.

i4glbcd - Global Location Broker Database Cleaner Subsystem

The i4glbcd subsystem automatically cleans up incorrect entries in the global location broker database. Do not start more than one instance of i4glbcd in an NCS cell.

Syntax

startprc i4glbcd [**-nq**]

Parameters

-nq
Verbose mode. This causes i4glbcd to display debugging information to standard output. Use this information if you need to call IBM support.

---

i4lct - License Creation Tool

The license creation tool is intended for:

Software vendors, to create test passwords while enabling a product

Software vendors, to create production passwords

Sales representatives, who can be provided with a compound password containing many licenses, from which they extract licenses for individual customers.

This tool is not intended for administrators or end users.

The i4lct command is used to create passwords. Run this command on a machine where License Use Runtime is installed.

1. The passwords you generate with the license creation tool of License Use Runtime Version 4.5.5 also work on License Use Runtime servers and clients of previous releases. High-availability licenses, introduced in Version 4.5.1, can be installed only on machines running Version 4.5.x. Licenses of types introduced in Version 4 (such as reservable and per-seat), cannot be installed on machines running earlier releases of License Use Runtime. Custom configuration licenses, introduced in Version 4.5.5, can be installed only on machines running Version 4.5.5.

In the enrollment certificate file, the *PasswordVersion* parameter is set as follows:

7 If the password is for a custom configuration license
6 If the password is for a high-availability license (and can therefore be installed only on machines running Version 4.5.x)
5 If the password is for a license type, or exercises a policy, introduced in Version 4 (and therefore is not installable on machines running earlier versions)
4 Otherwise

2. To create test passwords, use **test** as the value of the **-i**, **-k**, and **-v** parameters.
3. To extract licenses for individual customers from a compound password assigned to a sales representative, use **supplier** as the value of the **-k** parameter.

   To create production licenses, vendors must acquire the license for this tool from IBM or from Isogon Corp.

   The address of Isogon Corp. is:

   Isogon Corporation
   330 Seventh Avenue
   New York, New York 10001
   U.S.A.
   Tel: (+1) 212-376-3200
   Fax: (+1) 212-376-3280

Table 9 on page 132 summarizes the valid combinations of license type, password use control level, password type, and enabled policies the vendor can specify with i4lct.

i4lct - License Creation Tool

Table 9. Valid Uses of i4lct

License Type	Password Use Control Level	Password Type	Policies
Concurrent (**-l c**)	Customer-Managed (**-R c**)	Compound (**-w c**)	Hard Stop/Soft Stop (**-A s**) Multiuse Rules (**-m**) License Annotation (**-a**)
Concurrent (**-l c**)	Vendor-Managed (**-R v**)	Simple (**-w l**)	Multiuse Rules (**-m**) License Annotation (**-a**) Custom Configuration (**-C**)
Concurrent (**-l c**)	Vendor-Managed (**-R v**)	Compound (**-w c**)	Multiuse Rules (**-m**) License Annotation (**-a**)
Reservable (**-l r**)	Customer-Managed (**-R c**)	Compound (**-w c**)	Hard Stop/Soft Stop (**-A s**) License Annotation (**-a**)
Reservable (**-l r**)	Vendor-Managed (**-R v**)	Simple or Compound (**-w l** or **-w c**)	License Annotation (**-a**)
Use-Once (**-l u**)	Customer-Managed (**-R c**)	Compound (**-w c**)	License Annotation (**-a**)
Use-Once (**-l u**)	Vendor-Managed (**-R v**)	Simple or Compound (**-w l** or **-w c**)	License Annotation (**-a**)
Per-Seat (**-l pt**)	Customer-Managed (**-R c**)	Simple (**-w l**)	Hard Stop/Soft Stop (**-A s**) License Annotation (**-a**)
Per-Server (**-l ps**)	Customer-Managed (**-R c**)	Simple (**-w l**)	Hard Stop/Soft Stop (**-A s**) Multiuse Rules (**-m**) License Annotation (**-a**)
Simple Nodelocked (**-l n**)	Vendor-Managed (**-R v**)	Simple (**-w l**)	License Annotation (**-a**) Custom Configuration (**-C**)
Simple Nodelocked (**-l n**)	Vendor-Managed (**-R v**)	Compound (**-w c**)	License Annotation (**-a**)
Simple Nodelocked (**-l n**)	Vendor-Managed (**-R v**)	Compound Nodelocked (**-w cn**)	Try-and-Buy (**-A t**)* License Annotation (**-a**)
Concurrent Nodelocked (**-l cn**)	Customer-Managed (**-R c**)	Simple (**-w l**)	Hard Stop/Soft Stop (**-A s**) Multiuse Rules (**-m**) License Annotation (**-a**)
Concurrent Nodelocked (**-l cn**)	Vendor-Managed (**-R v**)	Simple (**-w l**)	Multiuse Rules (**-m**) License Annotation (**-a**)
Use-Once Nodelocked (**-l un**)	Customer-Managed (**-R c**)	Simple (**-w l**)	License Annotation (**-a**)
Use-Once Nodelocked (**-l un**)	Vendor-Managed (**-R v**)	Simple (**-w l**)	License Annotation (**-a**)

Note: * When **-w** is set to **cn**, the try-and-buy attribute is required.

i4lct - License Creation Tool

Syntax

i4lct

Parameters required to generate a license:

-i {*vendor_id* | **create** | **test** }
-k { *vendor_key* | **test** | **supplier** }
-v { *vendor_name* | **test** }
-l *license_type*
-p *product_id*
-N *product_name*
-w *password_type*
{ **-d** *duration* | **-e** *expiration_date* }
-r *revision*
-R *password_registration_level*
{ **-T** *target_id* **-t** *target_type* | **-X** *extended_target_id* **-x** *extended_target_type* }

Parameters valid only if -w is set to c or cn (compound or compound nodelocked passwords):

[**-S** *derived_start_date*]
[**-E** *derived_expiration_date*]

Parameter valid only if -R is set to v (vendor-managed product):

[**-n** *number_of_licenses*]

Parameter valid only if -w is set to c or cn (compound or compound nodelocked passwords) *and* -R is set to v (vendor-managed product):

[**-D** *aggregate_duration*]

Parameter valid only if -l is set to c, cn, or ps (concurrent, concurrent nodelocked, or per-server license):

[**-m** *multi-usage_specification*]

Optional parameters:

[**-a** *annotation*]
[**-A** *attributes*]
[**-c** *customer_information*]
| [**-C** *serial_number*]
[**-L** *log_file*]
[**-O**]
[**-P** *16_bit_flag*]
[**-s** *start_date*]

i4lct - License Creation Tool

Parameters valid only when entered without any other parameters:

[**-f** *batch_file_name*]
[**-h**]
[**-V** *version*]
[**-u** *upgrade_flag*]
[**-U**]

Parameters

-a *annotation*
The license annotation string, up to 80 characters long.

-A *attributes*
Possible values are:

s To enable the end user to modify the product policy from soft stop to hard stop and vice versa. Valid only for customer-managed products (**-R** set to **c**).

t To specify a try-and-buy license. Valid only for vendor-managed products (**-R** set to **v**) with nodelocked licenses (**-l** set to **n**) and password type compound nodelocked (**-w** set to **-cn**).

-c *customer_information*
Specifies additional customer details for logging purposes. This parameter is useful only if used with the -L *log_file* parameter.

-C *serial_number*
Specifies the serial number of a custom configuration license. The serial number is a string of up to 31 alphanumeric characters that uniquely identifies a custom configuration.

-d *duration*
The duration of the password. If the password type is license, this value indicates the number of days for which the licenses are valid. If the password type is compound, this value indicates the number of days during which license passwords can be derived from the compound password. Its maximum allowed value is 32 767.

For vendor-managed compound passwords, the product obtained by multiplying **-d** (duration) and **-n** (number_of_licenses) cannot exceed 2 147 483 647.

For example, if -n is 70 000, the maximum duration is 30 678 days (2 147 483 647/70 000).

You must specify at least one of **-d** and **-e**.

-D *aggregate_duration*
Valid only for vendor-managed products (**-R** set to **v**) and compound or compound nodelocked passwords (**-w** set to **c** or **cn**). This is the maximum aggregate duration, in days, of all licenses that are to be derived from a compound password. Its maximum allowed value is 2 147 483 647.

In the case of a try-and-buy license (**-w** set to **cn**, **-A** set to **t**, and **-l** set to **n**), this represents the duration of the try-and-buy license extracted from the compound password.

For example, a compound password from which 100 licenses may be derived might have an aggregate duration of 36500 days. From this password there can be derived 100 1-year licenses, or 50 6-month licenses and 50 18-month licenses, and so on.

-e *expiration_date*
The end date of the password. The date format is mm/dd/yyyy. If the password type is license, this value indicates the end date beyond which the licenses are no longer valid. If the password type is compound, this value indicates the end date beyond which license passwords can no longer be derived from the compound password.

The latest expiration date that can be specified with the -e parameter is 02/05/2106. Note, however, that the standard time functions of the operating system do not properly handle expiration dates later than 12/31/2037, so it is recommended that you not create licenses that expire after that date. Note also that the current version of the operating system does not allow system dates later than 01/18/2038.

You must specify at least one of **-d** and **-e**.

Valid combinations of the start, duration, and end options are as follows:

-d The start date defaults to the current date. i4lct calculates the expiration date for you.

-s and **-d** i4lct calculates the expiration date for you.

-e and **-d** i4lct calculates the start date for you.

-s and **-e** i4lct calculates the duration for you.

-E *derived_expiration_date*
Valid only with compound or compound nodelocked passwords (**-w** set to **c** or **cn**). The date format is mm/dd/yyyy. This is the derived license end date, the date after which no license password derived from the compound password is valid.

-f *batch_file_name*
Specify the fully qualified path and file name of a batch file containing the full i4lct command to issue the full i4lct command contained in such a file.

-h Displays help for the i4lct command.

i4lct - License Creation Tool

-i *vendor_id*
Specifies the vendor ID. It can also assume the following values:

create Specify it to generate a new vendor ID while generating a production password.
test Specify it if you are creating test passwords.

-k *vendor_key*
Specifies the vendor key. This must be an integer between 1 and 2 147 483 647, or one of the following values:

test Specify it if you are creating test passwords.
supplier When you specify this value the license server must be up and running, and there must be a compound password enrolled for a vendor-managed use product.

By specifying this value you create an enrollment certificate file for a simple password extracted from the existing compound. You specify the compound password by means of the other i4lct parameters. The following example creates the certificate file for 497 concurrent licenses with duration 10 days, extracted from the compound password of the vendor-managed use product *cmpLev3* of the vendor Operatix:

```
i4lct -i 6pw4ci1xw   . n.  . 3.4g.5y.  .  .   -k supplier
-n 497 -l c -d 1  -N "cmpLev3" -p 317 -r 1.  -t any -T any
-v "Operatix" -w l
```

The use of this parameter is suggested when you have sales representatives in other locations. You can generate a compound password with a big number of licenses, and provide them with it. They enroll the password and then generate the licenses for customers extracting simple passwords from your compound. This will prevent you from generating the enrollment certificate files for all the customers, or from having to supply the production i4lct to all your representatives.

-l *license_type*
The license type. Use one of the following keywords:

c Concurrent
cn Concurrent nodelocked
n Nodelocked
u Use-once
un Use-once nodelocked
ps Per-server
pt Per-seat
r Reservable

Multiuse rules, **-m**, can be specified only if this parameter is set to concurrent, concurrent nodelocked, or per-server.

-L *log_file*
Specify the i4lct log file path and name. If you do not specify it the default is

```
var/lum/i4lct.log
```

-m *multi-usage_specification*
This argument is optional and is used to define multiuse rules for concurrent, concurrent nodelocked, and per-server licenses.

You can define conditions for multiuse of a single concurrent license as any combination of the following key letters: **u** (same user), **n** (same node), **g** (same group or same display, depending on the license-enabled application), **j** (same job ID).

You can define conditions for multiuse of a single concurrent nodelocked or per-server license as any combination of the following key letters: **u** (same user), **g** (same group or same display, depending on the license-enabled application), **j** (same job ID).

For details about the **g** (same group or same display) parameter, see "Defining Rules for Multiple-Use Concurrent Licenses" on page 140.


Specify the letters without spaces, commas, or other separators. For example, **-m un** means that if the user and node are the same as those associated with a previously granted license, granting a new concurrent access license is not required.

-n *number_of_licenses*
For a compound password, this is the maximum number of licenses that can be derived from the password. It is valid only for vendor-managed products (**-R** set to **v**). Its maximum allowed value is 65 534. For customer-managed products, you cannot specify this parameter, and the value is set to 65 535.

For vendor-managed compound passwords, the product obtained by multiplying **-d** (duration) and **-n** (number_of_licenses) cannot exceed 2 147 483 647.

For example, if -n is 70 000, the maximum duration is 30 678 days (2 147 483 647/70 000).

-N *product_name*
The name of the product. It can be up to 31 characters long. If it is omitted, a product name with value NULL is created by i4lct. All product name specifications must be enclosed within double quotation marks ("***product_name***"). Product name specifications are case-sensitive.

-O Specify this option to generate, at the top of the enrollment certificate file, the command the end user issues to enroll the password. If the license is a type supported in releases of License Use Runtime earlier than Version 4.0, two commands are generated: the i4blt command for use with License Use Runtime Version 4 and the ls_admin command for use with previous releases. Otherwise, only the i4blt command is generated.

-p *product_id*
The product ID. This is an integer between 1 and 2 147 483 647 that identifies a vendor's licensed software product. Product IDs are used by the license server to

distinguish between different products from the same vendor. Product ID must be unique among all the products you create licenses for.

-P *16_bit_flag*
The *product_id* field in the password is limited to 16 bits.

-r *revision*
A string that identifies a particular version of a product; by means of version identifiers, the license server can distinguish between products that use the same product ID. It can be up to 11 characters long. If this parameter is omitted, a revision with value NULL is created by i4lct.

-R *password_registration_level*
Specifies the password registration level. Its allowed values are:

c Specify that the password is for a customer-managed use product.
v Specify that the password is for a vendor-managed use product.

Issue the `i4lct -h` command and see the *Notes:* at the end for information about the valid values of this parameter.

-s *start_date*
Specifies the start date of the password. The date format is mm/dd/yyyy. If the password type is license, this value indicates the effective start date of the licenses; if the password type is compound, this value indicates the start date at which you can create license passwords that are derived from the compound password.

To provide concurrency of licensing across the international date line, you can specify a date value of *current date − 1 day*. If you specify a date earlier than that, i4lct issues an error message and does not create a license certificate file.

The maximum start date you can specify is 4095 days from the current date.


If this option is omitted, the start date of the password defaults to the current date.

-S *derived_start_date*
Valid only with compound or compound nodelocked passwords (**-w** set to **c** or **cn**). The date format is mm/dd/yyyy. This is the derived license start date, the date before which no license password derived from the compound password is valid.

To provide concurrency of licensing across the international date line, you can specify a date value of *current date − 1 day*. If you specify a date earlier than that, i4lct issues an error message and does not create a license certificate file.


If this option is omitted, the derived start date of the password defaults to the current date.

-t *target_type*
The target type of the license server on which the licenses are to be installed. Valid values are **any, aix, dg[ux], do[main], h[pux] i[ntergraph], m[sdos], ne[xt], no[vell], os2, os2mac, sco, sgi, sun, svr4, u[ltrix], v[ms], apollo, open, sun, vax, hposf, clipper, osfl, win32, win32mac, hiux, nec**.

The **win32mac** and **os2mac** parameters specify that the target ID to be used is based on the network adapter. The **win32** and **os2** parameters specify a software-based target ID.

-T *target_id*
Specifies the target ID of the license server where the license password is to be installed. The target ID can be either the old style (32-bit) or the new style (64-bit).

If the target type, **-t**, is set to **any**, the target ID, **-T**, is set to **any** by default.

-u The upgrade flag for a custom configuration license. This flag indicates whether the customer's initial configuration and password have been modified. The replacement password is used thereafter. For concurrent network licenses, the initial password is deleted, leaving only the replacement password available. For simple nodelocked licenses, the initial password remains in the file and must not be deleted, though only the most recent replacement password is used.

-U Display the command line usage information.

-v *vendor_name*
Specifies the vendor name. It can be up to 31 characters long. All vendor name specifications must be enclosed within double quotation marks ("***vendorname***"). Vendor name specifications are case-sensitive.

If you are generating test passwords, specify the value **test**.

-V Display the i4lct version string.

-w *password_type*
The type of password to be created; supply one of the following keywords:

l Simple password
c Compound password
cn Compound nodelocked password; valid only in conjunction with the try-and-buy attribute (**-A** set to **t**)

-x *extended_target_type*
The type of target for an extended target ID. In License Use Runtime Version 4.5.x, the only valid value for **-x** is **cluster**.

-X *extended_target_id*
The ID of the extended target on which the password is to be installed. In License Use Runtime Version 4.5.x, **-X** is the ID of a cluster.

Defining Rules for Multiple-Use Concurrent Licenses

Examples

The following command creates an enrollment certificate that contains the password to test a vendor-managed use product. It represents 100 concurrent access licenses, with one year of duration, and with multiuse rules specified.

```
i4lct -i test -k test -v "test"
-N "Example Licensed Product" -p 1 -r 1.  -R v
-w l -l c -t any -a "Example Product" -s  1/ 1/1998 -d 365 -n 1   -m ug
```

The following command creates the enrollment certificate that contains the password to test a customer-managed use product with a per-seat license.

```
i4lct  -i test -k test
-v test -N "Example Licensed Product6"
-p 6 -r 1.1 -w l -l pt -a "Example Product Core Package"
-s 1/1/1998 -d 365  -t aix -T any -R c
```

| The following command creates an enrollment certificate that contains an initial custom configuration key for a nodelocked license:

```
i4lct -i 5242378dbf8d. 2.c . 9.c8.93.  .  .   -k 53989 -l n -p 5
-N "Mechanical Design" -d 73  -t aix -T 152c234 -v "Mechanical Systems"
-w l -r 1.2 -C 85AB2215691 -a "MD2"
```

| The following command creates an enrollment certificate that contains a replacement custom configuration key for the nodelocked license in the preceding example. In this example, the duration of the license is extended from the initial 730 days to 5000 days. The other values remain unchanged.

```
i4lct -i 5242378dbf8d. 2.c . 9.c8.93.  .  .   -k 53989 -l n -p 5
-N "Mechanical Design" -d 5    -t aix -T 152c234 -v "Mechanical Systems"
-w l -r 1.2 -C 85AB2215691 -a "MD2" -u
```

Defining Rules for Multiple-Use Concurrent Licenses

Multiuse rules define the conditions under which multiple invocations of a product require only a single license. These rules are applicable only to concurrent, concurrent nodelocked, and per-server licenses.

See “Multiuse Rules” on page 11 for general information about multiuse rules.

Multiple use rules are specified for individual passwords when the software vendor runs i4lct, rather than in calls from the product to the license server. This means that rules are applied to individual licenses, rather than to the product itself. (The only exception is that the product itself can override the default meaning of a multiuse rule of type **g**, or a combination that includes **g**, to put a vendor-specific rule into effect.

The vendor can therefore specify multiple use rules for each customer, without making any changes to the product itself, and without affecting other customers' licenses for the product.

Defining Rules for Multiple-Use Concurrent Licenses

The following scenarios describe how the multiuse rules work when:

A license with same group rule is installed on the server (10 licenses are available on the server).

Two clients are in the same group.

Scenario 1

1. Client1 requests 1 license; License Use Runtime shows 1 license in use
2. Client2 requests 1 license; License Use Runtime still shows 1 license in use

Scenario 2

1. Client1 requests 5 licenses; License Use Runtime shows 5 licenses in use
2. Client2 requests 2 licenses; License Use Runtime still shows 5 licenses in use

Scenario 3

1. Client1 requests 2 licenses; License Use Runtime shows 2 licenses in use
2. Client2 requests 5 licenses; License Use Runtime shows 7 licenses in use

When the second request in a scenario is higher than the first, License Use Runtime adds the requests, ignoring the multiuse rule.

Chapter 6. Hints and Tips

Read this chapter to better manage your licensing environment.

Using the Built-In Backup and Recovery Procedure

Because the breakdown of license servers may have a potentially severe impact on production, it is important to be prepared in case definitions and database files are corrupted.

The minimum backup activity the administrator should do is to keep the enrollment certificate files (or e-mail or hard copy equivalents) received from the license provider in a secure place.

License Use Runtime implements a backup procedure of all databases on license server machines.

Causes for Corrupted Definition or Database Files

There are many situations that can cause the definition or database files to become corrupted. The most common causes may be split into two groups:

NCS-related issues
License Use Runtime-related issues

NCS-Related Issues

The NCS definition and database files are static and linked to network addresses. For this reason, changing definitions or adapters within the network may lead to connection errors. The following files are used by the local location broker (*llbd*) and global location broker (*glbd*) subsystems during startup to establish connection with the network and to register objects.

The *llbd* subsystem uses the */tmp/llbddb.dat* file
The *glbd* subsystem uses the */etc/lum/ncs/glb.e* and the */etc/lum/ncs/glb.p* databases.

License Use Runtime-Related Issues

Since License Use Runtime uses the database files dynamically, any disk-related problems such as the following may cause the database files to become corrupted:

Hardware failures (media surface errors)
File-system problems (for example, file system full)
Synchronization errors during writing of data (that is, loss of electrical power)

When a License Use Runtime database is corrupted, after the database has been recovered, try to find out the real cause of the problem.

Recovery Procedure

The contents of the definition and database files used by NCS and by License Use Runtime are changed only by defined administrative commands and tools.

Automatic Backup Procedure

License Use Runtime does an automatic periodic backup on license servers by copying all files and databases

You can choose to get the backup on any other device by changing the *BackupPath* parameter in the configuration file (i4ls.ini). You can set the automatic backup to occur daily, at a certain time, (the default), or weekly, on a certain day, or at every change on the license database, according to the *BackupMode* and *BackupParm* parameters specified in the configuration file. You can also disable the automatic backup procedure by setting the *BackupMode* parameter to **none**.

For detailed information on the configuration file (i4ls.ini) see Appendix A, “License Use Runtime Configuration File” on page 173.


Be sure that the *BackupMode* and *BackupParm* parameters have the same value on all servers in the licensing environment.

The objects listed in:

```
/var/lum/scripts/db_back.sh
```

are backed up if found.

Recovery Procedure

To recover the files and databases saved with the automatic procedure described in “Automatic Backup Procedure”:

1. Identify the machine that has corrupted files or databases.
2. Stop the License Use Runtime services by issuing the following command:

   ```
   i4cfg -stop
   ```

3. Issue the following command from the appropriate directory for your operating system:

   ```
   /opt/lum/ls/os/hpux/bin/i4lmd -r
   /opt/lum/ls/os/svr4.sgi/bin/i4lmd -r
   /opt/lum/ls/os/solaris/bin/i4lmd -r
   ```

4. Start services by issuing the following command:

   ```
   i4cfg -start
   ```

This replaces the current objects with those saved with the backup procedure.

Important: In case of corruption, run this command according to the following rules:

If the BackupMode in the configuration file (i4ls.ini) is set to **changes**, run the recovery command only on the server where corruption occurred.

If the BackupMode is set to **daily** or **weekly**, first check that the backup copies have the same date on all the servers of your licensing environment, then run the backup command on *all* the servers.

Manual Backup

You can run the backup procedure manually by running:

```
/var/lum/scripts/db_back.sh
```

On a machine configured only as a network license client, only manual backup is available.

The command copies the files and databases to the backup file:

```
/tmp/iforls_bak_DATE_SERVERNAME
```

Manual Recovery

To start the recovery procedure in case of corruption, use:

```
/var/lum/scripts/db_recov.sh
```

on the failing machine. This script restores the files and databases that were saved by the db_back.sh script. Use the command:

```
/var/lum/scripts/db_recov.sh iforls_bak_DATE_SERVERNAME
```

(where iforls_bak_DATE_SERVERNAME is the name of the backup file.)

Managing the Reports Log Files

When you ask for a report, the Basic License Tool reads the current log files:

`/var/lum/logdbnn_` (network license server)
`/var/lum/crlognn_` (central registry license server)
`/var/lum/llmlgnn_` (nodelocked license server)

The names of the current log files end with an underscore.

The files `logdbnn`, `crlognn`, and `llmlgnn` contain all the collected License Use Runtime events. You can specify which events are to be collected when you configure each license server. *nn* can assume values from 00 to 99. When a file is full, a new one is started. You determine the maximum value nn can assume and the maximum size of each file by setting the *NumberOfLogFile* and *MaxLogFileSize* parameters in the configuration file, i4ls.ini. When the maximum value for nn is reached, License Use Runtime wraps to 00. The filled log files are retained so that you can archive them if you wish before the numbering wraps. For details see Appendix A, “License Use Runtime Configuration File” on page 173.

The numbering of log files starts from 00. Suppose you have the following files on the machine:

```
logdb
logdb 1_
logdb 2
```

The second file is the current, the first is the previous, and the third is the oldest.

If the current files, marked with the underscore, get too big, do not delete them. You can decrease the size of the current files with the following command:

```
i4blt -x delete_date -n server_name
```

where:

delete_date
Specifies an end date for the delete operation. All log entries recorded up to the delete date are removed from the log files. If you do not specify a date all the entries are deleted.

server_name
Specifies the license server where you want to delete the entries of logdbnn_, crlognn_, and llmlgnn_ if they exist on the server.

Managing Trace Files

Because the trace function can produce a large amount of output, it would be helpful to have a procedure to store only the most recent part of the trace. The following example procedure enables you to split the server's trace output across several files. These files can then be periodically removed in their chronological sequence, starting with the oldest files.

To start the server in trace mode and split the output across several files, issue the command:

```
print "/usr/opt/ifor/ls/os/aix/bin/i4lmd -v -z -no lcwvptms 2>&1 | split
-a3 -l 12     - /tmp/i4lmd.trc" | at now
```

The output of the license server is written to files whose names are in the format:

```
i4lmd.trcxxx
```

where *xxx* identifies a particular file in the sequence. For example, the first three files would be named `i4lmd.trcaaa`, `i4lmd.trcaab`, and `i4lmd.aac`. You can change the path and base name of the output files (`i4lmd.trc`).

To periodically remove the oldest files, set up a cron job. Use the crontab command to add to the crontab file a line similar to this:

```
 ,15,3 ,45         rm -f `ls -1r /tmp/i4lmd.trc  | tail -n +4`
```

This cron job deletes all but the three most recent trace files.

Managing Coexistence of NCS and DCE (HP-UX only)

If in your network environment you have applications, such as Directory Services and Security (DSS), that use the Distributed Computing Environment (DCE), and License Use Runtime configured in namespace binding, read this section.

The default operation of the startup process, as described in this section, will probably be appropriate if both DCE and the License Use Runtime subsystems are started at machine startup and DCE is started first. Check the file to verify that this is how your machine is configured.

Both the NCS local location broker and the DCE daemon use the same TCP/IP port number, 135, which has been assigned to them. Since the NCS local location broker can be replaced by the DCE daemon, when you start services, License Use Runtime checks whether DCE is installed before starting the local location broker. If DCE is installed, License Use Runtime checks if the DCE daemon is running. If it is not running, License Use Runtime waits for 20 seconds (default value), then, if the DCE daemon does not start, the local location broker is started. The local location broker is started if DCE is not installed or if it does not start within the 20 seconds.

If the 20-second delay is either too much or too little for your environment, open the configuration file:

```
/var/lum/i4ls.ini
```

and change the entry:

```
DCEDWAITTIME=
```

in the section:

```
[iFOR/LS NCS-Server]
```

If your machine is not configured to start the DCE daemon, and therefore you do not want any delay, change this entry to 0.

Tuning the Environment to Manage the Workload

When a high volume of client/server interactions reaches the server in a short timeframe (for example, 15 license requests per second), the server may not be able to keep up with the volume of workload. The external symptom is that the server seems to hang.

To optimize the license server daemon throughput and better balance its workload, use the environment variables:

```
PASSIVE_TIME
MAX_ACTIVITIES_THRESHOLD
MAX_ACTIVITIES
```

Tuning and Monitoring Your Environment

Experiment with the PASSIVE_TIME, MAX_ACTIVITIES, and MAX_ACTIVITIES_THRESHOLD environment variables, assigning different values to them, to find the best combination of values. Use the LOG_TRACE environment variable to enable tracing.

PASSIVE_TIME	Default value: 300 sec Minimum value: 1 sec Maximum value: 300 sec
MAX_ACTIVITIES	Default value: 512 Minimum value: 1 Maximum value: 512
MAX_ACTIVITIES_THRESHOLD	Default value: 100 Minimum value: 1 Maximum value: 100
LOG_TRACE	Default value: Not set If LOG_TRACE is set to YES, the i4lmd daemon writes the messages to stdout.

These environment variables are used whether you start the server manually or by issuing the **i4cfg -start** command.

If you start the license server by issuing the command **i4cfg -start**, and the value of the LOG_TRACE environment variable is YES, trace output is redirected to the file i4lmd.out.

Changing the Values of the Environment Variables

Each time you want to change the values of the environment variables, stop the i4lmd daemon, set the required values, and restart the i4lmd daemon. Changing environment variables while the i4lmd daemon is running has no effect.

Displaying the Trace Output on the Monitor

If you want to display the trace output on the monitor, do not start i4lmd as a subsystem, but instead follow this procedure:

1. Stop the i4lmd process
2. Set the required environment variables. For example:

```
>export MAX_ACTIVITIES=1
>export MAX_ACTIVITIES_THRESHOLD=5
>export PASSIVE_TIME=12
>export LOG_TRACE=YES
```

3. Start i4lmd:

HP-UX	`>/opt/lum/ls/os/hpux/bin/i4lmd`
IRIX	`>/opt/lum/ls/os/svr4.sgi/bin/i4lmd`
Solaris	`>/opt/lum/ls/os/solaris/bin/i4lmd`

Allowing for Log File Growth

The standard_output_file grows by about 100 bytes each time a message is logged. Its growth rate depends on the server's workload. Make sure, therefore, that the file system on which the standard_output_file is placed is large enough. For example, if you leave i4lmd running for a week, the standard_output_file will grow to about 12 MB. When the daemon is stopped and restarted, the log file is overwritten.

Removing the Log Files

You can remove the standard_error_file and standard_output_file even if i4lmd is active, but no more messages will be logged. The only way to start logging the messages again is to stop and restart the i4lmd daemon.

The Effect on Performance

Tests have indicated that the effect on i4lmd daemon performance when writing log messages to a file or to the monitor does not exceed 5%.

Measuring Performance

To measure performance, enter:

```
i4lmd -p
```

For information about this command, see “i4tv - Test Verification Tool” on page 122.

Suggested Parameter Tuning

Tune the parameters as follows:

1. Set a small PASSIVE_TIME value, so that new requests (for example, license request or license release) overwrite old requests as soon as possible.
2. Set the following environment variables:

   ```
   LOG_TRACE=YES
   MAX_ACTIVITIES_THRESHOLD=1
   ```
3. Change MAX_ACTIVITIES to find the minimum value for which the message:

   ```
   (get_activity) ... maximum of activities (value) is reached
   ```

   is not issued while the server is managing licenses. This value is the maximum activities threshold, beyond which only LicenseCheck and LicenseRelease requests will be accepted.
4. Set MAX_ACTIVITIES to *x*% more than the limit found in step 3. An initial suggested value is 20%.
5. Set MAX_ACTIVITIES_THRESHOLD=100-*x* (where x was determined in step 4).

Note: For each license server, the minimum value for:

```
MAX_ACTIVITIES    MAX_ACTIVITIES_THRESHOLD / 1
```

is limited by the minimum number of simultaneous license requests you need the server to handle. For example, if a license-enabled application starts with *X* license requests, set:

```
MAX_ACTIVITIES    MAX_ACTIVITIES_THRESHOLD / 1
```

to a value greater than or equal to *n*X* for each license server, where *n* is the number of applications that users might try to start at the same time.

Background Reference Information

Each LUM API call results in one or more client/server interactions between the calling application and one or more license server daemons (i4lmd). For each client/server interaction, the license server daemon allocates a control block in memory called an "activity block," which represents a virtual connection between the client and server.

An activity is kept in the activity pool of the license server daemon until the timeout specified in the PASSIVE_TIME parameter expires. A subsequent client/server interaction between the same client and server pair overrides a previous, unexpired activity.

When the current number of activities reaches the threshold value specified in MAX_ACTIVITIES_THRESHOLD, the following actions are performed:

All new LicenseCheck or LicenseRelease requests are accepted, and the server writes the following message to stdout:

```
(get_activity) Can allocate slot even if maximum of
activities threshold (value) is reached
```

This message means the request is being processed, but the maximum activities threshold has been reached.

All new requests other than LicenseCheck or LicenseRelease (that is LicenseRequest or requests for administrative actions) are immediately rejected, to prevent them hanging, until the end of the server timeouts. The server writes following message to stdout:

```
(get_activity) Can't allocate slot: maximum of activities
threshold (value) is reached
```

This message means the request was rejected, and the maximum activities threshold has been reached.

When the number of activities reaches the maximum value specified in MAX_ACTIVITIES, the following actions are performed:

If an old activity in Passive state can be overwritten, the request is accepted and the server writes the following message to stdout:

```
(get_activity) Can allocate slot even if maximum of
activities (value) is reached
```

```
This message means the request is being processed, but the
maximum activities value has been reached.
```

Otherwise, any kind of new request is immediately rejected and the server logs to stdout the following message:

```
(get_activity) Can't allocate slot: maximum of activities
(value) is reached
```

This message means the request was rejected, and the maximum activities value has been reached.

Managing a Custom Configuration

This section offers advice about custom configurations and their licenses.

Before Requesting a License Upgrade

Before you request an upgrade to your current custom configuration license, double-check the serial number.

Deleting Products or Reducing Numbers

When you upgrade a custom configuration, you can add products and increase the number of seats; however, you can neither delete products nor reduce the number of seats.

Deleting Keys

The initial key is always required. Do not delete it from either the network license server or the nodelock file. You can, however, delete intermediate upgrade keys from a nodelock file. (These intermediate keys are deleted automatically on network license servers.)

Chapter 7. Troubleshooting

This chapter provides suggestions for improving performance, problem determination, and debugging when using products managed with License Use Runtime. This chapter assumes you have read the preceding chapters in this book. It suggests steps you can take should certain problems occur:

At a local machine running products with nodelocked licenses
Using various types of network licenses
Running License Use Runtime and NCS subsystems
With performance
With the binding between servers and clients
With network protocols and hardware

Checking License Details

Before you proceed, be sure you know the following details about the product that is not starting properly. Check the product enrollment certificate file for all these details.

Product name (ProductName tag)

Product version (ProductVersion tag)

Vendor name (VendorName tag)

Target ID (TargetID tag)

Target type (TargetType tag)

Whether the product implements customer-managed or vendor-managed use control (RegistrationLevel tag; 1=customer-managed, 3=vendor-managed)

| Whether the product is enabled for a custom configuration policy (SerialNumber
| tag)

Whether the password is simple or compound (LicenseStyle tag=compound, or LicenseStyle tag=license type if the password is simple)

License type (LicenseStyle tag if the password is simple; DerivedLicenseStyle tag if the password is compound)

Whether the product is enabled for the hard stop/soft stop policy (SoftStop tag)

When the license becomes valid and when it expires (LicenseStartDate and LicenseEndDate tags)

Whether the password specifies a license type or a policy introduced in Version 4
| (PasswordVersion tag: 7=new in Version 4.5.5, 6=new in Version 4.5.1, 5=new in Version 4.0, 4=not new in Version 4)

For example, this is the enrollment certificate file for the DataMaster product that was used as an example in “Example 2: Managing Reservable Licenses” on page 75:

```
i4blt -a -v "'IBM Corporation' 8499f53d66dd.8d. 1.51.32.4c.  .  .   gm898vcvtpiq8"
-p "'DataMaster' '2.1a' qj4y2zjivvr9ryffuw8x9se48vvaaaa "

[LicenseCertificate]
CheckSum=7B33C C  71 128534 916679A859 54
TimeStamp=898711 18
PasswordVersion=5
VendorName=IBM Corporation
VendorPassword=gm898vcvtpiq8
VendorID=8499f53d66dd.8d. 1.51.32.4c.  .  .
ProductName=DataMaster
ProductID=2222
ProductVersion=2.1a
ProductPassword=qj4y2zjivvr9ryffuw8x9se48vvaa
ProductAnnotation=
LicenseStyle=reservable
LicenseStartDate= 6/24/1998
LicenseDuration=14436
LicenseEndDate=12/31/2 37
LicenseCount=1
MultiUseRules=
RegistrationLevel=3
TryAndBuy=No
SoftStop=No
TargetType=ANY
TargetTypeName=Open Target
TargetID=ANY
ExtendedTargetType=
ExtendedTargetID=
SerialNumber=
Upgrade=No
DerivedLicenseStyle=
DerivedLicenseStartDate=
DerivedLicenseEndDate=
DerivedLicenseAggregateDuration=
```

The i4blt command at the top of the certificate file is the command that could be used to enroll the password. In the actual enrollment certificate file it would appear on one line; here it is shown on two lines because of space constraints.

Troubleshooting Licenses (All Types)

If a user tries to start a license-enabled product and it does not start, some of the first things to check are:

First, check the product documentation.

Check to be sure the license for the application you are running is installed, and, if not, install it. See “Example 1: Managing a Licensed Product” on page 74 for information on how to install a license.

Check that the license you have installed is the correct license for the version of the software you are trying to run.

Check that the date and time on the machine are set correctly. Each license has a start date and an end date built in. If the date or time is set incorrectly on the machine where you are trying to run the product or on a license server, the license may not be recognized as active.

Check that the time zone and daylight saving time settings are correct.

Check that the start date of the enrolled license is not later than the current date, and that the license has not expired.

Troubleshooting Nodelocked Licenses

If a machine with a nodelocked license does not allow an end user to use a license-enabled product, try the suggestions in this section.

If the product uses non-runtime-based enabling:

Check that the enrollment certificate file is in the path specified by the vendor of the product and that its permissions are set so that all users can read it.

Check that the license is correctly installed in the nodelock file specified by the vendor and that its permissions are set so that all users can read it.

The default location of the nodelock file is:

```
/var/lum/nodelock
```

If the file is not in the default directory, check your product documentation or contact the product vendor.

If the product uses runtime-based enabling:

Check that the nodelocked license server (i4llmd) is up and running (see “Starting Required Subsystems” on page 161).

If the request waits for some time and then fails with error message:

```
Inter process communication failure:  check log file i4ipc.out
```

it may be that the maximum wait time for an application to receive a response from the nodelocked license server via Interprocess Communications, as specified in the configuration file, is too short. Edit the configuration file and increase the value of the ReadTimeout parameter, for example to 20:

```
ReadTimeout=2
```

Troubleshooting

Troubleshooting Network Licenses (All Types)

If a user tries to start a product with a network license and the product does not start, try the following steps. These suggestions apply to concurrent, use-once, reservable/reserved, and per-seat licenses.

Use the `i4tv` command from the client machine to verify the connection to the license server where you have the licenses installed.

- If i4tv shows no active servers, check that the network license server is running on the server machines where you have the licenses installed.
- If i4tv does display active servers, check that they include a machine where licenses for the product are installed. Use i4blt -lp to display the licenses installed on each server. You may need to reconfigure the client to connect to the proper servers.
- In direct binding, verify that the client is configured to connect to the correct servers.
- If you are using namespace binding, verify that the client is in the same cell as a server where the licenses are installed. (See “Quick Checklist” on page 163.)
- If you are using namespace binding, verify that the location brokers are running. See “Starting Required Subsystems” on page 161.
- If you are using namespace binding, use the lb_admin tool to verify that the network license server where you have the licenses installed is registered to the global location brokers.
- If you are using namespace binding with more than one global location broker, use the drm_admin tool to verify that the global location broker databases are synchronized.

If you get the error message:

```
Time disparity is too large
```

check that the date and time on the servers and client are synchronized. If server and client are in different time zones, be sure that time zone and daylight saving time have been set correctly.

If an enabled application requests more than one license to run, be sure the requested number of licenses is available on one server.

License Use Runtime does not combine licenses installed either on different servers or on the same server but with multiple enrollment actions, to satisfy the same request.

Similarly, if you received the licenses in a compound password, check that you have distributed, in one single distribution, on one server, at least the number of licenses requested. License Use Runtime does not combine licenses distributed either on different servers, or on the same server but with multiple distribution actions, to satisfy the same request.

For the same reason, if the product is enabled for soft stop, you may see soft stop licenses in use even if there are still some available licenses.

Troubleshooting Reservable and Reserved Licenses

Reservable licenses are enrolled on a network license server. When reserved they are moved to the central registry, and when granted they are moved to the nodelocked license server on the client machine. If a license has been reserved for a user but, when that user tries to use the product, it does not start:

Check that the central registry license server is up and running (see “Starting Required Subsystems” on page 161).

Check that the client machine can reach the central registry.

Check that the nodelocked license server is up and running at the client (see “Starting Required Subsystems” on page 161).

Check that the date and time set on the central registry are the same as that set on the network license client. It is possible that, according to the date and time set on the central registry, the license is not yet valid or has expired.

Double-check the name of the user, group, or node for which licenses are reserved. Be careful with leading and trailing blanks. Note that the domain is part of the node specification.

Check the local host name specification of the client machine in the `/etc/hosts` file. Make sure it is in the form:

```
ip_address  hostname_including_domain  hostname_without_domain
```

For example:

```
69.1  .67.7   lab67 7 .rome.tivoli.com  lab67 7
```

Troubleshooting Per-Server and Per-Seat Licenses

If an application with per-server licenses fails to start, be sure the nodelocked license server is running on the machine where the application server runs (see “Starting Required Subsystems” on page 161).

If an application with per-seat licenses fails to start:

- Be sure the central registry license server is up and running, and that the nodelocked license server is running on the machine where the application server runs (see “Starting Required Subsystems” on page 161).
- Be sure the per-seat license is enrolled and that per-seat licensing has been enabled (see “Example 3: Switching from Per-Server to Per-Seat Licenses” on page 76).
- Be sure the machine where the application server runs can reach the central registry.

Troubleshooting

Troubleshooting Licenses of Customer-Managed Use Products

If you are unable to enroll, update, or distribute licenses for a customer-managed product, a customer-managed use product fails to start, if soft stop does not work, or the high-water mark does not work:

Be sure that the central registry license server is up and running, and that you have defined only one central registry license server in the direct binding servers list or NCS cell (see “Starting Required Subsystems” on page 161).

Be sure the machine where you are working can reach the central registry.

If you received the licenses in a compound password, make sure you have distributed the licenses.

If soft stop does not work, be sure the soft stop policy is enabled.

Troubleshooting Licenses of Vendor-Managed Use Products

If enrollment of a vendor-managed use product fails, check that the target ID and the target type in the license match the target ID and target type of the machine where the license is installed. To get the target ID of the machine, run the i4target tool (“i4target - Target View Tool” on page 123) on that machine.

If there is a mismatch, it is possible that:

The vendor put the wrong target ID or target type into the license.
You are trying to use the license on the wrong machine.
The CPU planar of your license server machine has changed (see “Troubleshooting the Hardware” on page 166).

Troubleshooting Performance Problems

Read this section for assistance with optimizing performance.

Performance in a Direct Binding Environment

In a direct binding environment, careful configuration can help you to optimize performance. See the performance notes under “Planning Direct Binding” on page 39.

Performance in a Namespace Binding Environment

In a namespace binding environment when an NCS cell is running two or more global location brokers (GLBs), the database at each GLB node must be kept synchronized with the others, so that any GLB in the cell can satisfy a location request by a client. Occasionally, a license server is removed or is stopped without being shut down properly, with the result that invalid entries are left in the GLB databases. The invalid entries can introduce significant delays when applications attempt to get licenses, or when running the Basic License Tool.

In such situations, you can clean up the database manually (“Manual Cleanup of GLB Databases” on page 159). To schedule automatic periodic cleanup of the databases, see “Periodic Cleanup of GLB Databases” on page 160.

Manual Cleanup of GLB Databases

To do an immediate cleanup by hand, you must remove the invalid entries by using local broker administration (lb_admin) and resynchronize the GLB databases by using GLBDs replicas administration (drm_admin). Both tools are interactive. For more information on how to use these tools refer to Chapter 5, “License Use Runtime Commands” on page 81.

To remove the invalid entries, follow these steps:

1. Start the lb_admin tool at one of the GLB servers. Enter the command:
   ```
   lb_admin
   ```
2. Set the object to be worked on to be the local location broker:
   ```
   lb_admin: use local
   ```
3. Enter the clean subcommand to remove any invalid entries:
   ```
   lb_admin: clean
   ```
4. If prompted to remove entries, type y.
5. Set the object to be worked on to be the global location broker:
   ```
   lb_admin: use global
   ```
6. Use the clean subcommand to remove any invalid entries:
   ```
   lb_admin: clean
   ```
7. If prompted to remove entries, type y.
8. Exit lb_admin by using the quit subcommand:
   ```
   lb_admin: quit
   ```

To synchronize the GLB databases at all nodes, follow these steps:

1. Start the GLBD Replicas Administration tool by entering the following command:
   ```
   drm_admin
   ```
2. Set the object to be worked on to *global location broker* on your machine (replace HostName with your actual machine host name):
   ```
   drm_admin: set -o glb -h ip:HostName
   ```
3. Synchronize all the GLBs in the cell:
   ```
   drm_admin: merge_all
   ```

4. If messages inform you that a host is unreachable, remove it from the global replica list:

   ```
   drm_admin: purgerep ip:HostName
   ```

   where `HostName` is the host name of this machine that is no longer acting as a server.

   If a host machine is purged from the replica list, it should no longer be running the global location broker process (glbd). If the global location broker needs to be run on this machine at a later date, configure it and join it to the cell.

5. Synchronize all the GLBs in the cell:

   ```
   drm_admin: merge_all
   ```

6. To exit drm_admin, type the quit subcommand:

   ```
   drm_admin: quit
   ```

Periodic Cleanup of GLB Databases

An automatic periodic cleanup of stale entries in the global location broker database is set up by default. If you want to change the settings of the periodic cleanup, edit the i4ls.ini configuration file and set the values of the following tags:

SelfClean The cleanup enabling flag. Its possible values are:

yes
no

The default value is **no**.

When a network license server is heavily loaded, its performance could be severely impacted. In such situations, the i4glbcd subsystem may clean up the network license server entry in the global location broker. To prevent this, set the **SelfClean** parameter to **no**.

Frequency The number of minutes between cleanups. The allowed values are 15 to 43200. The default value is 180.

Timeout The type of timeout. Its possible values are:

long
short

The default value is **long**.

Troubleshooting Heavy Server Workloads

When a License Use Management server is stressed by a heavy workload, performance could deteriorate to the point that the server can no longer manage licenses. To avoid this situation, spread the workload over two or more servers.

Troubleshooting License Use Runtime Subsystems

This section covers problems that could arise if License Use Runtime and NCS subsystems are not started or go down.

Starting Required Subsystems

When a license-enabled product fails to start, the problem may be that a required License Use Runtime or NCS subsystem is not running.

To get a list of the License Use Runtime and NCS subsystems that are running on a machine, use the `i4cfg -list` command. The names of the subsystems are shown in Table 10. For an overview of which license servers are required for each license type, see Table 7 on page 65. In a namespace binding environment, the local location broker is required on every network license server and the central registry license server. The global location broker is required on one license server, and the global location broker database cleaner is required on one license server.

Table 10. License Use Runtime and NCS Subsystems

Subsystem	Name
Nodelocked License Server	i4llmd
Network License Server	i4lmd
Central Registry License Server	i4gdb
Local Location Broker	llbd
Global Location Broker	glbd
Global Location Broker Database Cleaner	i4glbcd

To start the subsystems, use the `i4cfg -start` command to start all subsystems configured on a machine.

If the subsystem fails to start when you issue the command, check the error messages in the i4ls.log, i4lmd.log, l4llmd.log, and l4gdb.log files in the `/var/lum` directory or the glb_log file in the `/etc/lum/ncs` directory.

Restart and Recovery

If a network license server, a nodelocked license server, or the central registry license server goes down, a record of users who currently have licenses is kept on disk. When the server is restarted, the record is reinstated and the licenses are still assigned to those users.

If you want a cold start (that is, if you want the server to restart as if it had granted no licenses before going down), use the **-c** parameter on the command used to restart the server (i4lmd, i4llmd, or i4gdb, all described in Chapter 5, “License Use Runtime Commands” on page 81). To change the default permanently to cold start, edit this parameter of the i4ls.ini file:

```
ColdStart=yes
```

and then restart services (`i4cfg -start`).

Cold start is not possible for reserved and per-seat licenses.

If the client machine goes down or the network fails, the licenses it was using become *stale* (after a check period expires, if the application is enabled using concurrent access or reservable licenses, and the application is programmed to check in with the server after a specified check period). In this case, the licenses are available to be granted to other clients. Note that those licenses will still be displayed as in use until you perform the **Clean up stale licenses** function or until a license is newly requested and none is available, in which case the server does its own cleanup of stale licenses.

The client behavior depends on the software product that is in use.

Troubleshooting Custom Configuration Licenses

Cannot Install a Custom Configuration License

If, for a custom configuration, you are unable to install a network concurrent license or nodelocked license from the certificate file:

Check the serial number.

Check whether another license with the same serial number is already installed. For a concurrent license, use i4blt. For a nodelocked license, use the nodelock file.

- If you are installing the initial key, no other key can already be installed.
- If you are installing a replacement key, another key must already be installed.

Troubleshooting Network Connections

If connections to license servers seem not to be working properly, use the i4tv (test verification) tool to verify that the license servers are up and running, or use the `i4blt -ln` command to get a list of active servers (network license servers and the central registry). For more information about these commands, see Chapter 5, “License Use Runtime Commands” on page 81.

Troubleshooting Namespace Binding

If the license server uses namespace binding, a failure in NCS can cause License Use Runtime to degrade in performance or fail altogether. It may be the case that a License Use Runtime problem is actually a problem in the state of NCS.

Under high-volume conditions, if all client machines are unable to contact a server that runs the global location broker, it is possible that the global location broker database cleaner was unable to contact the server and therefore deregistered it.

It is not necessary to have the database cleaner running on every global location broker server. It is enough to run the database cleaner on one global location broker server in the cell. Choose one that has relatively low-volume traffic, and on the others, do the following to stop the database cleaner:

1. Stop services (i4cfg -stop command).
2. Edit the i4ls.ini file and set SelfClean=no.
3. Start services (i4cfg -start command).

Quick Checklist

1. Check that the llbd subsystem is running.
2. Check that the glbd subsystem is running.
3. Check that all the system clocks specify the same time. Use the `setclock` command to synchronize all systems for a short-term solution. It is recommended to implement external time providers and a distributed time service on the network.
4. Check that the `/etc/lum/ncs/glb_obj.txt` file is the same on all machines in the NCS cell and that it has at least permission -rw-r--r 644 (all users can read it).
5. Check that the `/etc/lum/ncs/glb_site.txt` file (if any) points to one or more valid GLB hosts and that it has at least permission -rw-r--r 644 (all users can read it).
6. Check that the GLB database files still exist. Check particularly for the existence of the files `/etc/lum/ncs/glb.e` and `/etc/lum/ncs/glb.p`
7. Check whether llbd was able to create its temporary file `/tmp/llbddb.dat`.
8. Check that no more than one default cell has been defined at your location.
9. If you choose to use the default cell, be sure no other user of NCS at your location has created or might create a default cell. Since the default cell always has the same UUID, results would be unpredictable.

License Use Runtime Clients Fail to Communicate with Servers

If a client is not communicating with a server properly, it is possible that the client machine is in a different NCS cell from the license server. To put the client in the same cell as the license server, reconfigure your client machine. Refer to “Planning Namespace Binding” on page 40

It is also possible that the client machine is in a different communications subnetwork from the global location broker (GLB), and cannot contact the GLB. In this case, see “Configuring to Reach a Global Location Broker in a Different Subnetwork” on page 72.

Troubleshooting

License Use Runtime Servers Fail to Communicate with Global Location Broker

On License Use Runtime servers that run the global location broker, if the UUID stored in the file /etc/lum/ncs/glb_obj.txt is changed, the glbd subsystem continues to use the old UUID even after the glbd subsystem is stopped and restarted. The communication between the glbd subsystem and the i4lmd or i4gdb subsystems will fail.

For an example, observe the following scenario on the server `rouse`:

```
Cell UUID:
      65d6f8f6471e. 2. 9. 3. 1.45.  .  .

Content of the /etc/lum/ncs/glb_obj.txt
      657cab79f66f. 2.81.23.1c.51.  .  .
```

The i4tv command displays the following error message:

```
i4tv Version 4.5  -- LUM Test and Verification Tool
(c) Copyright 1995-1998, IBM Corporation, All Rights Reserved
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp
(c) Copyright 1991-1998, Gradient Technologies Inc, All Rights Reserved
(c) Copyright 1991,1992,1993, Hewlett-Packard Company, All Rights Reserved

?(ls_tv) request_license: No servers available for this vendor
Active License Servers:
No servers found
Active Central Registry License Server:
No Central Registry License Server found
```

This failure will occur even if there is a glbd replica on another node in the cell. The UUID is a 16-byte alphanumeric string and is hard to remember; therefore it is recommended that a copy of the current valid UUID be kept in a secure place.

The lb_find command is still able to communicate with the glbd subsystem and displays a message similar to the following:

```
# lb_find
sent to broadcast address 9.3.1.255
waiting for replies received response from glbd subsystem at
ip:rouse.itsc.austin.ibm.com(9.3.1.69)
port 1765......
replicatable   ip:rouse.itsc.austin.ibm.com   alternate_2
65d6f8f6471e. 2. 9. 3. 1.45.  .  ..
```

If the change to the file glb_obj.txt was made without the administrator's awareness, the administrator probably will not compare the displayed UUID with the UUID currently stored in the glb_obj.txt file.

The problem can be solved with the following manual steps:

1. Stop all running NCS and License Use Runtime subsystems:

   ```
   /var/lum/i4cfg -stop
   ```

2. Remove the files:

   ```
   /etc/lum/ncs/glb_log
   /etc/lum/ncs/glb.e
   /etc/lum/ncs/glb.p
   /tmp/llbddb.dat
   ```

3. To put the correct UUID into the `/etc/lum/ncs/glb_obj.txt` file, reconfigure the license server to start the global location broker or to start a replica from any other existing global location broker in the cell (if any).

4. Restart the NCS and License Use Runtime subsystems:

   ```
   /var/lum/i4cfg -start
   ```

Troubleshooting Direct Binding

If servers and clients are not communicating correctly in a direct binding environment (the i4tv command reports `No servers found`), check that exactly the same server names and direct binding port numbers were configured for each client, each network license server, and the central registry license server.

For example, if you have a server called *louise* running the network license server and the central registry license server, and clients connected to *louise*, the direct binding configuration for each machine, including *louise* itself, must contain the entries:

```
ip:louise[1 999]
ip:louise[1515]
```

Note that the port numbers must match the values of the *ipPort* and *ipGDBPort* parameters in the configuration file of *louise*. Note also that *louise* and its clients must have TCP/IP installed.

Troubleshooting TCP/IP

Following is a brief checklist to help you make sure your TCP/IP system is OK:

1. Check that the TCP/IP system is up and running.

2. Check whether IP addresses or network interfaces have been changed.

   The global location broker database may not reflect changes to the network. Use the `lb_admin` command to clean up the location broker databases, as explained in “Performance in a Namespace Binding Environment” on page 158.

3. Check whether normal TCP/IP communications are working between the nodes you want to be connected (for example, using `ping` or FTP).

4. Be sure your routing setup definition is valid.

   The `netstat` command shows the local definition. To see the hubs, use the `traceroute` command.

5. Is name resolution working?

   Name resolution is very often the reason for long startup times or many sorts of problems in large networks. Use the DNS (Domain Name System), and spend some time developing a good layout.

6. Is the MTU (Maximum Transmission Unit - Internet protocols) size equal on all hosts?

7. Is the token ring speed equal on all hosts?

8. NCS and License Use Runtime are based on universal datagram protocol (UDP). In a very highly loaded network, UDP connections may receive timeouts before data is delivered. This is normal behavior; you need to reduce the total network load.

Troubleshooting the Hardware

Following is a brief checklist to help you make sure your hardware is OK.

1. If you get the error message:

   `Invalid target ID`

   check that the CPU planar of your License Use Runtime server has not changed.

   Vendor-managed licenses on the License Use Runtime server may be tied to the CPU ID, which changes after a CPU planar swap.

2. Check that the cables are still where they should be.

3. Check whether you have reached the Ethernet length limitations on your LAN.

4. Check whether a security feature has been enabled on a router.

   Some routers allow enabling of security features. It is possible to block certain TCP/IP ports. In namespace binding, the llbd program is runs on port 135. The glbd, i4lmd, and i4gdb programs use runtime-assigned ports whose port numbers are greater than 1024. In direct binding, the ports are predefined in the configuration file.

5. Check whether any adapters or other network definitions have been changed. Because the NCS definition and database files are linked to network addresses, changes may lead to connection errors.

Collecting Error Log Data

In order to help IBM help you in problem determination, you should gather additional information to send to your IBM representative when you request support. License Use Runtime subsystems and tools can be run in traced mode as explained in the following sections.

Running Subsystems in Traced Mode

To run License Use Runtime subsystems in traced mode, follow these steps:

1. Stop all active subsystems by issuing the following command:

   ```
   /var/lum/i4cfg -stop
   ```

2. Edit the file:

   ```
   /var/lum/i4ls.ini
   ```

   and set the following tags to **yes**:

   ```
   DebugProc=yes
   DebugNCS=yes
   DebugToFile=yes
   TraceActivities=yes
   ```

3. Restart the subsystems by issuing the following command:

   ```
   /var/lum/i4cfg -start
   ```

4. Stop the subsystems again using the following command:

   ```
   /var/lum/i4cfg -stop
   ```

In the directories `/var/lum` and `/etc/lum/ncs` a file named `subsystem_name.out` is generated for each subsystem you had started. In the directory `/var/lum` the files `i4lmd.err`, `i4gdb.err`, and `i4llmd.err` are generated.

Depending on the activity performed by the subsystems, these files could become extremely large. Make sure you have enough space in the /var file system.

Running Enabled Applications in Traced Mode

To run enabled applications in traced mode set the environmental variables I4LIB_VERB and IFOR_LT_DEBUG as follows:

```
export I4LIB_VERB=Yes
export IFOR_LT_DEBUG=Yes
```

Trace messages will be displayed in the same window where you have set the variable and from which you run the application.

Running Tools in Traced Mode

To run tools in traced mode use the flag **-B** as first option when you invoke i4blt and i4cfg. The trace records will be printed into the window where you run the tools.

Collecting Other Data

Other information concerning License Use Runtime servers is automatically collected by the global location broker (glbd) and by the license server subsystems (i4llmd, i4lmd, and i4gdb). This data is stored in the following files:

```
/etc/lum/ncs/glb_log
/var/lum/i4ls.log
```

Note that most of the messages you find in these files and the related return codes are not documented.

Other files you need to provide are:

`/var/lum/i4ls.ini` (the configuration file)

`/var/lum/user_file` (the user file)

`/etc/lum/ncs/glb_obj.txt` (must be always present when the machine is part of a non-default NCS cell. Its content must be the uuid of the cell this machine belongs to, the same as the *NCSCell* keyword in the i4ls.ini file.)

`/etc/lum/ncs/glb_site.txt` (if any; a list of servers running the global location broker that this server can reach.)

License Use Runtime Databases

- License Databases

  ```
  /var/lum/licdb.dat
  /var/lum/licdb.idx
  /var/lum/llmdb.dat
  /var/lum/llmdb.idx
  /var/lum/crpdb.dat
  /var/lum/crpdb.idx
  ```

- Log Databases

  ```
  /var/lum/logdbnn_.dat
  /var/lum/logdbnn_.idx
  /var/lum/llmlgnn_.dat
  /var/lum/llmlgnn_.idx
  /var/lum/crlognn_.dat
  /var/lum/crlognn_.idx
  ```

Troubleshooting LUM Java Client Support

If you are having trouble with LUM Java Client Support:

Check the WebSphere servlet_log and error_log files in:

`/usr/lpp/IBMWebAS/logs/servlet/servletservice`

To enable native DLL plug-in logging in the `/usr/lpp/IBMWebAS/logs/native.log` file, edit the file:

`/usr/lpp/IBMWebAS/properties/server/servlet/servletservice/jvm.properties`

and change `ncf.native.logison` from `false` to `true`.

To enable Java virtual machine logging in the `/usr/lpp/IBMWebAS/logs/ncf.log` file, change both `ncf.jvm.stdoutlog.enabled` and `ncf.jvm.stdoutlog.file` from `false` to `true`.

Web Server Fails

If the Web server fails because it lacks a permission:

For Lotus Domino Go

- In the file /etc/httpd.conf, set:

  ```
  UserID=root
  GroupID=other
  ```

- In the file opt/IBMWebAS/properties/server/servlet/server.properties set:

  ```
  server.user=root
  ```

For Netscape FastTrack and Netscape Enterprise

During installation of these products set:

```
UserID to root
GroupID to
other
```

Alternatively, for either Lotus Domino Go or for Netscape FastTrack or Netscape Enterprise:

1. Create a new user name and a new group name with the required permission.
2. Set the UserID variable to the value of the new user name and the GroupID variable to the value of the new group name.

Java Program Cannot Read the User Name

If a Java applet, loaded on Netscape Communicator, cannot read the user name, install the latest version of Netscape Communicator.

Incomplete View of an Applet

If, when you run LicenseTest as an applet, you cannot see the whole applet window in your Web browser, change the window's width or height, or both. These are specified in the LicenseTest.htm file.

Installing More than One Web Server on the Same Machine

If you install more than one Web server on a machine, the first Web server creates log files in the directory .../websphere_base_directory/logs/servlet. Depending on the access permissions set for those files, any Web server you may subsequently try to start may be unable to access those log files. In this case, the new Web server cannot start.

To start the second or subsequent Web server in such circumstances:

1. Delete the log files before you start the second Web server.
2. If the second Web server still will not start:
   a. Uninstall Java Client Support.
   b. Uninstall IBM WebSphere.
   c. Delete the IBM WebSphere directory.
   d. Reinstall IBM WebSphere 1.1 or 2.0.
   e. Reinstall Java Client Support.

Installing Java Client Support after Installing a Web Server

If a Web server, its plug-in, and IBM WebSphere 1.1 or 2.0 are already installed before you install Java Client Support, and if the Web server cannot find the Java Development Kit or License Use Management Java Client Support libraries:

Step 1 Try this first:

a. Stop all Web servers.
b. Uninstall the plug-in.
c. Install Java Client Support.
d. Reinstall the plug-in.
e. Restart Web servers.

Step 2 If, after you have tried step 1, the Web server still cannot find the libraries:

a. Edit the Web server start-up script file:
   For a Netscape FastTrack Web server:

   ```
   .../webserver_administrator_base_directory/
   httpd-[machine_host_name]/start
   ```

   For a Netscape Enterprise Web server:

   ```
   .../webserver_administrator_base_directory/
   https-[machine_host_name]/start
   ```

   For a Lotus Domino Go Web server:

   ```
   /etc/rc2.d/S88go_httpd
   ```

b. Add to the top of that file the line:

   ```
   export LD_LIBRARY_PATH=newpaths
   ```

   where *newpaths* are the paths assigned, after Java Client Support has been installed, to the variable ncf.jvm.libpath in the file:

   ```
   opt/IBMWebAS/properties/server/servlet/servletservice/jvm.properties
   ```

c. Restart Web servers.

Step 3 If the Web server still cannot find the libraries, add the following line to the .profile file:

```
export PATH=newpaths:$PATH
```

where *newpaths* are the paths assigned, after Java Client Support has been installed, to the variable ncf.jvm.libpath in the file:

```
opt/IBMWebAS/properties/server/servlet/servletservice/jvm.properties
```

Appendix A. License Use Runtime Configuration File

This appendix describes the License Use Runtime i4ls.ini configuration file. The file is located in the `/var/lum/` directory. You should normally use the configuration tool to configure License Use Runtime. In case you have no access to the configuration tool or you want to change just a few parameters of your configuration, the information in this appendix will enable you to modify the parameters by editing the file. Also, some parameters (designated in this appendix) can be changed *only* by editing the configuration file.

If a parameter has a default value, it is shown with the parameter name (for example, **BackupMode=daily**).

[iFOR/LS Machine-Configuration]

ConfigureAs=client
Not used on the HP-UX, IRIX, or Solaris platform.

Transport=tcpip
Specifies the transport protocol used in License Use Runtime client-server communications. The only possible value in HP-UX, IRIX, and Solaris is **tcpip**.

MachineName=
Not used on the HP-UX, IRIX, or Solaris platform.

NCSCell=333b91c50000.0d.00.00.87.84.00.00.00
The NCS uuid of the cell this machine belongs to. If you are configuring the machine as a network license client only, just specify the NCS cell you want to join. If you are configuring a network license server as a GLB replica, specify the NCS cell you want your server to join. The keyword **CreateFrom** must be set to the *ip:servername* of any of the replicable GLB replicas of the cell. If you are configuring a network license server as a first GLB, specify the NCS cell uuid of the cell you are creating (the keyword **Create** must be set to **new**). The uuid specified here must be the same as that specified in the /etc/lum/ncs/glb_obj.txt file, if the file exists. In the case of the default cell, there must not be a /etc/lum/ncs/glb_obj.txt file.

UserName=
Not used on the HP-UX, IRIX, or Solaris platform.

GroupName=
Not used on the HP-UX, IRIX, or Solaris platform.

DebugProc=no
Specifies whether or not the License Use Runtime subsystems must be started in debug traced mode. Possible values are **yes** and **no**. This parameter can be changed only by editing the configuration file.

Configuration File

DebugNCS=no
Specifies whether or not the License Use Runtime subsystems must be started in debug traced mode and additional communication-related information collected, and whether or not the NCS subsystems must be started in debug traced mode. Possible values are **yes** and **no**. This parameter can be changed only by editing the configuration file.

DebugToFile=no
Not used on the HP-UX, IRIX, or Solaris platform.

ConcurrentNodelock=
Not used on the HP-UX, IRIX, or Solaris platform.

LogLevel=
Not used on the HP-UX, IRIX, or Solaris platform.

LogMsgsMaxNum=
Not used on the HP-UX, IRIX, or Solaris platform.

LogFile=
Not used on the HP-UX, IRIX, or Solaris platform.

CommunVersion=V4R5
Version of License Use Runtime communication subcomponent.

RuntimeVersion=V4R5
Version of License Use Runtime runtime subcomponent.

NCSSupportVersion=V4R5
Version of License Use Runtime namespace binding support subcomponent.

Communication=yes
The machine is configured to communicate in a network.

NamespaceBindingSupport=yes
Namespace binding support is configured on the machine.

AdvancedConfiguration=no
The user selected **Advanced Configuration** when configuring the machine.

[iFOR/LS GLBD-Configuration]

Create=new
Whether or not the started GLB is the first one in the cell or one of the possible subsequent GLBs replicas. Possible values are **new** and **replicate**.

CreateFrom=
If you are configuring as a GLB replica, the tcpip *ip:servername* of any of the replicable GLBs of the cell. Also specify the associated NCS cell UUID in the **NCSCell** keyword.

Family=ip

Transport protocol used between GLB replicas of the same cell. The only possible value in HP-UX, IRIX, and Solaris is **ip**.

DefaultCell=yes

Whether or not you are starting the new GLB in a default NCS cell. If you do, make sure you also specified the default UUID in the **NCSCell** keyword and the **new** value in the **Create** keyword, and do not create the /etc/lum/ncs/glb_obj.txt file.

SelfClean=no

Whether of not you want an automatic periodic cleaning of the location broker's database. This parameter can be changed only by editing the configuration file.

Frequency=180

The frequency in minutes of the automatic periodic cleaning of the location broker's database. This parameter can be changed only by editing the configuration file.

Timeout=long

The timeout used to make sure the license server is alive in the automatic periodic cleaning of the location broker's database. Possible values are **long** and **short**. This parameter can be changed only by editing the configuration file.

[iFOR/LS LMD]

BackupMode=daily

The mode of the License Use Runtime database backup procedure. Possible values are:

daily The backup is started at the time specified in the BackupParm parameter.

weekly The backup is started at approximately midnight (00:00) of the day specified in the BackupParm parameter.

changes The backup is made each time the database is changed, such as when an object is added or deleted.

This parameter value must be the same on all servers within your licensing environment. This parameter can be changed only by editing the configuration file.

BackupParm=0

If **BackupMode** is **daily**, the hour when the backup occurs (midnight=0). If **BackupMode** is **weekly**, the day of the week when the backup occurs (Sunday=0).

This parameter value must be the same on all servers within your licensing environment. This parameter can be changed only by editing the configuration file.

Configuration File

BackupPath=/tmp
The path where the server files and databases are copied during the automatic backup procedure. This parameter can be changed only by editing the configuration file.

NumberOfLogFile=2
The number of log files License Use Runtime writes. For example, if logdb is the log file name, and **NumberOfLogFile** is set to 2, License Use Runtime changes the name to logdb00_. When it is full, it starts logging events on logdb01_. When this is full, it restarts writing on logdb00_. This parameter can be changed only by editing the configuration file.

MaxLogFileSize=10
The maximum length of the log files, in tens of kilobytes. After that size is reached, License Use Runtime starts writing on another log file. This parameter can be changed only by editing the configuration file.

ValidityPeriod=15
Internal period, in days, to validate per-seat licenses stored on the nodelocked license server against the central registry. This parameter can be changed only by editing the configuration file.

HALFrequency=30
The length, in seconds, of the interval at which servers in a cluster synchronize data among themselves. You can increase this number if you have performance problems, but doing so delays synchronization between members of a cluster.

[iFOR/LS NCS-Server]

llbd=no
Whether or not you want to start the local location broker subsystem on this server and have the License Use Runtime subsystem use it. Possible values are **yes** and **no**. The **llbd** and **glbd** parameters must always be set to the same value.

glbd=no
Whether or not you want the network and central registry license servers running on this machine to register themselves into the global location broker database. Possible values are **yes** and **no**. By specifying **no**, you disable namespace binding support on this server; it will support only clients locating the server in direct binding mode. The **llbd** and **glbd** parameters must always be set to the same value.

ipPort=1515
The TCP/IP port number the license server listens to when supporting its clients.

ipGDBPort=10999
The TCP/IP port number the central registry license server listens to when supporting its clients.

ipNDLPort=12999
The TCP/IP port number the nodelocked license server listens to for remote administration.

ipHALPort=11999
The TCP/IP port number used for internal communication between by servers in a cluster. Change this number only if 11999 is already used for some other purpose. If you change this value, change it on cluster members.

netbiosPort=115
Not used on the HP-UX, IRIX, or Solaris platform.

netbiosGDBPort=109
Not used on the HP-UX, IRIX, or Solaris platform.

netbiosNDLPort=12999
Not used on the HP-UX, IRIX, or Solaris platform.

ipxPort=1515
Not used on the HP-UX, IRIX, or Solaris platform.

ipxGDBPort=10999
Not used on the HP-UX, IRIX, or Solaris platform.

ipxNDLPort=12999
Not used on the HP-UX, IRIX, or Solaris platform.

RunGLBD=no
Whether the global location broker subsystem is to be started on this machine. Possible values are **yes** and **no**.

RunGDB=no
Whether the central registry license server is to be started on this machine. Possible values are **yes** and **no**.

DisableRemoteAdmin=no
Whether or not the administration of this network license server is to be disabled when using the administration tool started on a different server. Possible values are **yes** and **no**.

DisableRemoteNDLAdmin=yes
Whether or not the administration of this nodelocked license server is to be disabled when using the administration tool started on a different server. Possible values are **yes** and **no**.

LogAllEvents=no
Whether or not all the events are to be logged on the license servers. Possible values are **yes** and **no**.

LogFile=
Not used on the HP-UX, IRIX, or Solaris platform.

LogPath=/var/lum
Log file path of the license server subsystem log.

Configuration File

ColdStart=no
Whether the license servers restart from scratch, with no record of licenses in use granted before stopping (**yes**), or not (**no**). Cold start is not possible for reserved and per-seat licenses. This parameter can be changed only by editing the configuration file.

DCEDWAITTIME=20
The maximum number of seconds to wait for the dce daemon to start in place of the llbd subsystem. During i4cfg -start, if the dce is installed but not running after this number of seconds, the llbd subsystem is started. This parameter can be changed only by editing the configuration file.

RunNDL=yes
Whether the nodelocked license server is to be started on this machine. Possible values are **yes** and **no**.

RunLMD=no
Whether the network license server is to be started on this machine. Possible values are **yes** and **no**.

UseHostTable=no
Not used on the HP-UX, IRIX, or Solaris platform.

PassiveTime=300
Not used on the HP-UX, IRIX, or Solaris platform.

MaxActivities=512
Not used on the HP-UX, IRIX, or Solaris platform.

MaxActivitiesThreshold=100
Not used on the HP-UX, IRIX, or Solaris platform.

TraceActivities=no
Not used on the HP-UX, IRIX, or Solaris platform.

[iFOR/LS Server Logging]

LogGrant=no
Log when a license was granted or released. Possible values are **yes** and **no**.

LogCheckin=no
Log when a licensed product has sent a check-in call to the server to notify it that the product is running. Possible values are **yes** and **no**.

LogWait=no
Log when a license request cannot be satisfied because no licenses are available, and the user is added to a queue. Possible values are **yes** and **no**.

LogVendor=yes
Log when a new vendor was added or deleted Possible values are **yes** and **no**.

LogProduct=yes
Log when a product of a new vendor was registered or deleted. Possible values are **yes** and **no**.

LogTimeout=no
Log when the server has canceled the request for a license because the check period has expired. Possible values are **yes** and **no**.

LogErrors=yes
Log server errors that do not stop the server, but return a status code and a message. Possible values are **yes** and **no**.

LogVendorMsg=yes
Log error messages the vendor inserted in the product. Possible values are **yes** and **no**.

LogSvrStartStop=no
Log the successful start or stop of the license server. Possible values are **yes** and **no**.

[iFOR/LS NetBIOS-Configuration]

LanAdaptor=
Not used on the HP-UX, IRIX, or Solaris platform.

NCBS=
Not used on the HP-UX, IRIX, or Solaris platform.

HasOS2Clients=
Not used on the HP-UX, IRIX, or Solaris platform.

[iFOR/LS Client]

Threshold_Level=
Not used on the HP-UX, IRIX, or Solaris platform.

Threshold_Automatic=0
Not used on the HP-UX, IRIX, or Solaris platform.

Threshold_Frequency=60
Not used on the HP-UX, IRIX, or Solaris platform.

Refresh_Automatic=
Not used on the HP-UX, IRIX, or Solaris platform.

Refresh_Frequency=
Not used on the HP-UX, IRIX, or Solaris platform.

ReadTimeout=2
The maximum wait time, in seconds, for an application to receive a response from the nodelocked license server via Interprocess Communications. The minimum is 1 and the maximum is 60. You may need to increase this value if performance on your machine is poor. This parameter can be changed only by editing the configuration file.

Configuration File

[iFOR/LS NCS-Client]

UseDirectBindingOnly=no
Whether or not the client licensed applications running on this machine are to locate the license servers using direct binding only. The administration tool is considered a client application.

FilterNDL=no
Whether or not nodelocked licenses are to be excluded from the set of licenses administered by the Basic License Tool.

FilterNet=no
Whether or not network licenses are to be excluded from the set of licenses administered by the Basic License Tool.

NumDirectBindServers=3
The number of direct binding servers the client applications are configured to point to directly, using just name and port number. Specify the **DirectBindServer** keyword for each server the client points to. If you need to contact the central registry license server, there must also be an entry for it. The default ip port numbers are 1515 for the license server and 10999 for the administration server. Make sure you insert the correct ones if you are not using the defaults.

DirectBindServer1=ip:thelma.rnsl.ibm.com[1515]
The format is *ip:servername[port]*.

DirectBindServer2=ip:louise.rnsl.ibm.com[1515]
The format is *ip:servername[port]*.

DirectBindServer3=ip:louise.rnsl.ibm.com[10999]
The format is *ip:servername[port]*.

OS2NumServers=0
Not used on the HP-UX, IRIX, or Solaris platform.

OS2NetbiosServer1=no
Not used on the HP-UX, IRIX, or Solaris platform.

GDBServer=ip:louise.rnsl.ibm.com[10999]
The format is *ip:servername[port]*.

NumDirectBindNDLServers=2
The number of nodelocked license servers whose licenses can be administered remotely from this machine.

DirectBindNDLServer1=ip:lab68082.rome.lab.tivolicom[12999]
The format is *ip:servername[port]*.

DirectBindNDLServer2=ip:lab68084.rome.lab.tivolicom[12999]
The format is *ip:servername[port]*.

Appendix B. Using the Nodelock File

This appendix explains how to prepare the nodelock file manually and how to use it. You might need to modify the nodelock file for configurations without a nodelocked license server or for backward compatibility.

To prepare and use the nodelock file:

1. Log in as `root`, or use the **su** command.
2. Create or edit the file `/var/lum/nodelock`

   The format of the nodelock file is:

   ```
   # comment

   vendorID  productPassword  Annotation  version  [serialNumber]
   ```

   where:

   The first line starts with a comment character, #, and is included for information only. It indicates the product name and license expiry date.

   The second line is the product license. Its fields and their content are as follows:

   vendorID The vendor ID.

   productPassword The long alphanumeric password that enables the nodelock license.

   Annotation The annotation field, which is used by the application developer to provide any unique enablement options of the license. This optional field, which is set to null ("") in the example, can contain up to 80 alphanumeric characters.

   version The version number of the product.

   serialNumber The serial number of a custom configuration license. This field can contain up to 31 alphanumeric characters.

   Initially, this file could have entries similar to the following lines:

   ```
   # nodelock example for the licensed product expires 12/25/2  3

   543b f87c 93. 2.81.87.92.34.  .  .   gganccupqb5dauxabdw "" "2. " "85AB2215691"
   ```

3. To help yourself and others identify the license in the future, because there may be other nodelocked software on the same computer, you should enter a comment above the license. That comment should include the full product name, version, and any expiration date.
4. Double-check the information to ensure that it is the same as that supplied by the vendor.
5. Test the product.

Appendix C. Features and Functions Added in Version 4

Table 11 lists the features and functions that have been added to License Use Management in Version 4.

Attention: Do not use obsolete commands or APIs, which are supported for backward compatibility, with these newer features and functions.

Table 11. Features and Functions Added in Version 4

	Feature or Function	Page
License types	Concurrent nodelock	8
	Use-once nodelock	8
	Per server	9
	Reservable	10
	Per seat	11
License policies	Try and buy	11
	Custom configuration	13
Server enhancements	Hard-stop/soft-stop selection	13
	Customer-managed use control	7
	High-water mark	17
	Threshold	17
	High-availability licensing	19
NCS enhancement	Direct binding	38

Glossary

A

ACID. See *application client identifier.*

application client identifier. In License Use Management, the unique identifier of the application client. When a license is granted to a client, the ACID of the client is recorded in the central registry, which is checked at any new license request. This avoids granting a license twice to the same application client.

administrator. In License Use Management, the person who is responsible for setting up the License Use Runtime environment. The tasks of the administrator include:

- Installing and configuring nodelocked license servers, network license servers, network license clients, and the central registry.
- Installing the software product licenses on the servers.
- Monitoring the software products use through the Basic License Tool.
- Configuring the network.

application client. A computer that runs a software product and plays the role of the client in the traditional client-server model.

application server. A computer where an enabled product is installed, which provides shared access to the product to workstations (the application clients) over the network.

In License Use Management, the application server is the License Use Runtime client. It requests the licenses for all its application clients.

annotation. See *license annotation.*

B

Basic License Tool. In License Use Management, the administration tool included in License Use Runtime, which enables the administrator to add or delete licenses from the server database, display the licenses installed, distribute the licenses among the servers available on the network, and generate reports on license usage and server events.

binding. In License Use Management, one of two methods by which a network license client can locate a server in order to request a license. See *direct binding* and *namespace binding*.

C

cell. See *NCS cell.*

central registry. In License Use Management, a database that contains information about:

- The enrollment and distribution of customer-managed use control products.
- Which application clients already have a per-seat license.
- Reservation of reservable licenses.

check period. In License Use Management, a time period during which a product holding a concurrent or unreserved reservable license must check in with the network license server. If the product does not check in during this period, the network license server assumes that the product is not running, and may release a granted license to another user.

cluster. In License Use Management, a group of network license servers that jointly serve vendor-managed concurrent licenses that are tied to the cluster rather than to an individual server. While some servers in the cluster are serving licenses, one or more servers remain in reserve, ready to take over should an active server fail.

compound password. In License Use Management, a password from which it is possible to extract multiple simple passwords, each representing one or more licenses.

Enabled applications cannot use the compound password directly.

concurrent license. In License Use Management, a type of license, administered by the network license server, that can be used by different users from any node that is connected to a network license server. Concurrent licenses enable as many users to use a particular software product concurrently as there are licenses.

concurrent nodelocked license. In License Use Management, a nodelocked license that allows a limited number of concurrent uses of the licensed product on the node where the license is installed. Concurrent nodelocked licenses enable as many concurrent uses of a particular software product as there are licenses.

custom configuration. A selected combination of products, tailored by a vendor to the needs of one or more users. Each custom configuration is identified by unique serial number, which is incorporated into the custom configuration license.

custom configuration license. A special case of either a concurrent network license or a simple nodelocked license that contains a unique serial number identifying a custom configuration. See also license.

customer-managed use control. In License Use Management, a level of password use control in which the customer manages compliance with the terms of the software product acquisition. It is the customer's responsibility to set the upper limit on the number of licenses that can be extracted and distributed, based on the terms of the software product acquisition.

D

default NCS cell. A cell that is identified by the default GLB object UUID. Machines in the default cell do not have the *glb_obj.txt* file.

direct binding. In License Use Management, a type of binding between network license servers and clients in which client applications locate license servers by means of a local text file that contains network addresses of the license servers.

direct binding servers list. In License Use Management, a set of network license servers and a central registry license server that collectively serve a set of network license clients.

dynamic nodelocking. In License Use Management, a way of using licensing APIs in which a compound password installed on a network license server carries simple nodelocked licenses. Upon first invocation of the product at a client, a simple nodelocked license is extracted from the compound password and installed on the client machine.

E

end user. In License Use Management, a user of license-enabled software products. The tasks of the end users may include:

Installing License Use Runtime with the help of the administrator.
Configuring License Use Runtime as a network license client.

enrollment certificate. In License Use Management, a mechanism for the distribution of licenses to end users. It is usually in the form of an electronic file, and contains all the information that is related to the licenses acquired for a license-enabled product.

G

gdb server. See *central registry.*

GLB. See *global location broker.*

glbd replica. In License Use Management, a copy, on a newly configured network license server, of a global location broker database that already exists on another server.

glb_obj.txt. A file that specifies the object UUID of the global location broker. The glb_obj.txt file makes it possible to override the default value by specifying a different GLB object UUID for a particular machine. The glb_obj.txt file is used only in special configurations that require several disjoint GLB databases (each of which is possibly replicated). In most networks and internets, there is only one GLB database (possibly replicated), and machines do not need to have a glb_obj.txt file. If a machine has a glb_obj.txt file, the UUID in the file identifies the GLB object to which that machine directs lookups and updates.

global location broker. Part of the Network Computing System (NCS) that enables clients to locate servers in a network or internet. It is a process that manages a database that stores the locations (network addresses and port numbers) where server processes are running. The global location broker process maintains this database and provides access to it.

H

hard stop. In License Use Management, a policy according to which, if the end user starts the product and there are no licenses available, the product does not start.

high-availability licensing. In License Use Management, an option that makes it possible for a cluster of network license servers to jointly serve concurrent licenses, with one or more servers in reserve in case a server goes down. The software vendor must create passwords to be enrolled on the cluster rather than on an individual server.

high-water mark. In License Use Management, the maximum number of soft stop licenses that have been granted for a given product, over the number of licenses enrolled for that product. It is updated when the soft stop policy is set. In hard stop policy no updating of the high-water mark occurs, since it is assumed that the product stops its execution if no licenses are available.

I

internet. A set of two or more connected networks. The networks in an internet do not necessarily use the same communications protocol.

License Use Runtime supports the following protocols on OS/2:

NetBIOS
TCP/IP
IPX

License Use Runtime supports the following protocols on Windows NT:

NetBIOS
TCP/IP
IPX

On Windows 95 and Windows 98, NetBIOS is not supported. On Windows 98 and Windows NT Alpha, IPX is not supported.

On AIX, HP-UX, IRIX, Solaris, and Windows NT Alpha License Use Runtime supports only TCP/IP.

initial key. A license key for a custom configuration license generated without using the Upgrade flag. It is an encrypted character string that specifies some terms of the acquisition of the selected combination of software products in a customer's initial custom configuration. Contrast with replacement key.

IPX. A communication protocol that creates, maintains, and terminates connections among network devices (workstations, file servers, or routers, for example).

J

Java. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

JavaBeans. The platform-independent, component architecture for the Java programming language. JavaBeans enables software developers to assemble pieces of Java code (“Beans”) into a graphical drag-and-drop development environment.

K

key. See *password.*

L

license. Permission to use an instance of a licensed software product or service, according to the basis on which the vendor charges for the product or service. Sometimes, a user needs more than one license to make full use of a particular product features.

The term *license* as used in the context of License Use Management does not refer to the license agreement that governs use of and rights to a product.

license annotation. A string that the vendor can use to modify the use of a license.

license database. In License Use Management, the database of licenses that a license server maintains.

license-enabled product. A product that is enabled for license use management.

A vendor provides a license-enabled product together with a password that authorizes use of the product. The password contains an encryption of certain terms of the acquisition of the product (such as how many licenses the customer can use, the expiration date of the licenses, and the type of license).

license information. In License Use Management, the information that describes licenses. This information consists of product name, product version, number of

licenses, license type, start and end dates for the licenses, and a time stamp.

license key. See *password.*

license password. See *password.*

licensed product. See *license-enabled product.*

license server. A program that provides the license services, administering licenses for software products. It may be a network license server or a nodelocked license server.

local location broker. Part of the network computing system (NCS). It manages the local location broker (LLB) database, which stores information about NCS-based server programs that run on the local host.

location broker. See *local location broker* and *global location broker.*

log file. A database that records messages and errors from the license server, and sometimes from licensed products as well.

M

multiuse rules. In License Use Management, rules that define the conditions under which multiple invocations of a product require only a single license. These rules are applicable only to concurrent access, concurrent nodelocked, and per-server licenses. The vendor of the product defines multiuse rules.

N

namespace binding. In License Use Management, a binding mechanism in which the network license servers register themselves with the global location broker, which locates an appropriate license server when a client requests a license. Namespace binding is not available on Windows platforms.

NCS. A set of software components, developed by Apollo Computer Inc., that conform to the Network Computing Architecture. These components include the Remote Procedure Call (RPC) runtime library and the Location Brokers.

NCS cell. A logical grouping of clients and servers; a subset of a network. Machines in one cell cannot communicate with machines in other cells. Machines cannot be in more than one cell at a time. Machines in the same cell are identified by the same global location broker (GLB) object Universal Unique Identifier (UUID).

network. A group of nodes and the links that interconnect them.

network license. In License Use Management, a license that is maintained on a network license server for use upon request by a License Use Runtime client.

network license client. In License Use Management, a node configured to make use of licenses by requesting them from a network license server.

network licensed product. In License Use Management, a licensed product that is enabled such that the licenses are maintained on a server for use upon request by a License Use Runtime client.

network license server. In License Use Management, a node in the network on which network licenses are stored for use by License Use Runtime clients.

node. A machine in the network. In License Use Management, it can be configured as a nodelocked license server, a network license client, a network license server, the central registry license server, or a combination

nodelocked license. In License Use Management, a type of license locked to a specific node, so that the product can be used only at that node. The nodelocked license is installed on the machine for which it was created.

nodelocked license server. In License Use Management, a server on a node that manages nodelocked licenses on that node.

non-runtime-based enablement. In License Use Management, a type of license enablement for a product with simple nodelocked licenses that does not make use of License Use Runtime on the end user's machine. The password is stored in a special file when the enabled product is installed. When the enabled product is started, it checks the file to ensure that there is a valid license.

O

object. In the Network Computing System, an entity that is manipulated by well-defined operations. Databases, files, directories, devices, processes, and processors are all objects.

P

password. An encrypted character string that specifies some terms of the acquisition of a software product. See also *simple password, compound password.*

password use control level. In License Use Management, a level of control of compliance with the terms of the acquisition of a license-enabled product. The password use control levels are:

customer-managed use control
vendor-managed use control

per-seat license. In License Use Management, a license used to enable client/server applications that are constructed for multiple-server solutions. Assignment of a per-seat license to an application client is permanent. Unused application client licenses are kept in a central repository, which all the application servers share. They also share a central list of application clients that have an assigned license. If an application client connects to multiple application servers, only one license is assigned to it.

per-server license. In License Use Management, a license used to enable client/server applications that are constructed for multiple-server solutions. Each server license is associated with a specific number of clients. This number represents the maximum number of clients that may concurrently request that server application services at any given time. Assignment of a per-seat license to an application client is temporary. If an application client connects to multiple application servers at the same time, it is assigned more than one license.

product ID. In License Use Management, a number that identifies a vendor licensed software product. By means of product IDs, the license server can distinguish between products from the same vendor.

Q

queue. In License Use Management, a sequence of users who are waiting for a concurrent license to become available so they can run a product. The administrator can monitor the number of users in queue through the Basic License Tool.

R

replacement key. A license key for a custom configuration license generated using the Upgrade flag. It is an encrypted character string that specifies some terms of the acquisition of the selected combination of software products in a customer's upgraded custom configuration. Contrast with initial key.

replica. See *glbd replica.*

report. In License Use Management, a summary of the events related to the licenses that are installed on the selected servers, filtered as the administrator specified. Examples of events are:

Requests for licenses for a product in a given interval of time.

Server startup.

reservable license. In License Use Management, a network license that the administrator can reserve for the exclusive use of a user, a group, or a node. The reservation is for a specified time period.

reserved license. In License Use Management, a license that the administrator has reserved for the exclusive use of a user, a group, or a node.

runtime-based enablement. In License Use Management, a type of license enablement for a product with nodelocked licenses that uses License Use Runtime on the end user's machine to manage the licenses.

S

selected servers. In License Use Management, the servers that the administrator is working with through the Basic License Tool. All the products whose licenses are installed on the selected servers are displayed in the Basic License Tool main window.

simple nodelocked license. In License Use Management, a nodelocked license that allows an

unlimited number of simultaneous uses of the licensed application on the local machine.

simple password. In License Use Management, a password that, once enrolled on a license server, represents one or more licenses.

Enabled applications can use the simple password directly.

socket server. The process that allows License Use Management Runtime clients and servers to communicate among themselves through the NetBIOS protocol.

soft stop. A policy according to which, if the end user starts the product and there are no licenses available, the product starts.

T

target. In License Use Management, the node at which a password is to be installed. If the password specifies a nodelocked license, the target is the node where the licensed product is run. If the password specifies multiple nodelocked license (that is, a compound password for nodelocked licenses) or network licenses, the target is a node at which the network license server (i4lmd) is running.

target ID. In License Use Management, a unique identifier of a node. A vendor can generate a password that can be installed only on a node that has a specific target ID. The target ID can be based on hardware or generated by License Use Runtime.

TCP/IP. Transmission Control Protocol/Internet Protocol. A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

threshold. In License Use Management, a percentage of licenses; if more than this percentage of licenses for a product are in use, messages about the level of use are logged.

time stamp. In License Use Management, a number that identifies the date and time at which a set of licenses was created.

try-and-buy license. In License Use Management, a nodelocked license that has a fixed duration and a start date equal to the date when the license is enrolled. A try-and-buy license is made available for purposes of evaluating the application, and can be replaced by a production license after evaluation.

U

universal unique identifier. An identifier that is used by NCS to identify interfaces, objects, and types.

use-once license. In License Use Management, a type of license, administered by the license server, that is effective for only a single instance of starting a product or using a service. The license server decrements the number of available use-once licenses each time the product or service is used.

use-once nodelocked license. In License Use Management, a license that is valid for only a single instance of starting a product or using a service, on the node where the license is installed.

user file. In License Use Management, a flat ASCII file, which the administrator creates with a text editor, that lists users who specifically are or are not allowed to use specified products.

UUID. See *universal unique identifier.*

V

vendor ID. In License Use Management, the identifier of a vendor of licensed products. Vendor IDs are a License Use Runtime specific usage of NCS Universal Unique Identifiers (UUIDs).

vendor-managed use control. In License Use Management, a level of password use control in which the vendor manages compliance with the terms of the software product acquisition.

The customer of a vendor-managed use product supplies a unique identifier (target ID) of each machine where product licenses are to be installed. The vendor uses this information to create the password, which is tied to the target workstation and cannot be used on another workstation.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

IBM

Program Number: 5697-D34
5697-D35
5697-D36

Printed in Denmark by IBM Danmark A/S

SH19-4361- 2