


AIX Version 4.1

iFOR/LS System Management Guide

Printed in the U.S.A.

 Place Form Number barcode
lower left corner here.

First Edition (August 1994)

This edition of *AIX Version 4.1 iFOR/LS System Management Guide* applies to AIX Version 4.1 and to all subsequent releases of this product until otherwise indicated in new releases or technical newsletters.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: THIS MANUAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

It is not warranted that the contents of this publication or the accompanying source code examples, whether individually or as one or more groups, will meet your requirements or that the publication or the accompanying source code examples are error-free.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication.

It is possible that this publication may contain references to, or information about, products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that such products, programming, or services will be offered in your country. Any reference to a licensed program in this publication is not intended to state or imply that you can use only the licensed program instead. You can use any functionally equivalent program instead.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9630, 11400 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

© Copyright Gradient Technologies Inc., 1992.

© Copyright International Business Machines Corporation 1994. All rights reserved.

Notice to U.S. Government Users — Documentation Related to Restricted Rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract.

Reader's Comment Form

AIX Version 4.1 iFOR/LS System Management Guide

SC23-2665-00

Please use this form only to identify publication errors or to request changes in publications. Your comments assist us in improving our publications. Direct any requests for additional publications, technical questions about systems, changes in programming support, and so on, to your sales representative or to your dealer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that we may use or distribute whatever information you supply in any way we believe appropriate without incurring any obligation to you.

Note: To send comments electronically, use this commercial Internet address: `aix6kpub@austin.ibm.com`.

- If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

- If you would like a reply, check this box. Be sure to print your name and address below.

Page	Comments

Please contact your sales representative or dealer to request additional publications.

Please print

Date _____

Your Name _____

Company Name _____

Mailing Address _____

Phone No. () _____
Area Code

Tape

Please Do Not Staple

Tape

Fold

Fold

Cut or Fold Along Line

Publications Department
Department 997, Internal Zip 9630
11400 Burnet Rd.
Austin, Texas 78758-3493

Fold

Fold

Tape

Please Do Not Staple

Tape

Trademarks and Acknowledgements

AIX and AIX/6000 are trademarks of International Business Machines Corporation.

AIXwindows is a trademark of International Business Machines Corporation.

HP and Hewlett-Packard are trademarks of Hewlett-Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

iFOR is a trademark of Gradient Technologies, Inc.

InfoExplorer is a trademark of International Business Machines Corporation.

Gradient is a trademark of Gradient Technologies, Inc.

NetLS and Network Licensing System are trademarks of Apollo Computer, Inc., a subsidiary of Hewlett-Packard Co.

Network Computing System is a trademark of Apollo Computer, Inc.

RISC System/6000 is a trademark of International Business Machines Corporation.

UNIX is a registered trademark licensed exclusively by X/Open Company.

Contents

Chapter 1. Introduction to iFOR/LS	1-1
iFOR/LS Enablement for the AIX Version 4.1 Base Operating System	1-1
Types of iFOR/LS Licenses	1-1
Nodelocked License	1-2
Concurrent-Use Licenses	1-3
Use-Once Licenses	1-3
Vendor-Specific Administrative Considerations	1-3
License Annotation	1-4
License Passwords	1-4
The Vendor Key	1-4
License Databases	1-5
Components of iFOR/LS	1-5
A Walkthrough of iFOR/LS	1-5
Distributed Computing with NCS	1-6
iFOR/LS Platforms	1-8
Chapter 2: Installing and Configuring iFOR/LS	2-1
Requirements and Sample Scenarios	2-1
Case 1: Standalone Machine Running AIX	2-1
Case 2: Standalone Machine Running AIX and CSet++	2-1
Case 3: Networked Client Running AIX and CSet++ (iFOR/LS Server Is on Another Machine)	2-1
Case 4: Networked Machine Acting as iFOR/LS Server	2-1
Case 5: NCS Application Not Using iFOR/LS	2-1
Case 6: License Provider Wants to Create End-User Licenses from Compound Licenses	2-1
License Server Setup	2-1
Configuring NCS	2-2
Choosing a License Server	2-3
Installation Procedure	2-5
Post-Installation Configuration	2-5
netls_config Shell Script	2-5
When a GLB Daemon Already Exists	2-5
When a GLB Database Does Not Exist	2-6
When No Cells Exist	2-6
Record of Target ID for License Reissue	2-7
iFOR/LS Error Logging	2-7
iFOR/LS Directory Structure	2-7
Verifying Operation of the iFOR/LS License Server	2-7
Starting and Stopping the iFOR/LS Server Properly	2-8
Special Installation Considerations for iFOR/LS	2-9
Compatibility with RLM and NCS Version 1.1	2-9
Upgrading NetLS to iFOR/LS	2-9
iFOR/LS Installation Examples	2-10
Example 1: Scenario for an Existing NCS iFOR/LS Network	2-10
Example 2: Scenario for an Initial Installation	2-10

Chapter 3: Managing iFOR/LS	3-1
Management Tools for iFOR/LS	3-1
Summary of Tasks You Can Perform through SMIT	3-1
Tasks Requiring Commands	3-2
Adding or Deleting Nodelocked Licenses	3-2
Adding or Deleting Concurrent-Use and Use-Once iFOR/LS Licenses	3-6
Managing License Servers	3-11
Getting Status Information with ls_stat	3-12
A Sample ls_stat Session	3-14
License Server Reports	3-17
Choosing Events to Be Logged	3-17
Deleting Stale Log Entries	3-18
Maintaining Daemons (llbd, glbd, and netlsd)	3-18
Creating and Maintaining User Files	3-18
Chapter 4. Troubleshooting iFOR/LS	4-1
Troubleshooting Nodelocked Licenses	4-1
Troubleshooting Concurrent-Use and Use-Once Licenses	4-2
ls_tv Errors	4-4
NCS Problems and Solutions	4-5
Implications of NCS Communications Failures	4-5
General Hints on Handling NCS for iFOR/LS	4-5
Quick and Easy NCS Cell Fixup	4-6
Manual NCS Cell Fixup	4-6
Cleaning up Location Broker Databases	4-7
Common Customer Problems and Solutions	4-8
Problems Relating to Specific Configurations	4-11
Monitord Error Messages and Recovery Information	4-13
iFOR/LS (netlsd) Error Messages and Recovery Information	4-17
Chapter 5. Network Computing System (NCS)	5-1
NCS Components	5-2
Remote Procedure Call (RPC) Run-Time Library	5-2
Location Brokers	5-2
Configuring the glb_site.txt File	5-3
Manually Configuring the Network Computing System	5-3
Installing a Global Location Broker Daemon in the Default Cell	5-4
Establishing a New NCS Cell	5-5
Installing a Global Location Broker Daemon in the Alternate Cell	5-7
Setting up the iFOR/LS License Server	5-8

Appendix A: iFOR/LS Commands	A-1
drm_admin Command	A-2
glbd (NCS) Daemon	A-6
lb_admin Command	A-9
lb_find	A-12
llbd (NCS) Daemon	A-14
ls_admin Command	A-16
ls_dpass Command	A-21
ls_rpt Command	A-29
ls_stat Command	A-31
ls_tv	A-34
monitord Daemon	A-36
netlsd Daemon	A-37
nrglbd Daemon (NCS)	A-39
Appendix B. iFOR/LS Files	B-1
glb_obj.txt	B-1
glb_site.txt	B-2
ARK Log File	B-3
Format of the User File	B-8
Appendix C. RLM Compatibility	C-1
iFOR/LS Glossary	X-1
INDEX	X-4

About This Book

AIX Version 4.1 iFOR/LS System Management Guide guides you through setting up and maintaining the environment needed to support a software product licensed with the Information For Operation Retrieval/License System (iFOR/LS).

Note: The information in this book can also be found in the Hypertext Information Base Library 1.1 for AIX. This online documentation is designed for use with the InfoExplorer hypertext retrieval system.

Who Should Use This Book

The Version 4.1 edition of this book is intended for network system administrators who will be installing iFOR/LS concurrent-use licenses (also called floating licenses). Although readers of this book are expected to know basic operating system commands, references to specific commands are provided. This book should be used in conjunction with *iFOR/LS Tips and Techniques*, SC23-2666.

How to Use This Book

This book is organized to help you quickly find the information you need. You should first read Chapter 1 to get an overview of iFOR/LS, then install the software using the procedures in Chapter 2. You will then need to refer to Chapters 3 and 4 periodically as you maintain and troubleshoot iFOR/LS.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
<i>Italics</i>	Identifies parameters whose actual names or values are to be supplied by the user.
Monospace	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Related Publications

AIX Version 4.1 Installation Guide, SC23-2550, discusses installation of the operating system and other licensed programs as part of the base operating system.

iFOR/LS Tips and Techniques, SC23-2666, provides examples of various iFOR/LS configurations and provides an in-depth description of advanced administration topics.

Ordering Publications

You can order publications from your sales representative or from your point of sale.

If you received a printed copy of *Documentation Overview* with your system, use that book for information on related publications and for instructions on ordering them.

To order additional copies of this book, use order number SC23-2665.

Chapter 1. Introduction to iFOR/LS

This chapter describes the application Information For Operation Retrieval/License System (iFOR/LS), which helps companies control how the software they purchase is used. Through the use of encrypted keys, iFOR/LS can monitor the type and number of licenses used by standalone machine, or by machines within a network. In turn, the company can better ensure its own compliance with the terms of licensing agreements.

iFOR/LS represents an enhanced version of Network License System (NetLS) Version 2.0.1 (GRI 1.1.2b). The enhancements include changes to the run-time code, NCS, and the graphical user interfaces. Both NetLS and iFOR/LS are products of Gradient Technologies, Inc. While the name of product has changed, some of the path names and binary names continue to use “netls.” For example, the **nodelock** file can still be found in **/usr/lib/netls/conf**.

The distributed computing technology of NCS lets iFOR/LS users access software applications from any machine in the iFOR/LS network. See “Distributed Computing with NCS” on page 1-6 for more information on the basic NCS tools used by iFOR/LS.

Application developers use the iFOR/LS Application Developer’s Toolkit (ADK) to enable their applications; system administrators use the iFOR/LS Administrator Runtime Kit (ARK) to install and maintain the license server and the location broker system. In this book, the term “license server” will be used to refer to the ARK. Only the license server portion of iFOR/LS is shipped with the operating system.

iFOR/LS code is embedded in the client application, the remote service (in this case, the iFOR/LS license server), and the location broker system. All three of these are invisible to end users in a properly configured system. The figure “Typical License Server Installation” on page 1-5 shows the components of a typical license server installation and indicates how they communicate with one another.

Once you have set up and configured the environment for the first iFOR/LS-enabled product, the same environment will support other products licensed with iFOR/LS.

Please note that all examples presented here apply specifically to systems running AIX Version 4.1. Another platform may have different starting and file location requirements. Refer to the documentation shipped with that platform for specifics.

iFOR/LS Enablement for the AIX Version 4.1 Base Operating System

There are several ways to access the AIX Version 4.1 BOS system. For the purposes of AIX license management using iFOR/LS, the following are ways the system can be accessed when an AIX Version 4.1 license may be required:

- a. Logins provided by means of a **getty** (from an active, local terminal)
- b. Logins provided using the **rlogin** or **rsh -l** command
- c. Logins provided using the **telnet** or **tn** command
- d. Logins provided through the Common Desktop Environment (**visual login CDE**)

Types of iFOR/LS Licenses

iFOR/LS provides a variety of licensing models for software product providers. These models allow them to bundle compliance mechanisms with their software. By tracking license usage, iFOR/LS allows customers to easily comply with their software license agreements.

No one license type is appropriate for all products or for all customers. For example, nodelocking is often inconvenient for end users: anyone who wants to use a nodelocked product must work at the system to which it is locked. Furthermore, if the node is down or cannot be accessed, no one can use the product.

Three license types, nodelocked, concurrent-use, and use-once, are the most commonly used. The table below summarizes the attributes of each license type:

License Type	Uses iFOR/LS Server?	Application Access	Usage	Can Be Reused
Nodelocked	No	Limited to machine	1 license=unlimited usage on the installed node	Yes
Concurrent use	Yes	Whole network	1 license=1 usage	Yes
Use once	Yes	Whole network	1 license=1 usage	No

Each of the three license types share common traits:

- They are all used to satisfy license requests made by an iFOR/LS client.
- Each license is supplied as an encrypted set of characters.
- A license type can only be used if the product developer has enabled the product to use that specific license type.
- Each license type has an expiration date, such as March 10, 1995.
- Each license must be installed on a specific machine.

Nodelocked License

Nodelocking (also known as CPU locking) is a licensing mechanism requiring each node (workstation) on which the licensed software product operates to obtain an authorized license for its unique target_ID. The software product assumes the role of a client and, when invoked, requests a license from the **nodelock** license file (**/usr/lib/netls/conf/nodelock**). The file contains a list of valid licenses. A user can invoke the application as many times as wanted on the licensed node.

Nodelocked licenses do not require use of the network, NCS, or the license server.

To obtain a nodelocked license, you simply provide the target_ID of your computer. The required information is printed when you run the **ls_targetid** tool (in **/usr/lib/netls/bin/ls_targetid**).

Nodelocked licenses are often used when:

- Your site has only a small number of nodes. For such sites, nodelocked licenses may be more convenient to administer than a server.
- An end user of a software product does not want to compete for a concurrent-use license with other users of the product.
- One or more server nodes at a site have become unavailable (perhaps because they are being serviced). In this case, end users do not have access to software products, even though the licenses have not expired. Since you may not be able to move the concurrent-use licenses to a server node that is available, you may receive temporary nodelocked licenses from the vendors to use until your server nodes are working.
- The product is licensed only to a designated machine.

Concurrent-Use Licenses

Concurrent-use licenses (also known as “floating licenses”) allow a software product to be accessed from any machine in the network. The number of users who may use the product simultaneously is limited only by the number of nodes at the site and the number of licenses purchased. In addition, multiple-use rules allow a user to run multiple instances of an application simultaneously, without using multiple concurrent-use licenses. This method uses a typical client/server architecture in which an iFOR/LS server must be active in the network. The enabled application acts as a client, sending a request for a license to the iFOR/LS server. The machine on which the application is running can serve as both a client and a server.

License servers maintain information about the vendor names, software product IDs, and the number and expiration dates of licenses purchased for each product. The server uses this data to satisfy license requests from clients.

Communication is established by NCS protocols when the licensed product requests a license. If that license server has concurrent-use licenses available for that product, the license is granted and the product executes normally. If no concurrent-use licenses are available on the first server located, other servers automatically try to satisfy the request from their installed licenses. As more users run the application and request licenses, this procedure is repeated until all available concurrent-use licenses are in use.

Use-Once Licenses

A use-once license is identical to a concurrent-use license, except that it cannot be reused. This consumable licensing method enables use of a product only once within the period for which the license is valid. Using NCS protocols, each product initialization causes the license server to decrement the number of usable use-once licenses by one. When this value equals zero, the license cannot be used for any additional requests. A use begins when the software product requests and receives a license. Typically, a use is completed when the product session ends, although the application developer can decide to use another criterion.

Software vendors can provide use-once licenses as the primary license type for a product, as free-sample or demonstration licenses, or as backup licenses for products that are primarily licensed on a concurrent-use basis.

Vendor-Specific Administrative Considerations

A software vendor may offer different kinds of licenses for the same product. Sometimes a product that is usually licensed for concurrent use might also be offered on a nodelocked basis. Demonstration licenses for a new version of a product might be use-once, whereas established versions of the product might be licensed as concurrent-use. Vendors can also produce compound licenses, which allow system administrators to generate his or her own license passwords for a given number and type of licenses. Usually when a vendor offers different kinds of licenses for the same product, one of the license types is the primary licensing mechanism, and other types are offered to suit special circumstances.

A vendor might also provide hybrid licenses for a product. For example, if all concurrent-use licenses for a product are in use during a peak time, the product might be designed to look for use-once licenses in order to give access to more users. This type of product would require the system administrator to install and manage both types of licenses.

Under iFOR/LS, software vendors are not constrained to tie iFOR/LS licenses with product usage. For example, the total number of users allowed for a product may be greater than the total number of valid concurrent-use licenses. Or you may be able to use a product even after all of its licenses have expired. In these cases, the number and term of the licenses are not reliable indicators of how many users can use the product, nor for how long. In the

event of unavailable or expired licenses, the behavior of any product is solely the decision of the application vendor, and not the iFOR/LS server.

With the flexibility of iFOR/LS, software vendors can implement many different licensing and license-enforcement mechanisms. See the vendor's documentation for the manner in which iFOR/LS was implemented for that product. Also look for information about how the vendor uses license annotation and any other information about how the terms of the licenses are enforced. Information of this type makes it easier to understand and administer the licenses.

License Annotation

A *license annotation* is special data defined by the software vendor and optionally included as part of the license information when the license is created. The annotation can be used to specify additional conditions in the key. For example, an annotation could be included in a nodelocked license to extend its capabilities. A typical nodelocked license allows unlimited usage on a single machine. The product vendor may want to count each instance of the application. In this case, the annotation would include an integer representing the number of instances allowed by that nodelocked license. Note that this also requires the product developer to dynamically count the number of executing instances because the server is not used, and no counting occurs automatically.

As another example, suppose an interactive graphics application includes optional features A, B, and C, all of which are bundled with the product, but sold separately. The vendor can use license annotations to specify which options are available to a given customer, since different customers buy different options. Therefore, if a customer buys options A and C, that customer would receive licenses with an annotation specifying that options A and C, but not B, are accessible. If no options are purchased, no annotation is required.

License Passwords

A password is a string encoded with information about a software vendor (**vendor password**) or about a software product (**product password**).

The vendor password, together with a vendor ID, establishes the vendor of a license product in a license database.

A product password can be of two types:

license password

A string that specifies licenses for a product and contains other data that ensure the security of the licenses.

compound password

Specifies licenses indirectly by allowing the creation of license passwords. Compound passwords are indispensable if products are distributed by other software suppliers. For example a compound password can specify that one or multiple concurrent-use licenses can be generated. If the compound password is installed at the license server that it has been generated for, the license system administrator can use administrative tools (**ls_dpss**) to generate end user concurrent-use licenses from this compound password.

The Vendor Key

The license server encodes data for a product using a vendor-defined value called the vendor key. The network license system uses the key to both encode and decode license information in a password, and to encode and decode communications between license servers and the products they regulate.

License Databases

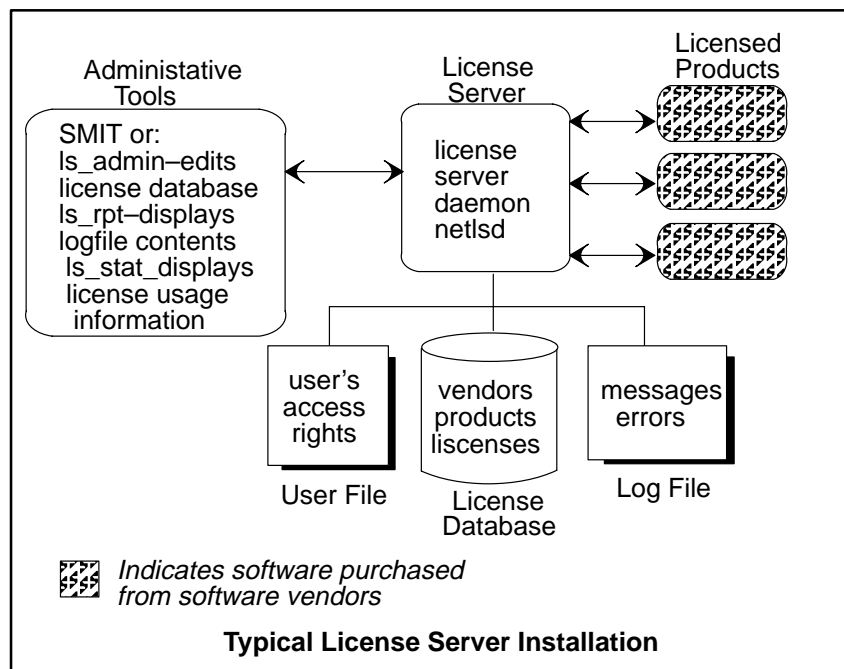
License databases contain information about licenses. This information consists of the product name, the product version, the number of licenses, the license type, the start and end dates of the licenses, the target type, the target ID, and a time stamp. License databases are maintained by a license server. The license databases maintained by iFOR/LS are **lic_db** and **cur_db**. **lic_db** stores information about all licenses maintained by iFOR/LS, while **cur_db** stores information about the current status of concurrent use licenses held by users. Both files are located in the directory **/usr/lib/netls/conf**.

Components of iFOR/LS

The following are parts of the licensed product and the iFOR/LS license server:

- The license server daemon (**netlsd**), which processes requests from the product by referring to information about licenses and users.
- The license database, which contains such data as vendor and product identifiers, the number and type of licenses for each product, and their expiration dates.
- The user file, which contains the names of users who are authorized, or not authorized, to use the licensed product.
- Administrative tools, which system administrators use to manage iFOR/LS.
- The log file, where the license server daemon keeps a history of license server events.

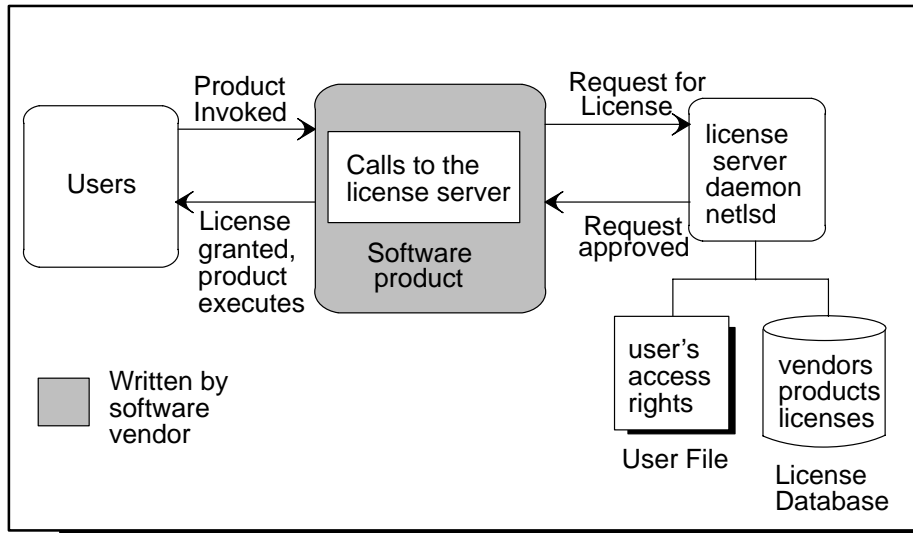
The figure “Typical License Server Installation” shows the components of a typical installation and indicates how they communicate with one another. These components will be discussed in detail in later chapters.



A Walkthrough of iFOR/LS

The figure “Invoking a Non-Nodelocked Licensed Product” shows the events that take place when a user invokes a software product that is administered by the license server daemon. In most cases, a user does not even know that a license request is being made. Note that both concurrent-use and use-once licenses have license requests satisfied by the iFOR/LS

server (**netlsd** daemon). Nodelocked license requests do not go through the server. Instead these requests are satisfied by accessing the **nodelock** file directly.



Invoking a Non-Nodelocked Licensed Product

The following is a simplified summary of the sequence of events associated with using a non-nodelocked licensed product:

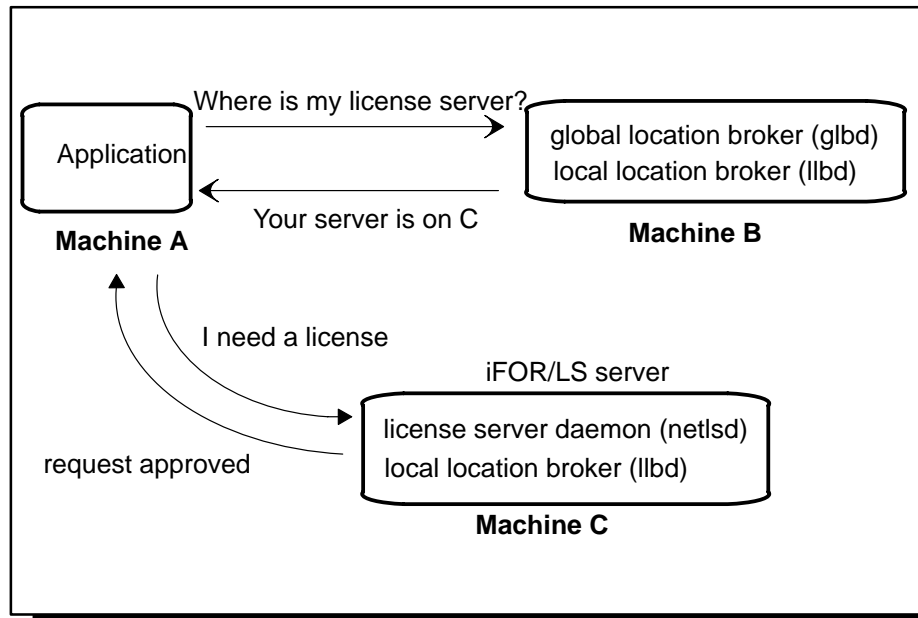
1. A user invokes the product.
2. The product requests a license from a license server daemon.
3. The license server checks the license database to see if a valid license is available, and the optional user file describing user access rights, to find out whether the user is allowed to use the product.
4. If a license is available and the user is allowed to use the product, the server approves the request.
5. When the user receives a license from the license server daemon, the product begins executing.
6. When the user exits the product, the product releases the license back to the license server.

Distributed Computing with NCS

Distributed computing is a way of spreading network resources around on a network. A major goal of distributed computing is to provide independence of services to avoid bottlenecks and provide higher availability. Another goal of distributed computing is to hide the details of where services are provided on the network. The distributed solution currently supplied with iFOR/LS is the Network Computing System (NCS) Version 1.5.1.

Two major components of NCS are the *global location broker* and the *local location broker*. The **glbd** is the global location broker daemon. The function of the **glbd** is to provide information that will allow an NCS client application to find and use the services of an NCS server application. The **glbd** is itself an NCS server application. The local location broker daemon (**llbd**) must run on every machine that has an NCS server running. The function of the **llbd** is to manage information about the NCS servers running on that machine. Using the **llbd** requires no special configuration at this level of operation.

iFOR/LS and NCS are used together in the following way. The user applications that make license requests are NCS clients. The license server (**netlsd**) is an NCS server. Therefore, before an application can get a license from an iFOR/LS server, it must first communicate with a global location broker to find out where the license servers are running. The global location broker receives the client request and replies with the information necessary for the client to establish communications with the server. The local location broker provides some data and communications management during the transaction. If the network is configured properly, all this is transparent to the user. The figure “Interaction of iFOR/LS and NCS” shows how **netlsd**, **llbd**, and **glbd** interact.



Interaction of iFOR/LS and NCS

Sometimes, it is desirable to logically partition a network so that some of the servers only serve some of the clients. This grouping of clients and servers is known as a *cell*. A user application that requires a license will only be able to have its license request satisfied by a license server in the same cell. For information on creating a new cell, see “Establishing a New NCS Cell” on page 5-5. An iFOR/LS server can be easily configured and reconfigured to change its cell membership.

In a large network with more than one subnet, the broadcast from the client application may not reach the host where the global location broker is running. If this happens, the client machine can be configured to specifically direct the NCS request to the global location broker host machine.

The following figure gives a brief description of basic NCS terms.

Basic NCS Terms

global location broker (GLB)	A service that manages the location of all NCS services in the NCS network. The GLB uses one or more daemons, glbds , to maintain this service.
local location broker (LLB)	A service that provides an interface to the GLB from the iFOR/LS server. The LLB has no information about network-wide services. The LLB daemon (llbd) runs continuously in the background to intercept and forward information to the glbd .
NCS cell	A logical concept of grouping together one or more machines in an NCS network. Any node belonging to an alternate cell can only have its license requests satisfied by iFOR/LS servers in that cell.
default cell (global access)	An NCS cell that allows access from any node in the network. Since it allows all other servers in the default cell to communicate freely, this is the most common cell used in iFOR/LS configurations.
alternate cell (isolated)	An NCS cell that restricts iFOR/LS license transactions to a partial group of network nodes.

iFOR/LS Platforms

iFOR/LS runs on a number of platforms. To determine interoperability with another platform, check the vendor documentation for any limitations of running with either Network License System Version 2.0.1 (GRI 1.1.2b or greater) or NCS Version 1.5.1. The version number for iFOR/LS can be found in **/usr/lib/netls/ark/VERSION.ARK**.

Chapter 2: Installing and Configuring iFOR/LS

The following sample scenarios should help administrators to clarify which iFOR/LS options they need for their environments. For more detailed information about iFOR/LS packaging, see “iFOR/LS Options: Packaging and Requirements” in *AIX Version 4.1 iFOR/LS Tips and Techniques*. Following the sample scenarios and requirements is a detailed explanation of the types of decisions that can be made and their implications.

The following sample scenarios should help administrators to clarify which iFOR/LS options they need for their environments. For more detailed information about iFOR/LS packaging, see “iFOR/LS Options: Packaging and Requirements” in *AIX Version 4.1 iFOR/LS Tips and Techniques*.

Requirements and Sample Scenarios

Case 1: Standalone Machine Running AIX

In this case, no options are required.

Case 2: Standalone Machine Running AIX and CSet++

Since CSet++ utilizes concurrent-use licenses an iFOR/LS server is required. This standalone machine must have *all four* options installed and configured.

Case 3: Networked Client Running AIX and CSet++ (iFOR/LS Server Is on Another Machine)

The `bos.ifor_ls.client` option can be installed to enable this client for license administration and license-usage reporting.

Case 4: Networked Machine Acting as iFOR/LS Server

An iFOR/LS server needs *all four* options.

Case 5: NCS Application Not Using iFOR/LS

The `bos.net.ncs` is required.

Case 6: License Provider Wants to Create End-User Licenses from Compound Licenses

In this case *all four* options are needed because the `ls_dpasm` utility within the `server` option is needed.

Before installing the licensed product, you must determine which types of license are used with the product (nodelocked and/or concurrent-use and use-once licenses). If you are installing iFOR/LS only for use with nodelocked licenses, you simply install the licensed product and create a **nodelock** file. You do not need to install the iFOR/LS license server. The nodelocked file is discussed in more detail in “Adding or Deleting Nodelocked Licenses” on page 3-2.

If you are installing iFOR/LS for use with concurrent-use and use-once licenses, read “License Server Setup” and then “Installation Procedure” on page 2-5.

License Server Setup

Follow these five steps:

1. Plan your setup.

- a. How many servers should be set up (default = 1)?
 - b. What machines will run the servers?
 - c. Which NCS cells should be used (default = "default cell")?
2. Check prerequisites (e.g., TCP/IP is required).
 3. Install the iFOR/LS server from media (use SMIT).
 4. Do the post-installation configuration.
 - a. Stop any existing related daemons (**netlsd**, **glbd**, **llbd**).
 - b. Run **netls_config**, then **netls_first_time**.
 5. Verify your work using the **ls_tv** tool.

For concurrent-use and use-once licenses, you must install the iFOR/LS license server module. The module (or iFOR/LS Administrator Runtime Kit package) for AIX consists of one installp image, **bos.ifor_ls**.

This section describes the procedure for installing the iFOR/LS license server on AIX systems.

Configuring NCS

The global location broker maintains a database of where all services reside on the network. A local location broker must run on each node that provides a service. It handles communication between the global location broker and the actual servers.

The local location broker daemon is **llbd**. The program is called a daemon because it runs continuously in the background, waiting for requests for the function it provides. As with most daemons, the function it provides is called infrequently. It is usually just waiting for requests and has little effect on system performance. Every node that provides services to NCS must run **llbd**, including the systems that run the global location broker service and the iFOR/LS license server.

At least one node on the network must run the global location broker daemon, **glbd**. Most configured networks use at least one machine with three daemons (**llbd**, **glbd**, and **netlsd**) running. If the network already uses NCS, one or more global location brokers will already be in place, and you may not need to worry about this issue. Otherwise, plan the global location brokers with these points in mind:

- 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 iFOR/LS license servers to run the global location broker.
- With a larger network, it may be best to set up at least one global location broker server and one iFOR/LS license server on each subnet, or LAN. The LANs may be connected by routers that do not support NCS broadcasts to license servers. The other alternative is to copy a **glb_site.txt** file onto each machine that requires access to a license server across the router. This file contains the hostname or address of the global location broker host. For information on this file, see "Configuring the glb_site.txt File" on page 5-3.
- Sometimes it is best to isolate the location broker servers from network-wide location broker servers. For example, if iFOR/LS services are set up for a department, and the department is networked throughout (or beyond) your company, you may want to make sure that the licenses are limited just to that department. This is done by setting up an NCS *cell* through creation of a **glb_obj.txt** file. A cell is a subset of a network or internet in which all hosts use the same GLB universal unique identifier (UUID). Location broker requests and replies are then limited to those between nodes in the same cell, limiting the

scope of the licensing. For information on creating a new cell, see “Establishing a New NCS Cell” on page 5-5.

- Some systems do not support replicatable location brokers. A location broker running on such a system is referred to as *non-replicatable* or **nrglbd**. Due to the implications below, using **nrglbd** is not recommended:
 - There can be only one **nrglbd** running per cell. This can have an impact on the availability of the license servers. If the system running the **nrglbd** goes down, applications will not be able to see the license servers.
 - If the cell is made up of mixed systems, some of which support replicatable location brokers and some of which do not, it is a good idea to stop the **nrglbd**, and move the location broker services to one or more of the systems that support the replicatable location brokers. You cannot run both a **glbd** and an **nrglbd** simultaneously within the same cell. If you decide to convert from non-replicatable to replicatable location brokers after iFOR/LS has been installed, you must stop all license servers first and then restart them after you have finished the conversion.

To summarize the NCS requirements for concurrent-use licensing with an iFOR/LS license server:

- There must be one or more nodes running the **glbd**, on the network.
- There must be one or more network license server daemon, **netlsd**, on the network.
- Any node that runs **glbd**, **nrglbd** or **netlsd** must also run the **llbd**.
- **llbd**, **glbd**, and **netlsd** should be running for each license server.

Choosing a License Server

When you use concurrent-use and use-once licenses, you must determine which computer or computers are to be used as iFOR/LS license servers. The following are a few suggestions for selecting license servers:

1. A license server should be a computer that stays on at all times, or at least at all times that someone might use the licensed product. Computers that are frequently unavailable-unreliable systems, or ones that are taken down often for testing or maintenance-are not good candidates.
2. Diskless nodes do not make good license servers.
3. Ideally, license servers should be on the same LAN as the majority of computers that will run the licensed product. Accessing license servers in another LAN, across a bridge or router, may not be quite as fast. Also, some routers can make setting up license servers less convenient.
4. Licenses can also be distributed (partitioned) between two or more iFOR/LS license servers. When a computer is down, the licenses assigned to the license server on that system are unavailable, but licenses assigned to other license servers remain available. If you have several licenses for one product, consider setting up more than one license server and distributing the licenses.
5. Computers that function as license server nodes can run the licensed product. The license server software does not have a noticeable effect on the licensed product's performance.
6. It is not necessary to install the licensed product software on license server nodes if you are not going to run the product on those nodes.
7. If you have a computer that is not in a network but you have purchased a concurrent-use or use-once license, you can install the iFOR/LS license server software on that computer. It then acts as both the license server and the client. This is a common

configuration; however, the licenses installed in that server will not be accessible by other workstations until it is connected to the network.

If a computer has already been designated as a server (for example, a file server), it becomes a good choice because it satisfies many of the previously mentioned criteria. In most cases, it will also function well as a license server.

A typical installation program lets you install the licensed product, the iFOR/LS licensing system, or both. Install the iFOR/LS licensing system only on the nodes that will function as license servers. Remember that you must log in as root (or use the **su** command) to install the iFOR/LS license server software. Follow the instructions supplied in the release notes for your product.

Now that you understand basic NCS requirements and have chosen a server, you are ready to install iFOR/LS. The following sections describe the system requirements for installation and procedures for installing the licensed program.

Prerequisites

- You must log in with root authority.
- The AIX version of the iFOR/LS license server is supported on IBM RISC System/6000 processors running AIX Version 4.1. You must have installed the base operating system run time, **bos.rte**, which is part of the AIX licensed program. Refer to the *AIX Version 4.1 Installation Guide* for instructions on installing AIX.
- You must have installed the Network Computing System 1.5.1 run time (**bos.net.ncs**), which is included with AIX Version 4.1.
- The clock on the machine must be set to the correct time; otherwise, the license may not be recognized as active.

Note: If NCS Version 1.5.1 is installed, the output of the **lslpp** command will display version and release levels of 03.02 similar to the following:

```
lslpp -h bos.net.ncs
Name
-----
Fix Id  Release          Status      Action      Date
-----
Path:  /usr/lib/objrepos
                                     COMPLETE   COMMIT      01/17/92

Path:  /etc/objrepos
                                     COMPLETE   COMMIT      01/17/92
```

- If you want to use the System Management Interface Tool (SMIT) to administer iFOR/LS, you must have **bos.sysmgt** installed.
- Administration commands for iFOR/LS support both a command line interface and a windows interface. If you want to use the iFOR/LS graphical user interface administration tools, you must have installed AIXwindows X11R5 (**X11.base.rte** in **AIX Version 4.1**). Refer to the *AIX Version 4.1 Installation Guide* for instructions on installing AIXwindows.

Warning: If you already have a version of iFOR/LS, NCS, or Resource License Manager (RLM) installed, please refer to "Installation Considerations with NCS Version 1.1 and RLM" on page 2-9 for instructions on how to stop the daemons or applications that are running before installing the product.

- You must have installed TCP/IP (**bos.net.tcp**), included in the operating system.
- You must have at least 5MB of disk space available in **/usr/lib**.

Warning: A backup should be performed before any installation.

Installation Procedure

1. Insert the tape or CD-ROM containing the iFOR/LS installp image (**bos.ifor_ls**) into the drive.
2. Use the **smit** command to install the code.

For more information about installation in general, refer to *AIX Version 4.1 Installation Guide*.

Post-Installation Configuration

You must perform all post-installation configuration steps. These steps require root authority. While they can be performed manually, we recommend using the provided scripts. If you want to configure iFOR/LS manually, see “Manually Configuring the Network Computing System” on page 5-3. Before performing post-installation tasks, stop the **llbd**, **glbd** and **netlsd** daemons, if they are running, either by running the **srv_stop** command or by running the following commands in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

netls_config Shell Script

The installation process installs a shell script called **netls_config** in the directory **/usr/lib/netls/conf**. When the install procedure is completed, run the **netls_config** shell script. This shell script gathers configuration information and creates another shell script called **netls_first_time**, also in the directory **/usr/lib/netls/conf**.

You can run **netls_config** and **netls_first_time** as many times as you need to set up the desired configuration. For example, to set up two alternate cells, you will simply run both scripts on each machine that will be a server on each cell.

Warning: You should not run one of these scripts without running the other, and you should always run **netls_config** first.

When you run the **netls_config** shell script, you will be asked if you wish to start the **llbd** daemon automatically each time the system reboots. If your choice is **y**, an entry is added to the **/etc/inittab** file to run **/etc/rc.ncs**. You will also be asked if you wish to start the license server daemon, **netlsd**, each time the system reboots. If your choice is **y**, another entry is added to **/etc/inittab** file to run **/etc/rc.netls**.

When a GLB Daemon Already Exists

If you have installed iFOR/LS on a system that already had a running NCS global location broker (GLB) daemon, you are prompted as to whether you wish to use the existing GLB database. The default answer is yes. If your choice is **y**, the configuration script completes without further intervention. If your choice is **n**, the existing GLB database is removed. It is recommended that you choose **n**.

See “NCS Problems and Solutions” on page 4-5 for the implications of using the existing GLB database as opposed to creating a new database.

When a GLB Database Does Not Exist

If a GLB database does not already exist, the configuration procedure first checks for global location broker server daemons running elsewhere on the network. The **netls_config** shell script sends out inquiries to the NCS global location broker server daemons and displays the results; these results show the following four items:

1. The GLB broker type (usually replicatable)
2. The server host's network address
3. A cell name (either default or alternate_N, where N is an integer)
4. The cell's universal unique identifier (UUID).

The following example output indicates one entry in the default cell and one entry in the alternate cell:

```
replicatable    ip:iliad.austin.ibm.com    default
                333b91c50000.0d.00.00.87.84.00.00.00
replicatable    ip:odyssey.austin.ibm.com  alternate_1
                65c2435e4cd1.02.81.23.92.91.00.00.00
```

When No Cells Exist

If no existing cells are found on your network, you are asked to choose one of the following options:

- Continue installation without choosing a cell name.
Since a cell name is not chosen, this means you will have to run **netls_config** again. Use this option only if you are unsure or out of time.
- Use the default for the system cell name.
This is the recommended option. It will allow all servers to communicate freely.
- Create a new alternate cell for the system cell name.
Use this option if you need to create an isolated cell. Such a cell may be needed for test purposes or if you do not want other machines to share your licenses. A **glb_obj.txt** file will be created in the **/etc/ncs** directory. This file contains the UUID of the global location broker. This allows the system to override the default GLB.

Note: When you choose a cell name, three files are created in **/etc/ncs**: **glb_log** (log file), **glb.e** (containing entries in the GLB), and **glb.p** (containing a propagation queue).

If one or more existing cells are found on your network, you are asked to choose one of the following options:

- Continue installation without choosing a cell name.
- Use the default for the system cell name. This is the recommended choice.
- Create a new alternate cell for the system cell name.
- Choose an existing alternate cell for the system cell name.

Choosing this option will replicate the **glb_obj.txt** file from the chosen host.

Again, in most cases you will want to use the default system cell name. The information gathered by the configuration procedure is placed in a shell script called **/usr/lib/netls/conf/netls_first_time**. You must run this shell script after the **netls_config** shell script completes.

Notes:

1. If you continue with installation without choosing a cell name, then iFOR/LS will not function until you either manually configure a cell or rerun the **/usr/lib/netls/conf/netls_config** shell script.
2. **netls_first_time** should only be run once, unless you run **netls_config** a second time. After the GLB database has been created, **glbd** must be started with *no* options on that machine. You cannot simply take down the brokers and rerun **netls_first_time**.

See “Starting and Stopping the iFOR/LS Server Properly” on page 2-8 for the procedure to start and stop the NCS daemons manually. You must start and stop daemons in an exact order.

Record of Target ID for License Reissue

In the event that your target ID must be replaced (for example, when the hardware planar has to be replaced), you must get new licenses based on your new target ID. The easiest way for your licenses to be verified is to provide your old target ID. To facilitate this, a file is kept that records the target ID each time **netls_config** is run.

The file is in **/usr/lib/netls/conf/list_targetids**. Below is an example of the file:

```
Wed Mar 16 12:08:56 CST 1994
20531
Tue Apr 12 15:32:25 CST 1994
47432
```

The most recent target ID, 47432, is listed last. The other ID, 20531, represents a target ID that was replaced with 47432.

iFOR/LS Error Logging

During the startup of the daemons, **glbd** and **netlsd**, standard error and standard output go to **/dev/console**. See “Troubleshooting iFOR/LS” for more information on iFOR/LS error messages.

iFOR/LS Directory Structure

The directory structure for iFOR/LS after installation is as follows:

/usr/lib/netls/conf	Contains license files, user files, and configuration scripts.
/usr/lib/netls/bin	Contains executable commands for managing the license server.

These path names are used throughout the book.

Permissions are established so that the administration tools (except **ls_admin**) can be used by anyone. The permissions of other files in the **/usr/lib/netls/bin** directory are such that only **root** or **bin** can start **netlsd**. Do not change these permissions.

Verifying Operation of the iFOR/LS License Server

After having set up one or more iFOR/LS license servers, verify that they are active on the network using the **ls_tv** (license server test and verification) program. This test can be performed from any node on which the licensed product or the iFOR/LS server module is installed. In fact, the test should be performed on every node on which the licensed product is installed. You need not be logged in as root for this test. Use this syntax:

```
cd /usr/lib/netls/bin
ls_tv
```

You should run this test periodically to ensure that all servers on your network actually respond. The report printed out should be somewhat like this:

```
LS_TV Version 2.0.1(GR1.1.2a) AIX - iFOR/LS Test and Verification
Tool
(c) Copyright 1991,1992,1993 Hewlett-Packard Company, All Rights
Reserved
(c) Copyright 1991,1992,1993 Gradient Technologies Inc., All
Rights Reserved
Completed license transaction on node 4991c running iFOR/LS
Active iFOR/LS Servers:
    illiad.austin.ibm.com (AIX) running iFOR/LS
    odyssey.austin.ibm.com (AIX) running iFOR/LS
```

All the iFOR/LS license servers in the network should be listed.

Make these checks if you have trouble getting a response from the iFOR/LS license servers on the network:

- Make sure the **glbd** is running somewhere on the network and that **llbd** is running on each of the global location broker nodes.
- Make sure that both **llbd** and **netlsd** are running on the iFOR/LS license server nodes.
- If an NCS cell is set up, make sure the **/etc/ncs/glb_obj.txt** files are identical for all the nodes in the cell.

If you continue to have difficulty, refer to “Troubleshooting iFOR/LS”.

Starting and Stopping the iFOR/LS Server Properly

Because iFOR/LS and NCS are components of a distributed network system, it is important that the servers be managed responsibly. Otherwise, location broker databases may become corrupted, and users will not be able to obtain the network services they expect.

By following a few simple rules, you can keep your network running smoothly:

1. Always use the license server configuration script when configuring your server for the first time, or when reconfiguring your server for a different cell.
2. If you have to shut down the license server for some reason, do so by running the **srv_stop** command located in the **/usr/lib/netls/bin** directory. This will ensure that the running daemons are stopped cleanly.

You can also stop the daemons by using the following commands in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

3. If you have to start up the license server manually, do so by running the **srv_start** command located in the **/usr/lib/netls/bin** directory. This will start your daemons in the proper order so that they can register themselves on the network.

You can also start the daemons by using the following commands in the order shown:

```
startsrc -s llbd
startsrc -s glbd
startsrc -s netlsd
```

4. On a large network with more than one license server running, periodically clean up the location broker databases to remove invalid entries and improve network performance.

See “Choosing Events to Be Logged” on page 3-17 and “Deleting Stale Log Entries” on page 3-18 for more information.

Special Installation Considerations for iFOR/LS

You need to consider the version of NCS that you are using, and you need to consider certain compatibility requirements if you are operating RLM along with iFOR/LS. You can upgrade from NetLS to iFOR/LS easily.

Compatibility with RLM and NCS Version 1.1

iFOR/LS can coexist in the same network partition as RLM. RLM-licensed client applications can also have their license requests satisfied by the iFOR/LS server, provided the iFOR/LS server is in the default cell.

Notes:

1. If you want to run RLM on the network that is running iFOR/LS, you must have one **nrglbd** running on the network in the default cell.
2. You cannot have **rlmd** and **netlsd** running at the same time on the same server.

If NCS Version 1.1 or RLM is installed on your system, you must follow the additional instructions provided below. You can verify the presence of NCS Version 1.1 with the following command:

```
lslpp -h bosnet.ncs.obj
```

To verify the presence of the RLM, refer to the documentation that came with the RLM product.

The NCS 1.1 shared library necessary for binary compatibility is shipped with AIX Version 4.1. However, NCS Version 1.1 is not equipped to handle replicatable global location brokers. To properly support replicatable global location brokers, you must have NCS Version 1.5.1. This version is shipped with the iFOR/LS license server.

You must take the following steps to ensure that iFOR/LS will install properly:

1. Log in as root.
2. If necessary, stop the **llbd** and **nrglbd** daemons using the **stopsrc** command as follows:

```
stopsrc -s nrglbd
stopsrc -s llbd
```

If RLM is also running on your machine, you must take the following steps to ensure that iFOR/LS will install properly:

3. Stop the **rlmd** daemon by issuing the following command:

```
stopsrc -s rlmd
```

4. If the file **/usr/lpp/rlm/db/nodelock** exists, the information it contains must be placed in the **/usr/lib/netls/conf/nodelock** file once iFOR/LS is completely and properly installed.

Upgrading NetLS to iFOR/LS

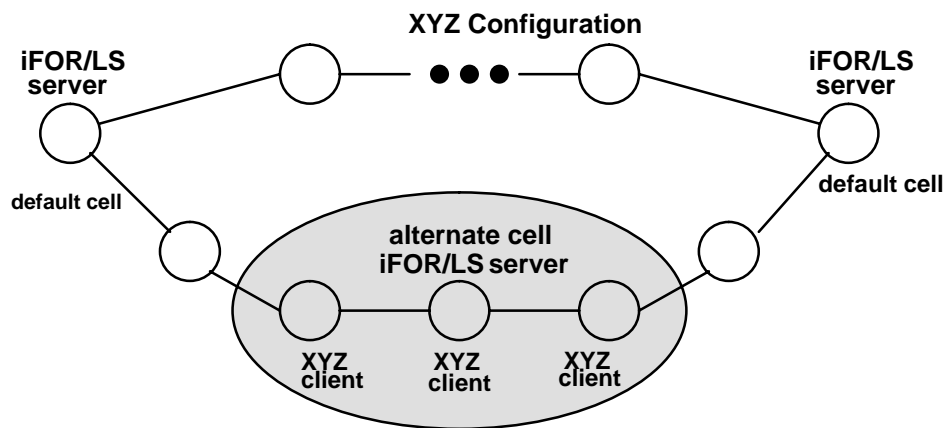
Any license databases you created using an AIX 3.2 version of NetLS can be used with iFOR/LS. If the Application Developers Kit (**NetLS.ADK**) is installed on your Version 3.2 system, it must be removed before you install **bos.ifor_ls**.

iFOR/LS Installation Examples

The following two examples illustrate common installation scenarios. Please refer to Chapter 5 for further information on NCS and NCS cells.

Example 1: Scenario for an Existing NCS iFOR/LS Network

Situation: The site has three people who use the XYZ design product. Since these three people need exclusive access to the product, they do not want other users to have access to the software. Additionally, this group of three users uses other experimental software that sometimes causes network problems. The site already has two iFOR/LS servers set up for other products (similar to the example above). The best solution is to set up a new alternate cell for these three users as shown in the figure “XYZ Configuration.”



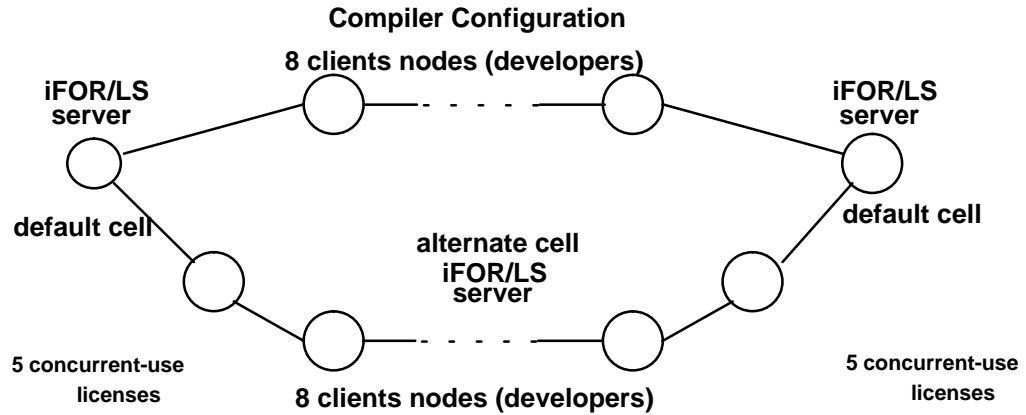
Note: This solution meets the requirements of the situation since it isolates the default cell iFOR/LS servers from the new alternate cell. This solution will require administration of a new iFOR/LS server; and if new users (for example,, a fourth user) are added to the alternate cell their machines will need administration also.

Actions required:

1. Purchase three copies of the XYZ product.
2. Set up the new server as a new alternate cell. Since the usage of this alternate cell is restricted to three users, use one of their machines for the server.
3. Copy the `/etc/ncs/glb_obj.txt` file to the other two machines that will join this alternate cell. At this point each of the three machines will have this file. For information on creating a new cell, see “Establishing a New NCS Cell” on page 5-5.
4. Transmit the target ID of the new server to the issuer of license passwords.
5. Install the iFOR/LS licenses on the new server (see “Managing iFOR/LS” on page 3-1).

Example 2: Scenario for an Initial Installation

Situation: The system administrator just purchased 10 copies of a FORTRAN compiler product that uses iFOR/LS licenses. There are 16 developers who will use the compiler. This is the first case of iFOR/LS on the network. The 10 compilers will require 10 concurrent-use licenses. The administrator decides to ask for two sets of 5 licenses and will set up two servers in the default cell. The configuration could be similar to that shown in the figure “Compiler Configuration.”



Note: This setup will satisfy many situations. The licenses are split into two groups. If the network fails and makes one of the servers unavailable, at least one half (5) of the licenses are still available. The situation assumes that only ten of the developers will require a license at any one time. If the peak use exceeds this level, then more copies of the compilers can always be purchased and their respective licenses installed.

Actions required:

1. Purchase ten copies of the FORTRAN product.
2. Set up the two servers in the default cell.
3. Transmit the target IDs of the two servers to the key generation center. Use the **ls_targetid** command or SMIT to obtain the target IDs.
4. Install the iFOR/LS licenses on the two servers (see "Managing iFOR/LS" on page 3-1).

Chapter 3: Managing iFOR/LS

iFOR/LS provides a comprehensive set of status and reporting tools. These tools are useful not only in the administration of the licensing system, but also in aiding the business relationship of the application vendor and the user.

Managing iFOR/LS

After installing a vendor's software products, the administrator manages iFOR/LS licenses by performing the following tasks:

Note: If you administer a small network where only nodelocked licenses are used, you do not need to run the license server daemon, or perform most of the tasks below; you simply need to perform the tasks described in "Adding or Deleting Nodelocked Licenses" on page 3-2.

Warning: Make sure the clock on your machine is set to the correct time; otherwise, the license may not be recognized as active.

Management Tools for iFOR/LS

System administration for iFOR/LS can be accomplished by means of the SMIT interface, iFOR/LS tools, or commands. The tasks that can be performed through SMIT and those requiring commands are described in the following sections.

Summary of Tasks You Can Perform through SMIT

The following tasks can be performed through SMIT, or through the use of iFOR/LS tools. To perform any of these tasks through SMIT, simply start SMIT with the **smit licenses** fast path command. This will take you directly to the Software License Management menu. Fast paths for each particular task are in parentheses beside each task. Simply preface each fast path with "smit." The helps will guide you through which SMIT fields to complete.

To perform the following tasks using iFOR/LS tools, refer to the procedures in this chapter.

- **Manage License Servers** – Add, delete, and change licenses, vendors, and products on iFOR/LS license server databases. (manage_servers)
 - **Manage Product Licenses** – Add, show, and delete product licenses. As a convenience, it is also possible to add vendor licenses from this menu. (manage_prod_licenses)
 - **Manage Vendors** – Add, show, and delete vendors. As a convenience, it is also possible to add product licenses from this menu. (manage_vendors)
 - **Show Server Characteristics** – Display information about the license servers on your network. Information includes address family, address, port number, target type, and target ID. (show_server_characteristics)
- **Show License Server System Status** – Display information about various license servers and their vendors, products, and licenses. (show_server_status)
 - **Show Total License Usage** – Display licenses in use and available licenses on the servers that you specify. (show_total_license_usage)

- **Show Installed Licenses** – Display the number, types, start date, and expiration date of installed licenses. (show_installed_licenses)
- **Show Current License Usage** – Display licenses currently being used. (show_current_license_usage)
- **Show Licenses held by a specific user** – Display licenses checked out by a specific user. (show_user_license_held)
- **Verify License Servers** – Execute the iFOR/LS test and verification tool. (verify servers)
- **Manage Nodelocked Licenses** – Allows the user to add, delete, and display nodelocked licenses on a system. (manage_nodelocked)

The `/usr/lib/netls/conf/nodelock` file contains a list of nodelocked licenses for this machine (or node).

The following options are available:

- **Add a Nodelocked License from a File** – Add a license to the `nodelock` file from a file that you specify. This is the preferred method to add a nodelocked license, because it allows you to cut and paste from AIXterm window.
- **Add a Nodelocked License from the Keyboard** – If you have a nodelocked license to add but it is not in a file, you can type it in here. Be careful to type it in as given; a single error will make it unusable. If possible, obtain the nodelocked license in written form; for example by means of a fax.
- **Delete a Nodelocked License** – Delete a nodelocked license from the `nodelock` file. Once this is done, applications that depend on the deleted license may not work.
- **Show Nodelocked Licenses** – Lists the contents of the `/usr/lib/netls/conf/nodelock` file.
- **Show Target ID** – Show the target ID of the current machine. This information is needed when ordering nodelocked licenses. (show_target_id)

Tasks Requiring Commands

The following tasks can be performed only through either operating system commands or iFOR/LS tools:

- Maintaining the `glbd`, `llbd`, and `netlsd` daemons (on page 3-18)
- Creating and maintaining a user file to specify who may use the product (on page 3-18) (optional)

Adding or Deleting Nodelocked Licenses

There are two kinds of nodelocked licenses:

Single license

Each password specifies a single nodelocked license. Vendors usually deliver nodelocked licenses as a set of single-license passwords that are installed individually by editing the `nodelock` file at each node to be licensed.

Dynamic

Each password specifies parameters for the creation of several single-license licenses. Dynamic licenses are usually issued for products that are licensed on a nodelocked basis and that are to be installed on a large network. These licenses allow the product to make a system call to a server when a single-license license is not available.

If only a few nodes are to be licensed, it doesn't take long to edit the `nodelock` file at each node manually, adding a single-license license at each node. If many nodes need to be

licensed, dynamic licenses can automate the otherwise tedious task of editing perhaps hundreds of **nodelock** files. This is known as dynamic nodelocking.

Dynamic nodelocking works as follows:

1. When a user invokes the nodelocked product, the product looks for a nodelocked license in the file **/usr/lib/netls/conf/nodelock** file.
2. If there is a nodelocked license, the software product is made available to the user. If there is not a nodelocked license, the product requests one from the license server daemon and passes the target ID of the node on which the product is running.
3. Assuming there is a dynamic license installed at the license server for the requesting product, the server creates and passes back to the requesting node a single-license license, the target of which is the ID of the requesting node. The license server then decrements by one the number of nodelocked licenses specified by the dynamic license.
4. The product automatically installs the single-license license in the **nodelock** file of the node on which it is running.

This procedure is repeated until all nodes have installed nodelocked licenses, or until all nodelocked licenses specified by the dynamic license are used up. Although you need the license server to install a dynamic license, once the single-license it specifies has been installed, you no longer need the license server to run the product.

Tips on Adding a License to the nodelock File

The following information can help you avoid making mistakes when adding a license to the **nodelock** file:

- The clock on your machine should be set to the correct time. Otherwise, the license may not be recognized as active.
- If you can cut and paste a license into the file using AIXwindows, you can easily avoid any typing errors.
- Licenses are case-sensitive. All letters are lowercase.
- Do not confuse the number one (1) with a lowercase L (l)
- Do not confuse the number zero (0) with an uppercase letter O.
- Do not attempt to substitute a single quote (') for a double quote (") in the license.
- Licenses cannot be split by a carriage return.
- The **nodelock** file must have permissions set as 644, so that everybody can read it. Issue the following command if its permissions are not correct:

```
chmod 644 /usr/lib/netls/conf/nodelock
```

Procedure

The software vendor should inform you of whether you are getting single-license or dynamic licenses for your nodelocked licenses. Then do one of the following:

- To add dynamic licenses, follow the instructions in “Adding or Deleting Concurrent-Use and Use-Once iFOR/LS Licenses” on page 3-6. The product must be invoked at least once at a specific node for the license to be enabled at that node.
 - To add single-licenses, follow the instructions below, or use SMIT (**smit manage_nodelocked**).
1. Use **ls_targetid** or SMIT (**smit show_target_id**) to find the system ID for your workstation. The syntax for **ls_targetid** is:

```

/usr/lib/netls/bin/ls_targetid
PREFERRED Target ID (CPU Planar ID)
-----
55004748
ALTERNATE Target ID (Link Level Address)
-----
EE2F0F23

```

The target ID (typically the CPU planar ID) is derived from the command `uname -m`; the link level address is the link level address of the LAN adapter.

2. Supply this response information to your software vendor to receive the license password. The password will arrive by mail, e-mail, or fax.

When you have the returned information:

1. Log in as root (or use the **su** command).
2. Create or edit the file **/usr/lib/netls/conf/nodelock**.

If the licensed product is the first iFOR/LS **nodelock** file on your system, you must create this file as root. This file could initially have entries similar to the following two lines:

```

# nodelock example for the licensed product expires 12/25/94
543b0f87c093.02.81.87.92.34.00.00.00 gganccupqb5d8auxabdsw2a "" "2.0"

```

The first line starts with a comment character, #, and is included for information only. It represents the name of the product and its license expiration date. The second line is the actual product license. The first number represents the system's vendor ID. Following the space is an example of a long alphanumeric *password* that enables the nodelocked license. The next field (" "), currently blank, is an optional field that can contain up to 80 characters. Known as the annotation field, it is used by the application developer to describe any unique enablement options of the license. The last field ("2.0") is the version field.

The password generated for your system will, of course, be different.

3. It is a good idea to include a comment line above the license to help you identify it in the future. Also, it is best to include the full product name, version, and expiration date (if any), since there may be other nodelocked software on the same computer.
4. Double-check the information to ensure it is the same as that supplied by the vendor. Then try the product. If all is as it should be, the product will function properly and you will not have to worry about it again.

Example

A vendor uses a command called **ls_dpass** to generate your license. Output from the command for a nodelocked license will look similar to the following:

-----NLSPASS-----

```
Vendor Password agkrmbnthcsc4
built as
  vendor name : Common
  vendor id   : 5ce6f0617216.02.81.23.85.e3.00.00.00
  target      : 1500910
```

```
Product Passwords built as
  name        : demo
  id          : 1926
  version     : 3.3
  license type : Nodelocked
```

```
Product password for 0x1500910 server :
wkseneaanmw6u6wjmsjp65t6w built as
  start date   : 04/08/94
  duration     : 30
  expiration   : 05/07/94
  target type  : AIX
  target       : 1500910
Add the following line to your nodelock file :
5ce6f0617216.02.81.23.85.e3.00.00.00 wkseneaanmw6u6wjmsjp65t6w "" "3.3"
-----
```

Your file might look like this after such an entry:

```
#Test Product Version 3.3. Valid 03/12/94 - 03/11/95
9ce6f0617216.02.81.23.85.e3.00.00.00 89pbxmywy8qh9mdu97u8yb586naa "" "3.3"

#Test Product Version 4.1. Annotation: "abc". Valid 04/08/94 - 06/01/94
90b82be84513.02.81.23.86.fb.00.00.00 26z8g3vnvdsn8b4u3n8bquspefmsnamj "abc" "4.1"

#Common "demo" Product Version 3.3 Valid 04/08/94 - 05/07/94
5ce6f0617216.02.81.23.85.e3.00.00.00 wkseneaanmw6u6wjmsjp65t6w "" "3.3"
```

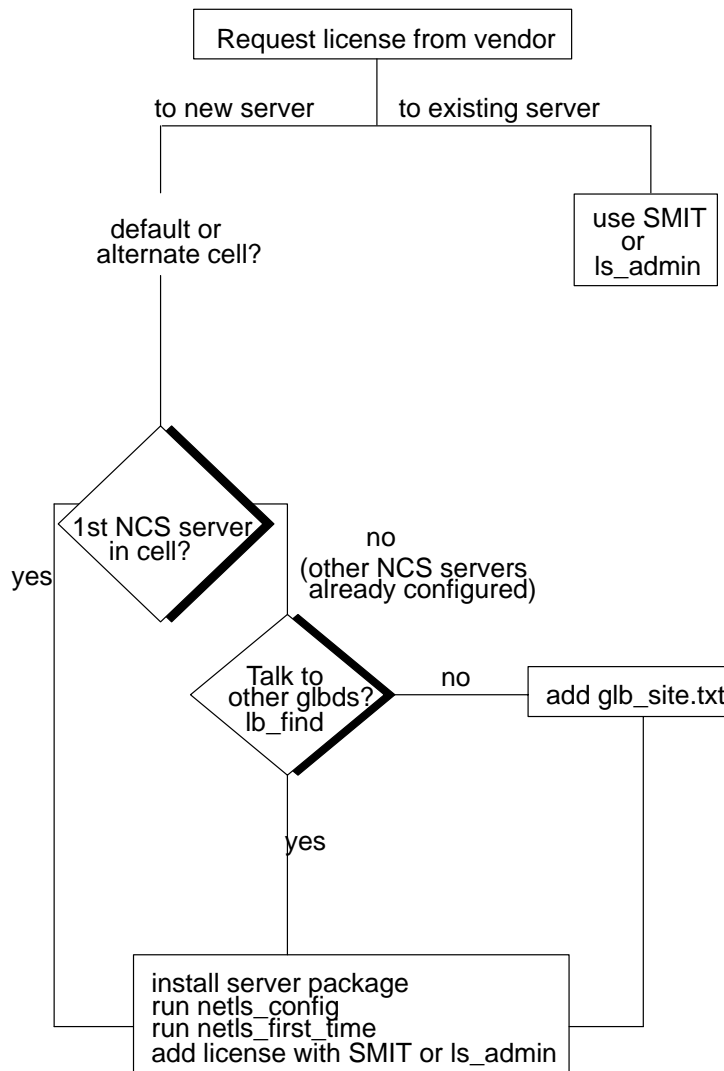
In Case of a Problem

If a node with a nodelocked license will not allow an end user to use the corresponding product, refer to "Troubleshooting Nodelocked Licenses" on page 4-1.

Adding or Deleting Concurrent-Use and Use-Once iFOR/LS Licenses

After verifying the operation of the iFOR/LS license servers, you are ready to request and to install concurrent-use and use-once licenses. The following flowchart shows the process for adding concurrent-use or use-once licenses:

Adding Concurrent-Use and Use-Once Licenses



To make additions to the database of vendors and products maintained by each license server, you can use either SMIT or **ls_admin**.

Usually, the vendor delivers the licenses at about the same time as the product, either by mail, fax, or telephone. This section explains how to use the **ls_admin** tool to add the vendor ID and password as well as the product passwords to the license databases so that users can begin using the product.

Prerequisite

You must have obtained the target IDs (typically, the CPU planar IDs) of the iFOR/LS license server nodes that will grant the licenses. To obtain the target ID for a node, run the **ls_targetid** program with this syntax:

```
/usr/lib/netls/bin/ls_targetid  
iFOR/LS Target ID  
-----  
55004748
```

The response will be different for each node, of course.

Procedure

1. Request licenses by providing the product vendor with the system ID and number of licenses you require. The vendor will use the information to create a vendor password and a product password for each node that will act as a license server. The vendor password is an encrypted key that contains information about the system vendor; once you have added a vendor password to your license database, you can add any product password from that vendor. The product password is an encrypted password containing information about the product and how you can use it (for example, number of licenses, their type and duration).

A vendor uses a command called **ls_dpss** to generate your license. Output from the command for concurrent-use license will look similar to the following:

```
-----NLSPASS-----  
Vendor Password agkrmbnthcsc4  
built as  
  vendor name : Common  
  vendor id  : 5ce6f0617216.02.81.23.85.e3.00.00.00  
  target     : 1500910  
ls_admin -a -v "Common" 5ce6f0617216.02.81.23.85.e3.00.00.00 agkrmbnthcsc4  
  
Product Passwords built as  
  name       : demo  
  id         : 1926  
  version    : 3.3  
  license type : Concurrent-use  
  
  Product password for 0x1500910 server :  
s8yewwjtbfh68jf4wewbvuf3jiaaa  
built as  
  start date   : 04/08/94  
  duration    : 90  
  expiration  : 07/06/94  
  number      : 1  
  target type : AIX  
  target      : 1500910  
ls_admin -s -p "Common" "demo" s8yewwjtbfh68jf4wewbvuf3jiaaa "3.3"
```

The **ls_admin** command used to add the license is listed below the information for each license. You can cut and paste this command directly to the command line.

2. When the information is returned from the vendor, log in as root (or use the **su** command).

3. Run the license administration program, **ls_admin**, which can be run with a graphic user interface by starting **ls_admin** from within an AIXterm window or run in command line form. For a sample **ls_admin** session, see page 3-8.

You can install the licenses for all nodes that allow remote servers to manage them from any one of the license server nodes. When a license server is set up, it can choose whether to allow remote iFOR/LS administrative control. The default is to ignore remote iFOR/LS administrative commands.

The first time a new vendor's product is installed in the license servers, the vendor ID and product ID must be entered. For subsequent license entries, this information will already exist.

A Sample **ls_admin** Session

Suppose you want to add concurrent-use passwords for a new product to two license server databases. Also suppose that the decision has been made to have the 50 licenses divided equally between servers **cobweb** and **apo11o**, and the vendor, Modern Solutions, is so informed. Modern Solutions then sends the following information:

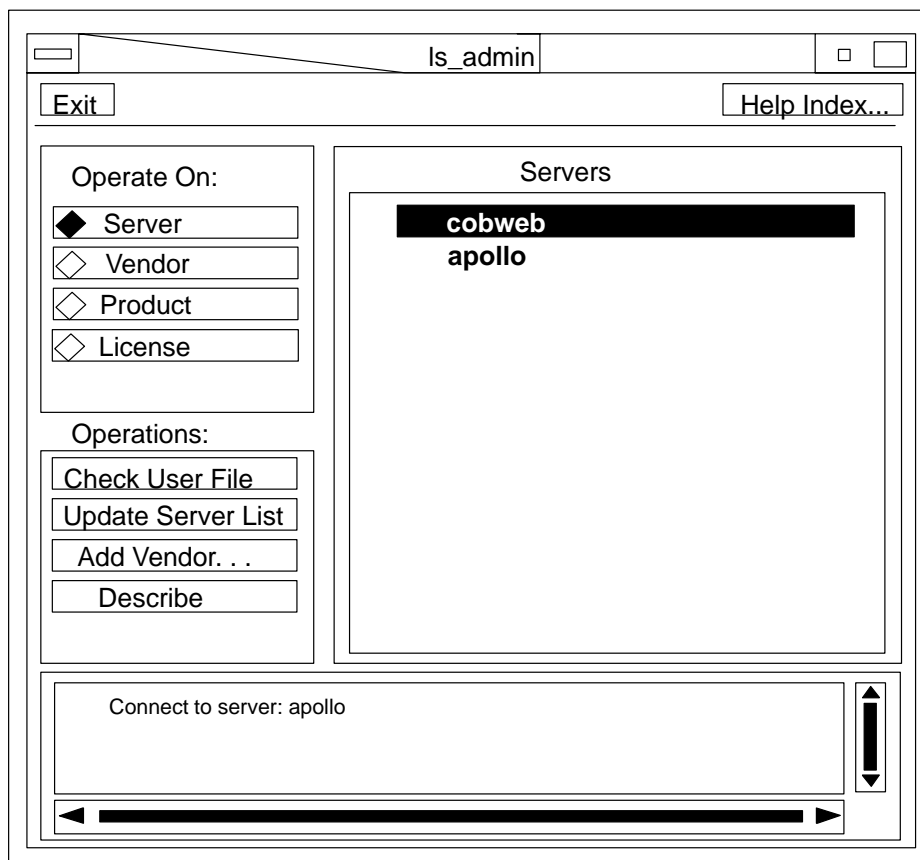
- Vendor: Modern Solutions
- Vendor ID: 3750e6cc9000.0d.00.00.5c.d3.00.00.00
- Vendor password: dasghgxt3yhk2
- Product/version: Kwik-Draw, Version 2.1
- Total licenses: 50 concurrent-use licenses from 12/10/90 to 12/9/91
- Server a5ef password: 53g55mqt5xmxcgiaacu89f2 (specifies 25 licenses)
- Server 4ce3 password: 53g55mqt5xmxcgiaabgqhf2 (specifies 25 licenses)

Assume license servers are already running on **apo11o** and **cobweb**. The vendor ID and passwords can be added either before or after installing Version 2.1 of the *Kwik-Draw* software.

Note: The following sample session is intended as a model for using **ls_admin**, not as an exercise. If you attempt to duplicate this sample session using the data supplied, *it will not work*.

First, start up **ls_admin**:

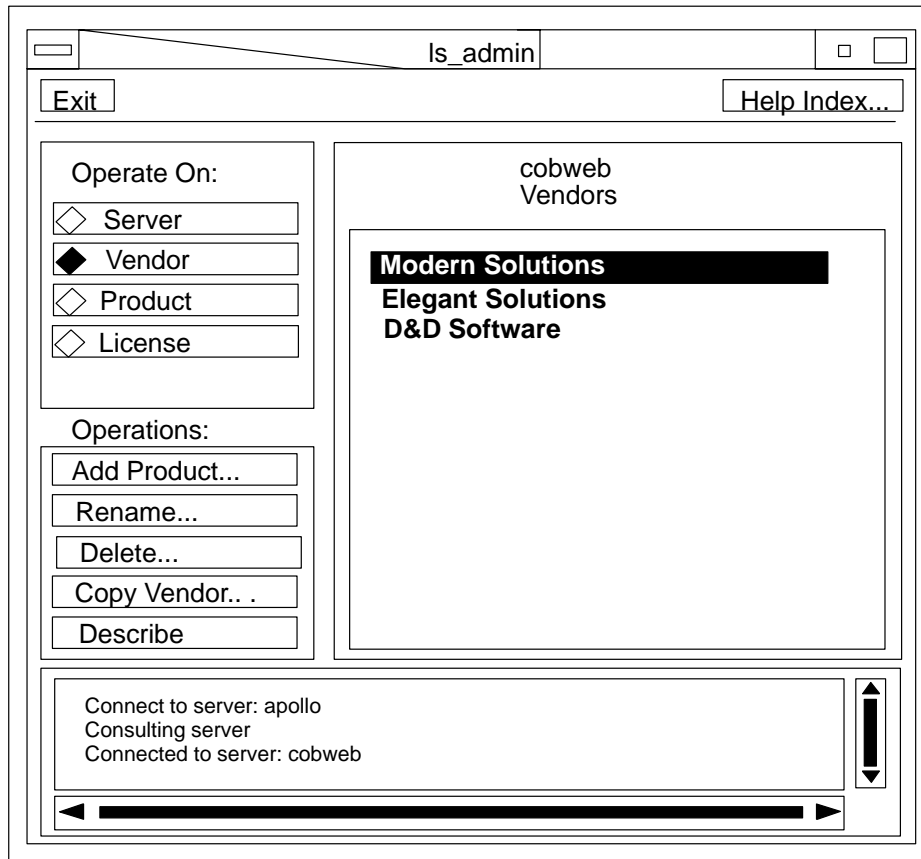
1. At the server node **cobweb**, change directory to the **/usr/lib/netls/bin** directory. Type **ls_admin** and a display like the one in the figure "Example of the **ls_admin** Startup Display" appears. (Depending on the type of node you are using, your display may look slightly different; but it works the same way.) Notice that the local server (**cobweb**) is selected by default. (If you want to see the node ID of the selected license server node in the data display area below the **ls_admin** window, select **Describe** by clicking the left mouse button on it.)



Example of the ls_admin Startup Display

2. Now you need to add the vendor, Modern Solutions, to the server's vendor list. (If you had previously installed a product from this vendor, skip this step and just select **Vendor** to display the Vendor list, and then the vendor name, before continuing to Step 3.)

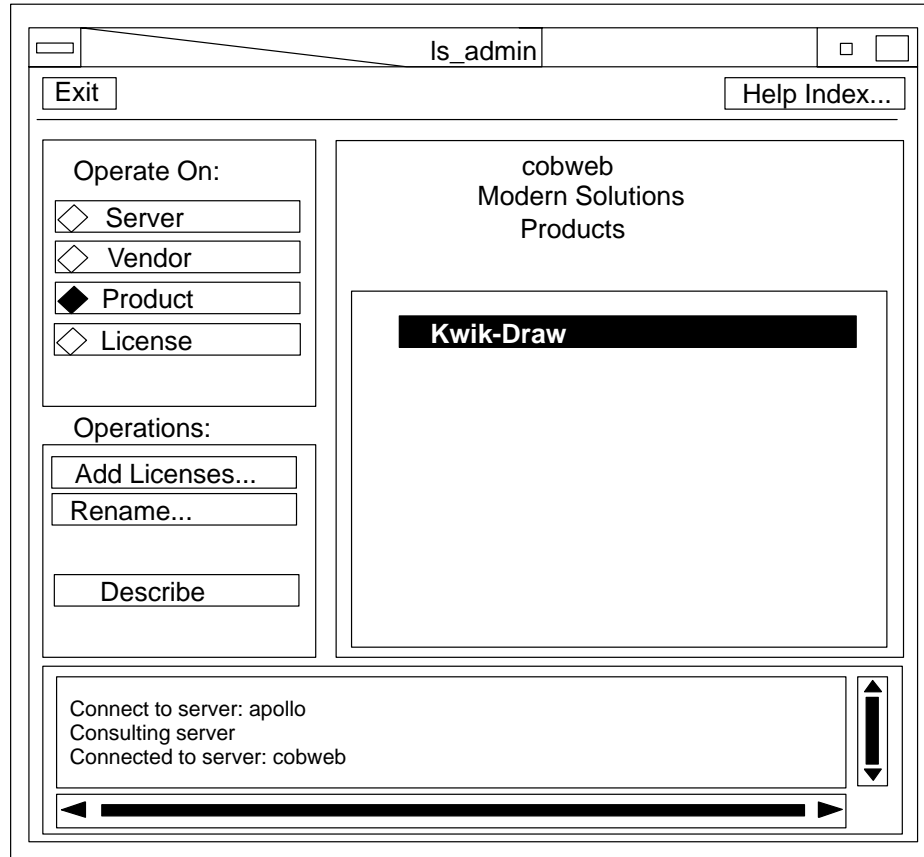
Use the left mouse button to select **Add vendor** from the Operations menu. Add the vendor name in the first field of the pop-up that appears, and then press Enter to enter the data. Add the vendor ID and vendor password in the same manner, pressing Enter after entering data in each field. Check to be sure that the vendor ID and password are correct, and then select the **Add vendor** button on the pop-up. When the **ls_admin** display reappears (see "Adding to the Vendor List"), notice that Vendor has become the selected object type and that the vendor Modern Solutions appears in the list of vendors at cobweb.



Adding to the Vendor List

- Now that you have added the vendor and the vendor's name is selected in the vendors list, the next step is to add the Modern Solution's product called Kwik-Draw and the password specifying the licenses, to the vendor's product list.

Select **Add product** from the Operations menu. On the pop-up that appears, add the product name, version, and password for server cobweb, pressing Enter after completing each field. (For this sample session, there is no license annotation, so skip the annotation field.) Then select **Add product**. Now when the **ls_admin** display reappears (see "Adding to the Product List"), notice that **Product** has become the selected object type and that the product Kwik-Draw is selected on the list of products for the vendor Modern Solutions at `cobweb`.



Adding to the Product List

At this point, you have added the vendor Modern Solutions, its product Kwik-Draw Version 2.1, and a password specifying 25 licenses to the license database at `cobweb`. The next step is to copy the vendor to the other server, `apollo`.

4. Select **Vendor** from the Operate On menu, and select the vendor Modern Solutions, Inc., from the list of vendors.
5. To add the vendor data to the server `apollo`, select **Copy vendor** from the Operations menu, then select `apollo` from the list of servers that appears.

Now that Modern Solutions has been added to the vendor list for `apollo`, you complete your task by adding the product Kwik-Draw and a password specifying 25 licenses to the license database for `apollo`.

6. Select **Server** from the Operate On menu, and select the server `apollo` from the list of servers at the right of the display.
7. Select **Vendor** from the Operate On menu, and select the vendor Modern Solutions from the list of vendors.
8. Select **Add product**. Add the product name, version, and password for server `4ce3`, then select **Add product** on the pop-up.

Managing License Servers

The following section discusses the **ls_stat** (license server status) and **ls_rpt** (license server report) commands, which enable checking of the license-server installation status

and display the history of license-server events. See Appendix A for complete reference information on these commands.

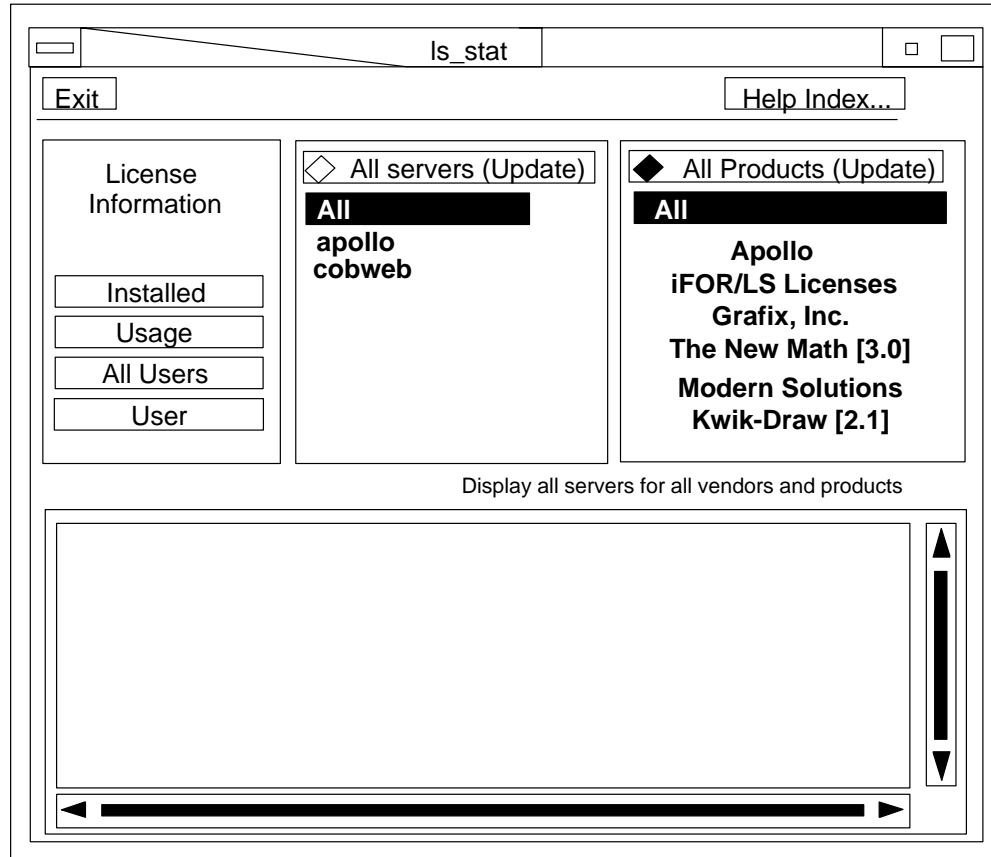
Getting Status Information with **ls_stat**

The **ls_stat** command provides both users and administrators with status information on product licenses. The information **ls_stat** provides includes:

- Lists of the **netlsd** servers on the network
- Lists of licensed products at all servers or at selected servers
- Information, listed by vendor, product and server, about current users of licensed products, including user ID, node name, group, number of licenses held, and start time
- Information, listed by vendor, product and server, about product licenses installed at selected servers, including number of active licenses, their start and end dates, their type, the number of licenses currently in use, and the length of the queue of users waiting for licenses
- Information, listed by vendor, product and server, about the usage of products, including licenses in use, total number of licenses, and licenses available

When **ls_stat** is invoked, a display such as that shown in the figure “ls_stat, Step 1” appears. (Depending upon the type of node you are on, the display may look slightly different, but it works the same way.) If not all of the display is visible, use the mouse to enlarge the window.

This section introduces the **ls_stat** graphical interface. The description of **ls_stat** in Appendix A provides reference information on the graphical interface and on the alternative command-line interface.



Is_stat, Step 1

At the top of the interface are the **Is_stat** configuration windows. Beneath the **Is_stat** configuration windows is a larger transcript window in which the data requested is displayed. (Messages are also displayed in this window.) The **Is_stat** configuration windows contain the following objects:

Exit Button Select (click the left mouse button on) this button to exit from **Is_stat**.

Help Index... The **Help Index...** button provides access to the online help system.

License Information Menu

This menu contains four buttons: **Installed**, **Usage**, **All Users**, and **Users**. Select these buttons to display information about users, installed licenses, and usage of the selected server and product.

Server List Box

This list box, directly to the right of the License Information menu, displays the server list. The **All Servers (Update)** button at the top of this box is explained below.

Product List Box

This list box, directly to the right of the Server List box, displays the server list. The **All Products (Update)** button at the top of this box is explained below.

All Servers (Update) Button

This button polls the network and updates the server list. When this button is selected, a small box on the left side is darkened, indicating that:

- All existing servers are displayed in the Server List box.

- The vendors and products listed in the Product List box are the vendors and products existing at the server currently selected in the Server List box.

All Products (Update) Button

This button polls the network and updates the product list. When you select this button, a small box on the left side is darkened, indicating that:

- All existing vendors and products are displayed in the Product List box.
- The servers listed in the Server List box are the servers that hold licenses for the product or vendor currently selected in the Product List box.

Status Message Field

This field, across the bottom of the window, describes the information currently displayed in the Server List box and the Product List box.

A Sample Is_stat Session

Suppose that, having installed licenses for Kwik-Draw, Version 2.1, we decide to check the license databases at `apollo` and `cobweb` to verify the installation and to find out if anyone is using that product. To do this, we run the license server status program **Is_stat**.

Note: The following sample session is intended as a model for using **Is_stat**, not as an exercise. If you attempt to duplicate this sample session using the data supplied, it will not work.

1. First, invoke **Is_stat** in an AIXterm window to display the **Is_stat** window.

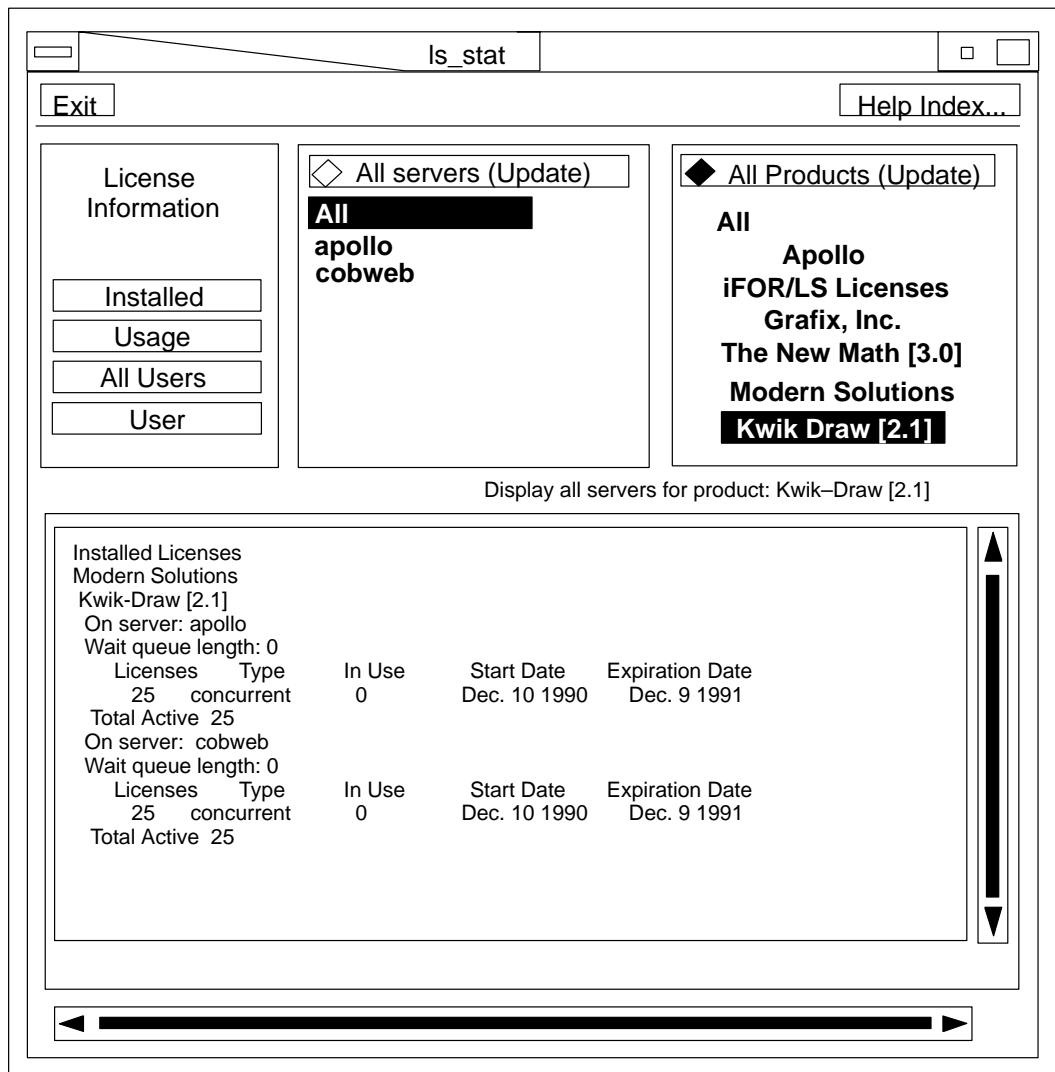
Notice that, by default, all servers and all products are listed. The display shows three vendors, each with one installed product: Apollo and its product, NetLS Licenses, Version 2.0; Grafix, Inc., and its product, The New Math, Version 3.0; and Modern Solutions and its product, Kwik-Draw, Version 2.1.

Note: Products are distinguished from vendors by the square brackets that enclose the product version text.

2. Now check to see if the licenses for Kwik-Draw have been correctly installed.

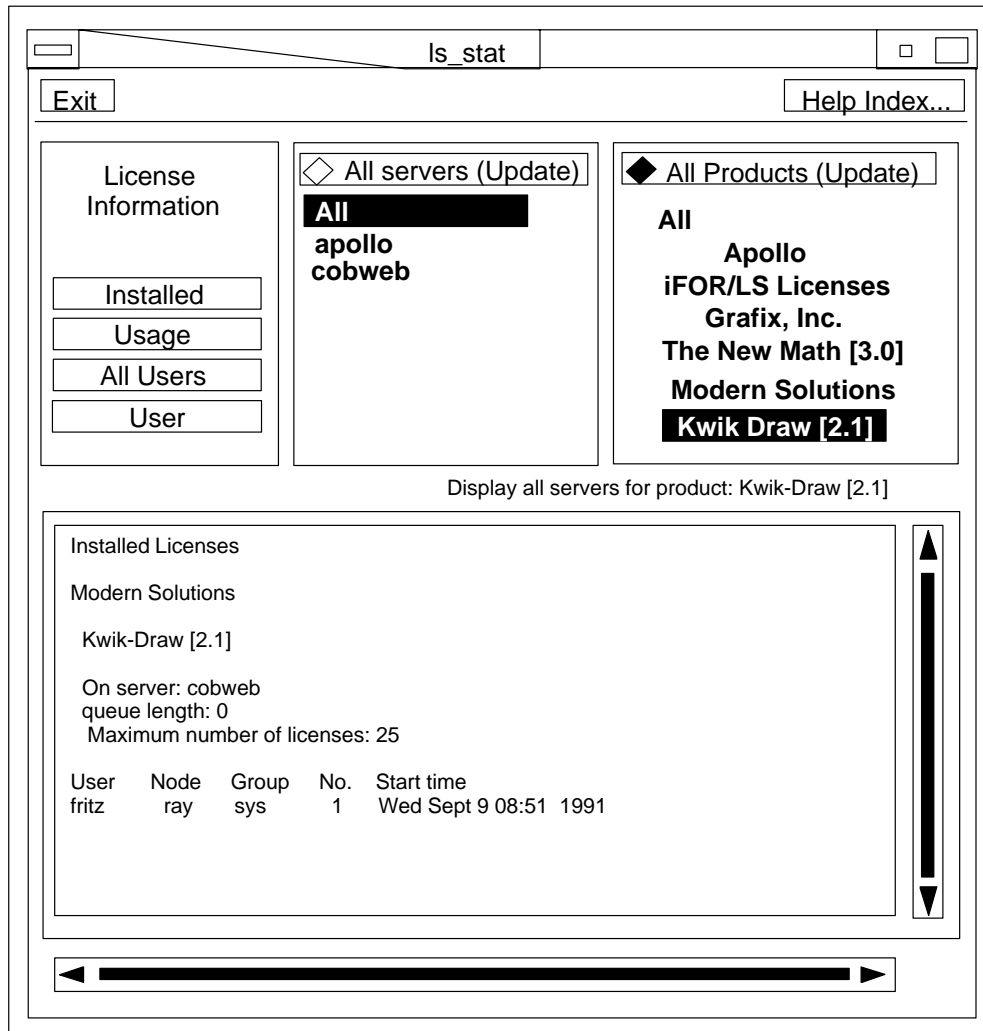
Use the left mouse button to select **Kwik-Draw [2.1]** from the product listing, then select **Installed**. The information in the transcript pad, shown in the figure “Is_stat, Step 2” shows that 25 licenses were installed at `apollo`, and 25 at `cobweb`, and one license (at `cobweb`) is in use. No users are waiting in the queue at either `apollo` or `cobweb` to get licenses for this product.

Notice that the status message field has changed: only servers having licenses for Kwik-Draw are displayed.



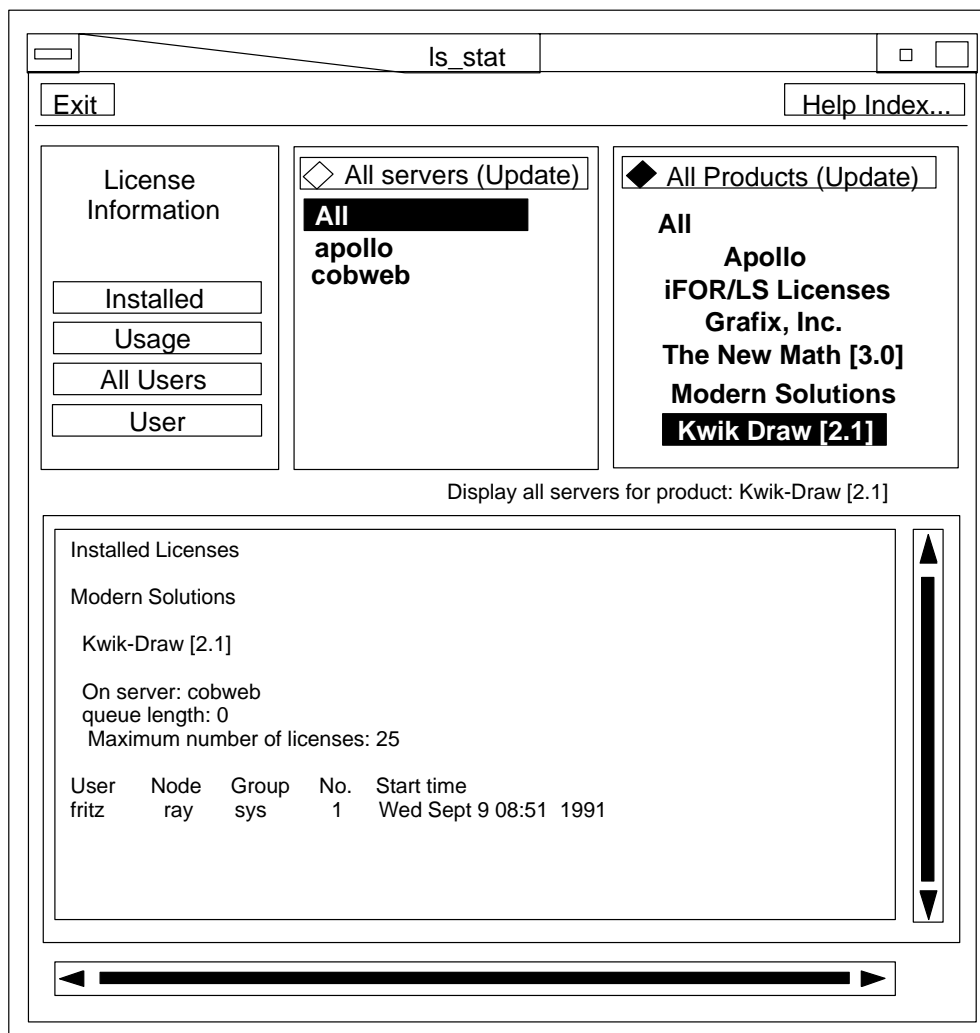
ls_stat, Step 2

- Next, see who is using the product. To find out who has the license, select **All Users**. As shown in the transcript pad (the figure "ls_stat, Step 3"), the user is `fritz` at node `ray`.



ls_stat, Step 3

- To find out usage of all installed products, select **All** from the Servers list and **All** from the Products list. Then select **Usage**. The display will look like the figure "ls_stat, Step 4":



ls_stat, Step 4

License Server Reports

The **ls_rpt** command generates reports on all kinds of license server activities. Different kinds of events can be reported (as well as all events), using dates, software vendors, products, and users as information filters. You need to determine which information is needed for **ls_rpt**; information is derived from the log file located in **/usr/lib/netls/conf/log_file**. Besides monitoring license server activities, license server reports can track demand for software products. This kind of information is useful for planning future purchases of licenses for those products.

You can also change what types of events are logged. For information on this, see "Choosing Events to Be Logged."

For complete information, see **ls_rpt** in Appendix A.

Choosing Events to Be Logged

The **netlsd** daemon logs certain events by default. After determining which events you no longer need, you must take down **netlsd** and restart it with the proper options. You must also change the startup script **/etc/rc.netls** on each machine to represent this change in the future.

The following is an example of a script entry that would start the license server without logging checkin, vendor, product, timeout, or message events:

```
startsrc -s netlsd -a "-no cvptm"
```

Deleting Stale Log Entries

If you have options set for **netlsd**, it is recommended that you periodically prune the log file by deleting old entries. There are two ways to do this:

1. You can delete the log entries written on and before a specified date. For example:

```
/usr/lib/netls/bin/ls_rpt -x 10/22/93
```

Note: No licenses will be granted until **ls_rpt -x** has finished executing. The server will restart properly. We recommend that you run this command during off-peak hours.

2. You can delete the entire log file by taking the server down, deleting **/usr/lib/netls/conf/log_file**, and then restarting the server. With this method, however, you will lose all entries.

Maintaining Daemons (llbd, glbd, and netlsd)

In the case of concurrent-use and use-once licenses, the license server uses the Network Computing System global location broker (GLB) to propagate information about the license server throughout a network. As the system administrator, it is your responsibility to monitor the location brokers and clean up unused or invalid entries.

If more than one global location broker is established in the network, use the **drm_admin** utility to check and maintain the network-wide location broker database. Make sure that the dates of the computers that global location broker daemons are running on are within two minutes of each other. Every week or so, merge the databases on all the global location broker daemons on the network.

In order to keep the location broker databases in sync, the iFOR/LS system administrator should periodically run **/etc/ncs/lb_admin** on one of the server machines, using the global and clean subcommands.

For more information about synchronizing location broker databases, see "Cleaning up Location Broker Databases" on page 4-7.

Creating and Maintaining User Files

License server user files are optional—they are necessary only if you want to give different users different access priorities, or prohibit them from accessing certain products.

A Sample User File

Below is a sample user file. In this example, the keywords are lowercase; they can be either uppercase or lowercase, but mixed-case keywords are not allowed. Comments are preceded by a percent sign.

```

%This line is a comment
crawl 10
%*****
vendor "Modern Solutions, Inc." "The New Math"
allow fritz -p 2 harry -p 1 monique -p 1 penny
%*****
vendor "Grafix, Inc." all
disallow heather jason
%*****
vendor "Unique Applications, Inc." dynamath
disallow bob
vendor "Unique Applications, Inc." versitex
disallow bob
vendor "Unique Applications, Inc." minigraph
disallow bob

```

Following are explanations of the keywords and their uses in the sample file.

```

crawl 10

```

In this section, the keyword **crawl** causes all users listed in the file to move up one level in priority after the product polls the queues of users waiting for licenses a specified number of times (in this case, 10 times). Using **crawl** makes sense only when different users have different priorities. The **crawl** keyword and user priorities apply only to products that use wait queues.

```

vendor "Modern Solutions, Inc." "The New Math"
allow fritz -p 2 harry -p 1 monique -p 1 penny

```

In this section, the user file declares the vendor Modern Solutions and its only installed product, The New Math. The vendor name and the product name contain spaces, and so are delimited by quotes. Four users may use The New Math: Fritz is a priority 2 user; Harry and Monique, priority 1 (the highest); the priority for Penny is unspecified, and so defaults to 3 (the lowest).

When a high-priority user requests a license for which low-priority users are already waiting, the high-priority user moves ahead of the lower-priority users in the queues. In this case, if Penny and Fritz are unsuccessful in obtaining a license and are willing to wait, **crawl** eventually changes their priority to 1. At that time, the declared priority 1 users (Harry and Monique) can no longer displace Penny and Fritz in the user queues.

```

vendor "Grafix, Inc." all
disallow heather jason

```

The keyword **all** means that all licensed products of this vendor (whatever those products may be) have the same user authorizations. In this case, everyone except Heather and Jason may use all Grafix software products.

```

vendor "Unique Applications, Inc." dynamath
disallow bob
vendor "Unique Applications, Inc." versitex
disallow bob
vendor "Unique Applications, Inc." minigraph
disallow bob

```

The lines in this section specify three products of Unique Applications, Inc.: Dynamath, Versitex, and Minigraph. (The product names are not delimited because the strings contain no spaces.) Note that the keyword **vendor** takes exactly one vendor and one product

declaration. The keyword **disallow** excludes only Bob from using the three products of Unique Applications, Inc. All other users are priority 3 by default.

That this user file specifies the products of Unique Applications individually probably indicates that Unique Applications has other installed products that anyone, including Bob, may use. If Dynamath, Versitex, and Minigraph were the only installed products of Unique Applications (and Bob was prohibited from using them) the user file would more likely use the keyword **all**, instead of specifying each product individually, as in the following example:

```
vendor "Unique Applications, Inc." all
disallow bob
```

You can use the **ls_admin** command to check that the format of a user file is correct; see Appendix A for more information.

Writing and Integrating User Files

If you decide to have a user file, it can be created by the use of any text editor. Put one copy on each license server node in the **/usr/lib/netls/conf** directory. Although you can write different user files for different server nodes or put user files on some nodes and not on others, such a policy might cause access to a product to vary depending upon which server is granting the license. The same user file should be placed on all server nodes in the interest of a consistent user authorization policy.

Maintaining User Files

When adding a new product or changing user priorities for existing products, remember to update the user files at all license server nodes accordingly. See Appendix B for detailed reference information on user files.

Chapter 4. Troubleshooting iFOR/LS

This chapter shows you how to troubleshoot both nodelocked and concurrent-use/use-once licenses.

If you want to know the implications of a specific error message or the general source of your problem, see one of the following sections:

- `ls_tv` Errors (page 4-4)
- NCS Problems and Solutions (page 4-5)

If you are experiencing other types of problems, check one of these sections:

- Common Customer Problems and Solutions (page 4-8)
- Problems Relating to Specific Configurations (page 4-11)

Troubleshooting Nodelocked Licenses

If a node with a nodelocked license will not allow an end user to use the corresponding product, check the following:

1. Verify that the name and path of the **nodelock** file are correct. The name should be: **nodelock**. The path should be: **/usr/lib/netls/conf**.
2. The permissions on the **nodelock** file should be set so that all users can read the file. You can check the file permissions by typing the command:

```
ls -l /usr/lib/netls/conf/nodelock
```

You should see output similar to the following:

```
-rw-r--r-- 1 root system 661 Apr 27 08:38  
/usr/lib/netls/conf/nodelock
```

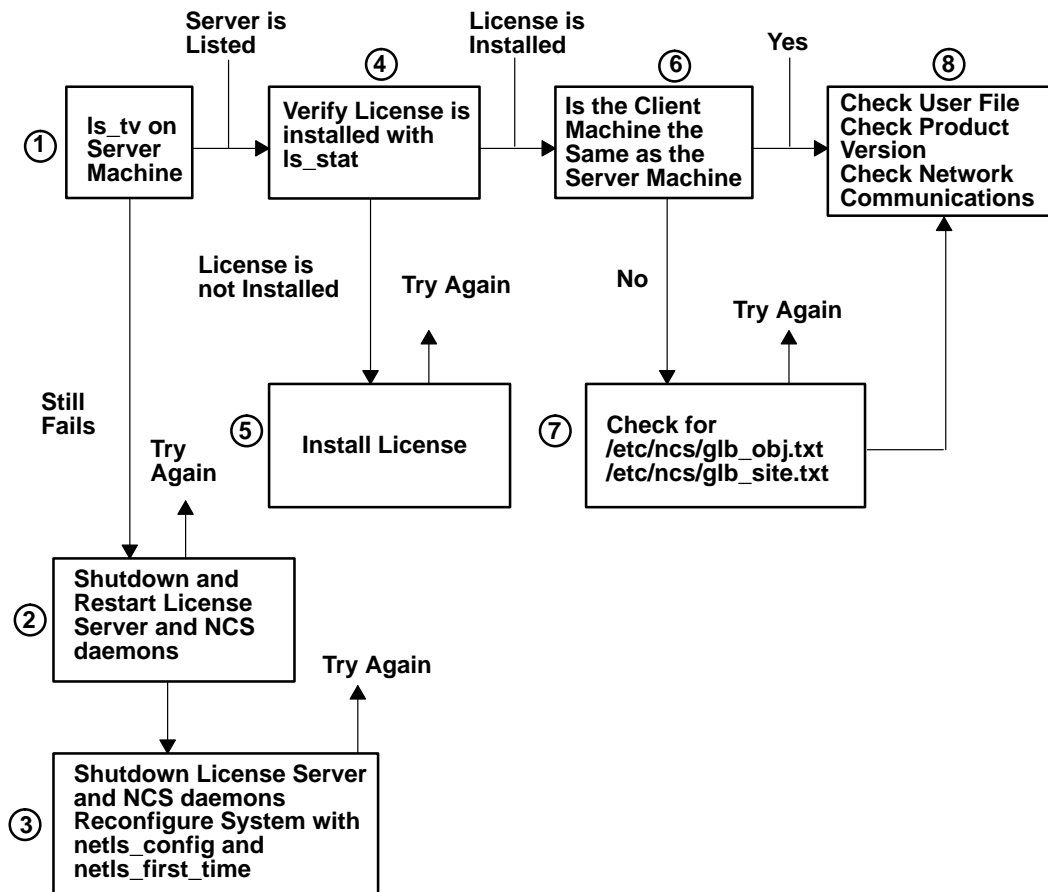
The `-rw-r--r--` at the beginning of the output line shows the correct file permissions. If this is not what you get as output, you should change the permissions by running the following command as the root user.

```
chmod 644 /usr/lib/netls/conf/nodelock
```

3. The date and time on your machine should be set correctly. Each nodelocked license has a start date and an end date built into it. If your machine date is set to the wrong date and time, the license may not be recognized as active.
4. The nodelocked license for your product must appear in the **nodelock** file exactly as it appears on the fax, e-mail, note, or other source in which it was delivered to you. Pay special attention to the following:
 - Licenses are case-sensitive. All letters are lowercase.
 - Do not confuse the number one (1) with a lowercase L (l)
 - Do not confuse the number zero (0) with an uppercase letter O.
 - Do not attempt to substitute a single quote (') for a double quote (") in the license.
 - Licenses cannot be split by a carriage return.
5. Verify that the license you received really is the correct license for the software you are running. The nodelocked license must be for the same version of the application that you are running.

Troubleshooting Concurrent-Use and Use-Once Licenses

If a node with a concurrent-use or use-once license does not allow an end user to use the corresponding product, you should perform the steps illustrated in “Troubleshooting Roadmap”. Each step is discussed in detail below. The machine where the licensed application is running is referred to as the client machine. The machine where the iFOR/LS license server is running is referred to as the server machine.



Troubleshooting Roadmap

1. Make sure that the license server and the required daemons are running on the server machine.
 - a. Go to the server machine where you have your licenses installed.
 - b. Run the command: `/usr/lib/netls/bin/ls_tv`

The `ls_tv` command returns a list of all servers from which your machine can get licenses.

If a list of servers is returned and your machine is one of them, go to step 4.

If errors are returned, or if your machine is not listed as a running server, go to step 2, or refer to “ls_tv Errors” on page 4-4 to further debug the problem.
2. If you received an error from `ls_tv`, the problem is probably due to a failure to start all the required daemons.

Shut down all the daemons normally required for the license server to work by running the command:

```
/usr/lib/netls/bin/srv_stop
```

The daemons can also be stopped by executing the following commands in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

Restart the daemons by using the command: `/usr/lib/netls/bin/srv_start`

The daemons can also be started by executing the following commands in the order shown:

```
startsrc -s llbd
startsrc -s glbd
startsrc -s netlsd
```

Go to step 1 to verify that your license server is now running. If your license server still does not respond, go to step 3.

3. At this point the easiest thing to do would be to reconfigure your license server.

- a. Shut down the license server and the NCS daemons by executing the command:

```
/usr/lib/netls/bin/srv_stop
```

The daemons can also be stopped by executing the following commands in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

- b. Refer to “NCS Problems and Solutions” on page 4-5 to determine what reconfiguration procedure is best for your machine.

Go back to step 1 to verify proper configuration.

4. Check to see if the specific license you need is installed on your server machine. Use SMIT, or run the command:

```
/usr/lib/netls/bin/ls_stat -i -n Server
```

Server is the host name of your machine.

The output of the **ls_stat** command lists all the installed licenses on your license server. The output also shows the license availability and usage. If license requests were not being granted for an application, it may have been because all the licenses were in use.

If the license you need is not installed, then go to step 5. If licenses are installed and available, then go to step 6.

5. If the license you need is not installed on the server, then you need to install the license. Refer to “Adding or Deleting Concurrent-Use and Use-Once Licenses” on page 3-6. Then go back to step 4 to verify that the licenses are now accessible.
6. Go to the client machine and try to run the application. If your client machine is not the same as your server machine, then go to step 7. If your client machine is the same as your server machine, then go to step 8.
7. If your license requests are still not being granted, then one or more of the following is probably true.

- a. The client machine may be in a different NCS cell than the license server. To join the same cell as the license server, look for the file `/etc/ncs/glb_obj.txt`. If the file exists on the server machine, copy it to the client machine, even if it already exists on the client machine. If the file exists on the client machine, but does not exist on the server machine, then delete it from the client machine. Refer to “Establishing a New NCS Cell” on page 5-5 for more details.
- b. The client machine may be in a different communications subnet than the global location broker (GLB), and cannot contact the GLB through the normal broadcasts (the GLB is used to help the client talk with the license server).

Create the `/etc/ncs/glb_site.txt` file on the client machine to direct communications to the GLB. Refer to “Establishing a New NCS Cell” on page 5-5 for more details.

- 8. Your license requests should now be granted properly by the server. If the licenses are still not granted by the server, verify the following:
 - a. There does not exist a user configuration file restricting access on the server machine. Refer to “Creating and Maintaining User Files” on page 3-18 for more information.
 - b. The product version of the installed license is the same as the product version of the application you are attempting to run.
 - c. Normal TCP/IP communications can occur between the client machine and the server machine.

Is_tv Errors

As you run the iFOR/LS test and verification utility, `/usr/lib/netls/bin/ls_tv`, you may receive messages. While most messages are documented in “iFOR/LS (netlsd) Error Messages and Recovery Information” on page 4-17, the three most common messages generated through the use of `Is_tv` are:

1d010001 486604801 netls_lic_not_fnd : License not found in database (network license server/server)

EXPLANATION: The product license requested by an application is not registered within the database.

ACTION:

- 1. Use the `Is_stat` command to determine whether the needed product license is installed on a reachable server.
- 2. Check the `nodelock` file and determine whether a license for the requested product is installed locally.

1d020006 486670342 netls_no_svrs_fnd : No servers available for this vendor (network license server/library)

EXPLANATION: No license servers were found that have licenses for the application.

ACTION:

- 1. Check the cell configuration of the machines running the applications and license servers. The machines should be in the same cell.
- 2. Use `Is_admin` to verify that the licenses needed are installed on a license server in the cell.

3. Install licenses for the application.

1d010014 486604820 netls_bad_timestamp : Time disparity too large (network license server/server)

EXPLANATION: The system date and time on the license server is too different from the system date and time on the machine where the application is running.

ACTION: Resynchronize the node clocks on the two machines to within two minutes of each other.

NCS Problems and Solutions

The iFOR/LS license server uses NCS as its current technology for distributed computing (see “Distributed Computing with NCS” on page 1-6 for more information). Occasionally, a failure in NCS causes iFOR/LS to degrade in performance or fail altogether. It is often the case that an iFOR/LS “problem” is actually a problem in the state of NCS at that instant. There are several ways to “clean up” NCS and return the system to a stable, smooth-running operation. The cleanup method you use will depend on the size and complexity of your network.

Implications of NCS Communications Failures

If the server fails, the following occurs:

- The current license transaction is stored on disk.
- The server will return to same state when restarted.

If the client fails, the server will reclaim license after the check period expires.

If the network fails, the following occurs:

- The server will reclaim the license after the check period expires.
- The application developer determines client behavior.

General Hints on Handling NCS for iFOR/LS

Although it is not absolutely required, you should always run the global location broker daemon (**glbd**) on the same machine as the iFOR/LS license server (**netlsd**). The local location broker daemon (**llbd**) must also be running on this machine.

When you configure or reconfigure the NCS cell for your iFOR/LS license server, always use the **netls_config** script. When running **netls_config**, if you are asked whether you want to use an existing global location broker database that is already installed, always answer no. The **netls_config** script will ensure that your new global location broker is started and configured properly.

Note: After running **netls_config**, remember to run **netls_first_time**.

After your iFOR/LS server has been configured for the first time, always stop your daemons by either using the **srv_stop** command or by using the following commands in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

Likewise, always restart the daemons by either using the **srv_start** command, or by using the following commands in the order shown:


```
startsrc -s llbd
startsrc -s glbd
startsrc -s netlsd
```

If you do not stop and start the daemons as described above, the location broker databases can become corrupted. This is the most common cause of delayed responses when requesting licenses.

Quick and Easy NCS Cell Fixup

If your iFOR/LS server is not working and you are seeing NCS communications errors, the easiest way to bring your server back up is to just reconfigure it from the beginning. You would do this by running the **netls_config** script, followed by the **netls_first_time** script, as described in “Post-Installation Configuration” on page 2-5.

This method is best for any of the following situations:

1. You are running a standalone (non-networked) server.
2. You are the only server in the cell, and the cell has few client machines.
3. You are a member of a cell that has other servers with replicatable global location brokers running.

When running **netls_config** (in `/usr/lib/netls/conf`), you may be asked if you want to use an already configured global location broker database. Answer “no” to this question. You want to create a new database that you know will have the correct information in it. With **netls_config**, you can cleanly start your server in a new NCS cell, or you can join an existing NCS cell. Either way, with **netls_config**, your database should be recreated to avoid possibly bringing a corrupted database into the reconfigured system.

Keep in mind that if you reconfigure your machine to start in a new alternate cell, you will need to copy the newly generated `/etc/ncs/glb_obj.txt` file from the server machine to all of its client machines. Keep in mind also that after you run **netls_config**, you must run **netls_first_time** (also in `/usr/lib/netls/conf`).

Manual NCS Cell Fixup

If your iFOR/LS license server machine is the only server machine in an alternate cell with many clients, it may not be practical to rerun the **netls_config** script. Doing so would regenerate the `glb_obj.txt` file on the server, and that would then need to be copied to each client machine. In such a situation, you can quickly reconfigure your NCS cell and restart your license server by following the instructions below. This will preserve your `glb_obj.txt` file, keeping your cell identifier the same so that it will not have to be changed on your client machines.

Note: These instructions are for reconfiguring a single server in an alternate cell. If this is not the only server machine in the alternate cell, or if you intend to run in the default cell, use the **netls_config** script, as described in the preceding section, “Quick and Easy NCS Cell Fixup.”

1. Stop the license server daemon and the location broker daemons by using the `/usr/lib/netls/bin/srv_stop` command, or by using the commands below in the order shown:

```
stopsrc -s netlsd
stopsrc -s glbd
stopsrc -s llbd
```

2. Delete any previously configured location broker databases by executing the following commands in the order shown:

```
cd /etc/ncs
rm glb.e glb.p glb_log
cd /tmp
rm llbbase.dat
```

3. Start the new local location broker by typing the following command:

```
startsrc -s llbd
```

4. Start the new global location broker by typing the following command:

```
startsrc -s glbd -a "--create -first -family ip"
```

5. Start the license server by typing the following command:

```
startsrc -s netlsd
```

6. Verify that your license server has restarted successfully by typing the following command:

```
/usr/lib/netls/bin/ls_tv
```

Cleaning up Location Broker Databases

When an NCS cell is running several global location brokers (GLBs), the database at each GLB node must be kept synchronized with the others. This allows any GLB in the cell to satisfy a location request by a client. Occasionally, a GLB will be removed, or a license server will be stopped without being shut down properly. This will usually result in invalid entries being left in the GLB databases. This is not normally fatal to the system, but it can introduce significant delays when applications attempt to get licenses for execution.

In such configurations, cleaning up the GLB databases can not be done by simply removing and adding back a server. The invalid entries must be removed by using the **lb_admin** tool, and the GLB databases must be resynchronized by using the **/etc/ncs/drm_admin** tool. Both tools are interactive. The instructions below will simulate the commands entered by a system administrator. See Appendix A for more information on how to use either **lb_admin** or **drm_admin**.

1. Start the **lb_admin** tool at one of the GLB servers. Use the commands shown below:

```
cd /etc/ncs
lb_admin
```

2. Set the object to be worked on to be the local location broker:

```
lb_admin: use local
```

3. Use the clean subcommand to remove any invalid entries:

```
lb_admin: clean
```

4. If prompted to remove entries, answer **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, answer **y**.

8. Exit **lb_admin** by using the quit subcommand:

```
lb_admin: quit
```

9. Start the **drm_admin** tool, which is also found in the **/etc/ncs** directory:

```
drm_admin
```

10. Set the object to be worked on to be the global location broker on your machine (replace *HostName* with your actual machine host name where you are running **drm_admin**):

```
drm_admin: set -o glb -h ip:HostName
```

11. Synchronize all the GLBs in the cell:

```
drm_admin: merge_all
```

12. If messages appear that declare that a host is unreachable, you should remove it from the global replica list (replace *HostName* with the actual host name of the machine that is no longer acting as a server):

```
drm_admin: purgerep ip:HostName
```

Note: If a host machine is purged from the replica list, it should no longer be running the global location broker daemon (**glbd**). If the global location broker needs to be run on this machine at a later date, it should be configured with the **netls_config** script, and joined to the cell.

13. Synchronize all the GLBs in the cell:

```
drm_admin: merge_all
```

14. Exit **drm_admin** by using the quit subcommand:

```
drm_admin: quit
```

Common Customer Problems and Solutions

I have NCS installed on the server machine but my clients do not have NCS. How can I make my clients get a token?

1. First, there needs to be at least one **llbd**, **glbd**, and **netlsd** running on the network.
2. Start the application, without doing any setup; you should be able to get a license. If you cannot get a license, it is possible that broadcasting is not working correctly.
3. Log in as root or issue the **su** command on the client machine.
4. On your client machine, create a directory called **/etc/ncs**.
5. Create a file called **glb_site.txt**.

Note: If an alternate cell was set up, you need to copy **/etc/ncs/glb_obj.txt** file as well.

6. Edit the file and enter the address family of the machine or machines running the servers. The following are sample **glb_site.txt** files for the IP address family:

```
ip:ren  
ip:#192.9.8.5
```

7. Run your application.

I am taking over for the system administrator, but I have no idea who did the installation for the product before. How can I tell what type of configuration they set up?

- Run **lb_find** on the system and check to determine if the **glbd** is replicated or not. You should see a GLB broker type, the network address of the server's host, a cell name (either default or alternate), and the UUID of the cell. You can also run **lb_find -q**.
- If nothing is running, you can check to see if a **glb_obj.txt** file exists in **/etc/ncs**. If it does, then it is most likely an alternate cell.

I have multiple products running, and during the installation of another product, netlsd tries to start. Will this produce a problem?

It should not. The **netlsd** will detect that an existing server is already running on this machine. If this is the case, the new server will exit gracefully and never start a process.

If you try to run another **llbd**, you will receive the error of LLBD: Unable to obtain any sockets. No process will start.

If you try to run another **glbd**, you will receive the following error: (GLB) cannot open log file. Nothing should happen.

I use ls_tv and it shows two servers. When I use ls_stat or ls_admin, it shows three. I only know of two and only two are working. How do I get rid of the third invalid entry?

The reason that **ls_tv** shows two servers is that it only reports on servers that are responding. **ls_stat** and **ls_admin** show all server entries good or bad. Run the following:

```
/etc/ncs/lb_admin
lb_admin: use global
lb_admin: clean
```

Answer **y** to all questions, and then quit the **lb_admin** by using the quit subcommand.

```
lb_admin: quit
```

How does my application get a license? Which NCS and iFOR/LS components actually get called?

1. Each license server registers itself and all installed vendors with the location broker when it starts up.
2. The user invokes an application, which in turn needs a license. The **netls_init** call prepares to request a license by finding all the possible servers that have licenses installed.
3. If a nodelocked entry for this vendor is in the **nodelock** file and the application allows a nodelocked license to be used, that license will be checked.
4. Otherwise, the **netls_init** call gets a list of iFOR/LS servers from the location brokers.

A broadcast message is sent out to a well-known port that all **llbds** are listening on. This message asks the question, "Any iFOR/LS servers out there with licenses that match my vendor ID?" Each **llbd** receiving that message will check if there is a **glbd** running on that machine and will forward the request to the **glbd**.

The **glbd** looks up the iFOR/LS servers and sends a reply to the application saying "Here are the places you can try to get a license for your application!"

The client receives this message and rearranges the list of servers into a random order. This causes the license requests to be spread among the servers so that the first server in the list doesn't get tried first every time.

5. The application requests a license from the server.

6. To determine whether the license request can be satisfied, the server checks the access rights of the user and the license database.
7. The server returns the status of the license request to the application.
8. If the status is success, the application is allowed to run.
9. If the status is unsuccessful and there are more servers to try, the application goes on and tries the next server (back to step 4).
10. If all the servers return unsuccessful status, the application is not allowed to run.

I'm installing iFOR/LS for the first time. I've tried everything, and it just won't work. I need to get this working ASAP.

1. Perform the following commands to ensure that **llbd**, **glbd**, **netlsd** are down:

```
lssrc -s llbd
lssrc -s glbd
lssrc -s netlsd
```

2. If the file **/tmp/llbbase.dat** exists, delete it.
3. Delete the files **glb.e**, **glb.p**, **glb_log** in **/etc/ncs**.
4. If you have a **glb_obj.txt** file delete it.
5. If you have a **glb_site.txt** file rename it.
6. Run **/usr/lib/netls/conf/netls_config**. If an existing database already exists, answer "no" to the question.
7. Select the default cell.
8. Run **netls_first_time**.
9. Check to make sure brokers are running as well as the server.
10. Run the following:

```
/etc/ncs/lb_admin
lb_admin: use global
lb_admin: clean
```

Everything should be running.

The glbd and llbd consistently use about 4% of the CPU, even though the daemons are idle for the most part. I have seen glbd and llbd as high as 3.9 % of the CPU on my system. Do you know why the daemons show 4% CPU usage?

Whenever you use the replicatable global broker, **glbd**, it runs with multiple threads within itself. One of these threads periodically wakes up to check other global brokers in the same cell in order to keep the distributed naming database consistent. As a result, it consumes CPU time.

If you are running a single global-broker cell and this CPU time is of concern, then you can use the alternate non-replicating global broker, **nrglbd**, which is not threaded and will not exhibit this behavior. However, as soon as you decide you need a replicated global name space in a cell, you have to go back to using the replicating global broker, **glbd**, and its associated multithreaded behavior to support the replicated naming database.

Although the figures indicate a noticeable consumption of CPU time, the perceived load on the servers running **glbds** is scarcely noticeable.

Problems Relating to Specific Configurations

I have created an alternate cell, but my client can't access the server. How can I get my client to access this server?

Since you created an alternate cell, the client can access this server but only this server. If this is what you set out to accomplish, you need to do the following:

- The client must also copy the **glb_obj.txt** file from the server to the **/etc/ncs** directory.
- The client may need to have a **glb_site.txt** file that points to that server in **/etc/ncs**. Most likely it will not need this file.
- Start your application, and the client should be able to get a token from that server.

I have set up two alternate cells, cell A on host1 and cell B on host2. Can users of host1 get licenses from cell B?

There is no mechanism in NCS to share services between cells. A host in any cell (default or an alternate) can only communicate with servers in the same cell. Cells were designed to isolate a group of servers and clients.

I'm on two different subnets, and I want to have three servers, each with 50 licenses. What is the best way to configure and set up my network?

1. You need to determine which subnet will be running two iFOR/LS servers and which subnet will be running one.
2. The subnet that is running two servers needs to have the following:
 - Both servers need to have the **llbd**, **glbd** and **netlsd** all running.
 - Both should be in the default cell.
3. The subnet that is running the third server:
 - If you already tried to bring up the servers and failed, make sure you take down the **llbd**, **glbd** and **netlsd** before continuing.
 - Before running **netls_config**, set up a **glb_site.txt** file that points to one of the default cells that is running on the other subnet. Run **netls_config**. If a database already exists, don't use it. You should see that **lb_find** actually broadcasts to the cell that was in the **glb_site.txt** file. Answer "2) Use the default for the system Cell Name". Run **netls_first_time**.
 - You have the option of keeping the **glb_site.txt** file or removing it. Once you have established that the system is actually running correctly, remove this file to avoid confusion in the future.

I have three servers each with 50 licenses. If one server goes down, do I still retain the total of 150 licenses?

No, if the server goes down, you only retain the 100 licenses on the servers that are currently up.

I have a configuration of six servers. Each server has 50 licenses, and each server is on a different subnet. I am going through several bridges

and routers, and `lb_find` does not see any additional `glbds` running. What do you recommend?

1. Install iFOR/LS in the default cell on a single workstation. Have `llbd`, `glbd`, and `netlsd` all running correctly. We will refer to this as server A.

Note: You must make sure no daemons are running before you do this next step.

2. On the second workstation B you can replicate the GLB database to increase its availability. Before running `netls_config`, create the `/etc/ncs/glb_site.txt` file. The file contains the IP address of server A.

3. Run `netls_config`.

Select the default cell. If there is a preexisting database, do not use it.

Note: For simplicity the `glb_site.txt` file needs only to contain the first server. If however you want to maintain a `glb_site.txt` file with all six servers, you would have to copy the file to each server machine before running `netls_config`. Also, `llbd`, `glbd`, and `netlsd` must be down at the time you run `netls_config`.

4. On all other servers, install iFOR/LS and run `netls_config`. If there is a preexisting database, do not use it. Select the default cell for the system cell name. If you still cannot see the replicated cell with `lb_find`, repeat the process above for the other four servers.

I want to have three servers each with 50 licenses, on the same subnet. What is the best way to configure and set up my network?

Each server should be running `llbd`, `glbd` and `netlsd`. Each `glbd` is using the default cell.

Running several global location broker daemons can help reduce server load and increase availability. It is also valuable when you have a complex network topology.

I created an alternate cell. No one else can communicate with this server. Why?

You have isolated this system from all other systems on the network. It now has its own cell. If you want your client machines to access this alternate cell, they must do the following:

- Create the `/etc/ncs` directory.
- Copy the `glb_obj.txt` file from the GLB host machine to the client's workstation.
- You may need to have a `glb_site.txt` file that points to the server. In most case you will not need this file.

Suppose you have two servers that each use the default cell, and you decide to create an additional alternate cell with another server. Each of the three servers has 10 licenses. You must now make a choice:

- The client machine can get licenses from the servers that are using the default cell for a total of 20 available licenses.
- The client can use the alternate cell and have access to only 10 licenses.

A client machine cannot have access to all 30 licenses.

Monitord Error Messages and Recovery Information

Following are the error messages you may see as a result of running **monitord**. Proceed through the steps until your problem is solved.

0349–010 Monitord is unable to create the lock file /etc/security/monitord_lock

EXPLANATION: The **monitord** lock file is created to let the **monitord** daemon determine if an instance of itself is already running.

ACTION:

1. Make sure the directory **/etc/security** is created.
2. Make sure the directory **/etc/security** has file permissions set for reading, writing, and running by root.

0349–011 Monitord is unable to fork the daemon process.

EXPLANATION: The **monitord** daemon must be created as a new process.

ACTION: Increase the number of processes allowed per user on the system.

0349–012 Monitord is unable to fork a child for daemon creation.

EXPLANATION: A child process is spawned as part of the daemon creation process.

ACTION: Increase the number of processes allowed per user on the system.

0349–013 Monitord is unable to change process groups.

EXPLANATION: **monitord** attempts to change the process group that it belongs to as part of its creation procedure.

ACTION: Contact your service representative.

0349–014 Monitord ran out of local storage for log in tracking.

EXPLANATION: The internal data structure that **monitord** uses to track user logins is full.

ACTION:

1. Stop **monitord** with a **kill -15** signal so the daemon will shut down cleanly.
2. Restart **monitord**.

0349–015 The monitord pipe could not be created.

EXPLANATION: **monitord** uses a special file called a “pipe” for communications. If the pipe can not be created, **monitord** will not be able to request and to release user login licenses.

ACTION:

1. Make sure the directory **/etc/security** is created.
2. Make sure the directory **/etc/security** has file permissions set for reading, writing, and executing by root.

0349–016 The monitord pipe could not be opened.

EXPLANATION: **monitord** uses a special file called a “pipe” for communications. If the pipe can not be opened, **monitord** will not be able to request and to release user login licenses.

ACTION:

1. Make sure the directory **/etc/security** is created.
2. Make sure the directory **/etc/security** has file permissions set for reading, writing, and executing by root.

0349–017 The monitord pipe could not be read.

EXPLANATION: **monitord** uses a special file called a “pipe” for communications. If the pipe cannot be read, **monitord** will not be able to request and to release user login licenses.

ACTION:

1. Make sure the directory **/etc/security** is created.
2. Make sure the directory **/etc/security** has file permissions set for reading, writing, and executing by root.

0349–018 The Action loop has been exited unexpectedly.

EXPLANATION: The Action loop is a loop that waits for input and executes an action based on the input that is received.

ACTION:

1. Check the file **/var/security/monitord_log** for errors and correct any that are found.
2. Restart the **monitord** daemon.

0349–019 Monitord could not write to the file /etc/security/monitord.cfg

EXPLANATION: The **monitord** daemon was not able to save configuration data to a file.

ACTION:

1. Make sure the directory **/etc/security** is created.
2. Make sure the directory **/etc/security** has file permissions set for reading, writing, and executing by root.

You must have root privileges to start monitord.

EXPLANATION: Only the root user has permission to run the **monitord** daemon.

ACTION:

1. Log in as the root user, or use the **su** command to become root.
2. Restart the **monitord** daemon.

usage: monitord [-t Minutes]

EXPLANATION: This is the error message that is returned when incorrect arguments are specified when starting **monitord**. The **-t** flag is optional. It is used to specify a time interval in minutes for letting the license server know that the licenses are still in use. If the **-t** flag is used, the value specified is stored in the file **/etc/security/monitord.cfg**. If no flag is specified, then the **/etc/security/monitord.cfg** file is checked to determine the time interval to use. The default interval is 15 minutes.

ACTION: Restart **monitord** with the correct arguments.

monitord: minutes argument must be a whole number.

EXPLANATION: The Minutes argument is specified incorrectly.

ACTION: Specify a whole number as the value supplied with the **-t** flag.

Monitord is running, or cleanup is required. : Remove /etc/security/monitord_lock before restarting.

EXPLANATION: The existence of the lock file indicates that the **monitord** daemon was started. It has either terminated unexpectedly, leaving the lock file; or it is still running.

ACTION:

1. If **monitord** is no longer running, but the lock file exists, it indicates that **monitord** terminated abnormally. If this is the case, then some licenses may still be held, pending timeout from the license server. These licenses are not recoverable until after the time-out period expires for the licenses. Remove the lock file before attempting to restart the **monitord** daemon.
2. If **monitord** is still running, it should only be stopped and restarted if the cell configuration for the machine has changed. Stop **monitord** with a **kill -15** signal to shut down **monitord** cleanly.

Monitord could not find any servers with valid AIX licenses installed.

EXPLANATION: **monitord** was unable to locate a license server that could satisfy license requests for user logins.

ACTION:

1. Verify that the license server has active licenses installed.
2. Verify that the machine running **monitord** can communicate with the machine running the license server. Check the cell configuration for both machines. They should be in the same cell.

A signal was caught by monitord. Monitord shutting down.

EXPLANATION: An interrupt signal was sent to the **monitord** daemon, causing it to exit.

ACTION: **monitord** can be restarted normally.

SERVER INITIALIZATION FAILED\n

EXPLANATION: **monitord** was unable to contact and initialize itself with a server.

ACTION:

1. Verify that the license server has active licenses installed.
2. Verify that the machine running **monitord** can communicate with the machine running the license server. Check the cell configuration for both machines. They should be in the same cell.

SERVER UNAVAILABLE FOR LICENSE CHECKOUT

EXPLANATION: A license could not be checked out because the server could not be contacted.

ACTION:

1. Restart the license server, if necessary.
2. Verify that the machine running **monitord** can communicate with the machine running the license server.

3. Check the cell configuration of the machines running **monitord** and the license servers. If the cell configuration was changed after **monitord** was started, either move the machines back to the original configuration, or stop and restart **monitord**.

NO LICENSE AVAILABLE

EXPLANATION: No licenses are available to be checked out from the server. All active licenses are in use.

ACTION:

1. Add more login licenses to the license servers to increase the number of licenses available for use.
2. Shut down and restart the **monitord** daemon.

MONITORD WAS UNABLE TO OBTAIN A USER LOG IN LICENSE

EXPLANATION: **monitord** could not obtain a license for a user login process. All active licenses are in use.

ACTION:

1. Add more login licenses to the license servers to increase the number of licenses available for use.
2. Shut down and restart the **monitord** daemon.

HEARTBEAT FAILED

EXPLANATION: The periodic check for the license server failed.

ACTION:

1. Verify that the machine running **monitord** can communicate with the machine running the license server. Check the cell configuration for both machines. They should be in the same cell.
2. Restart the license server if it has been stopped.

INVALID LICENSE RELEASE

EXPLANATION: A license was returned to the server, but it was not a valid license. This may occur after **monitord** fails to successfully communicate with the license server during one of its periodic checks (heartbeats).

ACTION: Avoid future communications delays and failures by making sure that communications is stable on the network, and by cleaning the location broker databases.

SERVER UNAVAILABLE FOR LICENSE RELEASE

EXPLANATION: The license server could not be reached when **monitord** attempted to return a user login license.

ACTION:

1. Check the cell configuration of the machines running **monitord** and the license servers. If the cell configuration was changed after **monitord** was started, either move the machines back to the original configuration, or stop and restart **monitord**.
2. Restart the license server if it has been stopped.

CLEANUP FAILED—LICENSES MAY STILL BE HELD

EXPLANATION: An error occurred when **monitord** attempted to release held licenses before shutting down.

ACTION: Check the file **/var/security/monitord_log** to determine why the clean up failed. If licenses are still held, they will be unusable until after the time-out period expires. If they are not held, they will be available for use immediately.

iFOR/LS (netlsd) Error Messages and Recovery Information

The error messages listed below use the following format:

Hex Value Long Value Short error message : Error Message

1d010001 486604801 netls_lic_not_fnd : License not found in database (network license server/server)

EXPLANATION: The product license requested by an application is not registered within the database.

ACTION:

1. Use the **ls_stat** command to determine whether the needed product license is installed on a reachable server.
2. Check the **nodelock** file and determine whether a license for the requested product is installed locally.

1d010002 486604802 netls_netls_lic_not_fnd : iFOR/LS license not found in database (network license server/server)

EXPLANATION: An internally used license for the iFOR/LS product is not registered within the database. The cause may be that the **/usr/lib/netls/conf/lic_db** database is not correctly initialized, or it is damaged.

ACTION:

1. Reinitialize the database by shutting down the **netlsd** daemon and removing the **lic_db**, **cur_db**, and **lic_db.bk** files in the **/usr/lib/netls/conf** directory.
2. Recover these database files from backup.
3. Restart the **netlsd** daemon.
4. Check with **ls_rpt -f** against fatal events

1d010003 486604803 netls_past_exp_date : Past expiration date on license (network license server/server)

EXPLANATION: The requested license has expired.

ACTION:

1. Check the expiration date shown in the license database and in the information that was delivered with the license.
2. Check the date and time of your machine. Make sure it is correct.
3. If your license has indeed expired, request a new license from the software vendor.

1d010004 486604804 netls_netls_past_exp_date : Past expiration date on iFOR/LS license (network license server/server)

EXPLANATION: The internal iFOR/LS product license has expired.

ACTION:

1. Check the date and time of your machine. Make sure it is correct.
2. Contact your service representative.

1d010005 486604805 netls_not_started : Before start date on license (network license server/server)

EXPLANATION: The start date of the requested license has not passed yet.

ACTION: Check the system time of your iFOR/LS server and compare it with the information delivered by the registration center for that product.

1d010006 486604806 netls_netls_not_started : Before start date on iFOR/LS license (network license server/server)

EXPLANATION: The start date of the internal iFOR/LS product license has not passed yet.

ACTION:

1. Check the date and time of your machine. Make sure it is correct.
2. Contact your service representative.

1d010007 486604807 netls_no_version : Version not found in database (network license server/server)

EXPLANATION: A license was found for the product, but it was for a different version of the product.

ACTION:

1. Use **ls_stat** to determine which version of the license is installed.
2. Check the application documentation to determine what version of the product is installed.
3. Contact your software vendor for a new license.

1d010008 486604808 netls_no_netls_version : iFOR/LS license version not found in database (network license server/server)

EXPLANATION: An internally used license for the iFOR/LS product is not registered within the database.

ACTION:

1. Reinitialize the database by shutting down the **netlsd** daemon and removing the **lic_db**, **cur_db**, and **lic_db.bk** files in the **/usr/lib/netls/conf** directory.
2. Recover these database files from backup.
3. Restart the **netlsd** daemon.
4. Contact your service representative.

1d010009 486604809 netls_not_enough_lics : Not enough licenses (network license server/server)

EXPLANATION: The product was unable to get as many licenses as it requested from the license servers.

ACTION:

1. Use the **ls_stat** command to check license usage.
2. Use the **ls_rpt** command to check the **log_file** for license requests, releases, and rejections. Perhaps an application died before releasing a license.
3. If an application died before releasing a license, you can free up licenses by stopping and restarting all license servers.
4. Request and install additional licenses for the product.

1d01000a 486604810 netls_not_enough_netls_lics : Not enough iFOR/LS licenses (network license server/server)

EXPLANATION: There are not enough installed licenses for an internally used license used by the iFOR/LS product.

ACTION:

1. Use the **ls_stat** command to check license usage.
2. Use the **ls_rpt** command to check the **log_file** for license requests, releases, and rejections. Perhaps an application died before releasing a license.
3. If an application died before releasing a license, you can free up licenses by stopping and restarting all license servers.
4. Request and install additional licenses for the product.

1d01000b 486604811 netls_not_authorized : Not authorized to use product (network license server/server)

EXPLANATION: The user is not authorized to use the requested product according to the user file **/user/lib/netls/conf/user_file**.

ACTION: Contact your system administrator to have the user allowed in the user file.

1d01000c 486604812 netls_wait_entry_deleted : Wait entry deleted (network license server/server)

EXPLANATION: The wait queue entry was deleted for the queued user. The user has been taken off of the license queue. This probably occurred because the application did not query the queue within two minutes after the previous query.

ACTION:

1. Restart your application.
2. Use the **lb_admin** and **drm_admin** commands to clean the location broker databases. This should improve communications performance and reduce the possibility that the queue queries are being delayed beyond the timeout.

1d01000d 486604813 netls_bad_io : Bad I/O (network license server/server)

EXPLANATION: The license server was unable to perform a read or write action.

ACTION:

1. Use the **ls_rpt** command to check the log file for error messages.
2. Stop **netlsd**.
3. Check available disk space on the license server machine.
4. Increase file system sizes for any full file systems.

5. Remove the files **log_file**, **lic_db**, **cur_db**, **lic_db.bak** from the **/usr/lib/netls/conf** directory.
6. Restore them from backup media.
7. Restart **netlsd**.

1d01000e 486604814 netls_others_waiting : Other users are waiting (network license server/server)

EXPLANATION: Other users are waiting in the wait queue.

ACTION:

1. Use the **ls_stat** command to check license usage and queue status.
2. Request and install additional licenses for the product.

1d01000f 486604815 netls_no_lics : No licenses available (network license server/server)

EXPLANATION: All licenses for the specific product are in use by other users.

ACTION:

1. Use the **ls_stat** command to check license usage.
2. Use the **ls_rpt** command to check the **log_file** for license requests, releases, and rejections. Perhaps an application died before releasing a license.
3. If an application died before releasing a license, you can free up licenses by stopping and restarting all license servers.
4. Request and install additional licenses for the product.

1d010010 486604816 netls_no_netls_lics : No iFOR/LS licenses available (network license server/server)

EXPLANATION: An internally used license for the iFOR/LS product is not registered within the database. The cause may be that the **/usr/lib/netls/conf/lic_db** database is not correctly initialized, or it is damaged.

ACTION:

1. Reinitialize the database by shutting down the **netlsd** daemon and removing the **lic_db**, **cur_db**, and **lic_db.bk** files in the **/usr/lib/netls/conf** directory.
2. Recover these database files from backup.
3. Restart the **netlsd** daemon.
4. If you still receive this error, contact your service representative.

1d010011 486604817 netls_still_waiting : Still waiting for a license (network license server/server)

EXPLANATION: Licenses are not available; user is still waiting on the license queue.

ACTION: None

1d010012 486604818 netls_non_matching_tid : No active license to be released (network license server/server)

EXPLANATION: The application released a license that was not valid at the time of release. The probable cause is that the application did not revalidate the license with the server before the time-out period expired for the license.

ACTION: Use the **lb_admin** and **drm_admin** commands to clean the location broker databases. This should improve communications performance and reduce the possibility that the license revalidations are being delayed beyond the timeout.

1d010013 486604819 netls_bad_entry : License is no longer granted to this user (network license server/server)

EXPLANATION: The license revalidation by the application tried to refresh a license which has already expired.

ACTION: Use the **lb_admin** and **drm_admin** commands to clean the location broker databases. This should improve communications performance and reduce the possibility that the license revalidations are being delayed beyond the timeout.

1d010014 486604820 netls_bad_timestamp : Time disparity too large (network license server/server)

EXPLANATION: The system date and time on the license server is too different from the system date and time on the machine where the application is running.

ACTION: Resynchronize the node clocks on the two machines to within two minutes of each other.

1d010015 486604821 netls_vnd_not_in_db : Vendor id not in database (network license server/server)

EXPLANATION: The software vendor ID was not found in the license database.

ACTION:

1. Check the database entries for that vendor using **ls_admin** and compare them with the information that was included with the license.
2. Use **ls_admin** to add the vendor information back into the license database.
3. Contact the software vendor to get a new license for the product.

1d010016 486604822 netls_duplicate_vendor : Duplicate vendor id (network license server/server)

EXPLANATION: The vendor ID used as input exists already in the license database.

ACTION: Normally, no action is required. The error simply means you have already added the vendor ID. You can simply proceed to adding the product ID. If you think you have not already added this vendor's ID, do the following:

1. Check your input data against the existing database entries.
2. Check your input data for the vendor against the data supplied with the license.
3. Check the vendor name used.

1d010017 486604823 netls_duplicate_product : Duplicate product record for this vendor (network license server/server)

EXPLANATION: The product license being added to the license server has already been added.

ACTION: Check your input data against the existing database entries.

1d010018 486604824 netls_no_such_product : No such product for this vendor (network license server/server)

EXPLANATION: The administrative action could not be performed because the specified product does not match any license server records for the specified vendor.

ACTION: Check the given input for product and vendor.

1d010019 486604825 netls_still_has_products : Could not delete vendor –still has products (network license server/server)

EXPLANATION: It is not possible to delete a vendor from the license database if product licenses for that vendor are still installed on the license server.

ACTION:

1. Delete all product licenses belonging to this vendor.
2. Delete the vendor.

1d01001a 486604826 netls_not_valid : Database not valid for 0this server (network license server/server)

EXPLANATION: The license database cannot be used. The database is either corrupted or applies to a different version of the license server product.

ACTION:

1. Stop the license server (**netlsd**).
2. Delete the files **lic_db**, **lic_db.bak**, and **cur_db** from the directory **/usr/lib/netls/conf**.
3. Restore the license database from a backup.
4. Restart the license server (**netlsd**).

1d01001b 486604827 netls_no_such_vendor : No such vendor (network license server/server)

EXPLANATION: it is not possible to add the specified product license for a vendor because no information for the vendor could be found in the license database.

ACTION:

1. Use **ls_admin** to verify whether the vendor has been added to the license database.
2. Check the given input for vendor and product.
3. Add the vendor ID to the license server before adding the product license password to the license server.

1d01001c 486604828 netls_database_corrupt : Database corrupt (network license server/server)

EXPLANATION: The license database (**/usr/lib/netls/conf/lic_db**) is corrupted and unusable.

ACTION:

1. Stop the **netlsd** daemon.
2. Remove the files **lic_db** and **cur_db** from **/usr/lib/netls/conf**.
3. Restore these files from backup media.
4. Restart the **netlsd** daemon.

5. Check with `ls_stat -i` to list all installed products.

1d01001d 486604829 netls_new_pname_unexpected : Warning: product name changed (network license server/server)

EXPLANATION: The product name in the database has been changed using `ls_admin`

ACTION: None

1d01001e 486604830 netls_cant_delete_netls_v : Vendor "Apollo" can not be deleted (network license server/server)

EXPLANATION: The license server may use an internal license from the vendor "Apollo".

ACTION: None

1d01001f 486604831 netls_fatal_error : Fatal error—check error log (network license server/server)

EXPLANATION: A fatal error occurred during license processing.

ACTION: Use the `ls_rpt -f` command to show all logged fatal events.

1d010020 486604832 netls_invalid_client : Vendor key is incorrect (network license server/server)

EXPLANATION: The application attempted to get a license from the license server, but was using a different key than the key used by the installed license.

ACTION: Contact your software vendor to obtain a new license that will work with the installed application.

1d010021 486604833 netls_server_not_found : iFOR/LS server not found (network license server/server)

EXPLANATION: No license servers could be found.

ACTION:

1. Verify that the license server (`netlsd`) is still running.
2. Check the cell configuration of the machines running the applications and license servers. The machines should be in the same cell.

1d010022 486604834 netls_old_server : Wrong version of iFOR/LS Server (network license server/server)

EXPLANATION: The current server cannot access the remote database of another server because the version differs. For example, you cannot change a license database on a Gradient iFOR server from an **iFOR/LS** server.

ACTION: Upgrade the servers to be at the same level.

1d010023 486604835 netls_log_corrupted : iFOR/LS log file contains invalid data (network license server/server)

EXPLANATION: The log file `/usr/lib/netls/conf/log_file` is corrupted and unusable.

ACTION:

1. Stop the license server (`netlsd`).
2. Remove the file `/usr/lib/netls/conf/log_file`.
3. Restart the license server.

1d010024 486604836 netls_too_early_to_derive : Start date of license is before compound derived start date (network license server/server)

EXPLANATION: The start date of the derived license occurs before the compound license start date.

ACTION: Contact the software vendor to obtain a compound license with an earlier start date.

1d010025 486604837 netls_too_late_to_derive : Expiration date of license is after compound derived expiration date (network license server/server)

EXPLANATION: The end date of the derived license occurs after the compound license start date plus the allowed duration

ACTION:

1. Use a valid end date for the derived license.
2. Contact the software vendor to obtain a compound license with a longer duration defined.

1d010026 486604838 netls_wrong_dest_type : License type contradicts license type specified in compound password (network license server/server)

EXPLANATION: The license type declared for the derived license must be the same as the license type declared in the compound license.

ACTION:

1. Check the information supplied with the compound license to determine the specified type for the derived license.
2. Contact the software vendor to obtain a compound license for the desired derived license type.

1d010027 486604839 netls_not_enough_dur : Aggregate duration is less than duration specified (network license server/server)

EXPLANATION: The specified duration for the license is less than the remaining duration allowed for derived licenses.

ACTION: Contact the software vendor to obtain a compound license with a longer derived duration.

1d010028 486604840 netls_not_expired : License has not expired – cannot delete (network license server/server)

EXPLANATION: The license cannot be deleted until after the license expiration date has passed.

ACTION: None.

1d010029 486604841 netls_bad_lic_annot : License annotation field is incorrect (network license server/server)

EXPLANATION: The information given within the annotation field does not conform to the allowed format for a license annotation.

ACTION: Check the input data against the information supplied for the license.

1d01002a 486604842 netls_attempted_renam : Attempting to install an existing product under a different name (network license server/server)

EXPLANATION: It is not possible to install an existing product under a different name.

ACTION: None

1d020001 486670337 netls_no_init : netlslib not initialized (network license server/library)

EXPLANATION: The product did not successfully initialize itself with the license server and attempted to continue license processing.

ACTION:

1. Verify communications between the client machine and the license server.
2. Contact your service representative.

1d020002 486670338 netls_no_queues : Not waiting in any queue (network license server/library)

EXPLANATION: The user attempted to exit from a queue that the user was not waiting on. This is probably because the user was removed from the queue when the queue position was not verified before the timeout expired.

ACTION:

1. Restart the application.
2. Use the **lb_admin** and **drm_admin** commands to clean the location broker databases. This should improve communications performance and reduce the possibility that the queue queries are being delayed beyond the timeout.

1d020003 486670339 netls_not_bound : Not bound to any server (network license server/library)

EXPLANATION: The product did not successfully initialize itself with the license server and attempted to continue license processing.

ACTION:

1. Verify communications between the client machine and the license server.
2. Contact your service representative.

1d020004 486670340 netls_invalid_svr : Vendor key is incorrect (network license server/library)

EXPLANATION: The application attempted to get a license from the license server, but was using a different key than the key used by the installed license.

ACTION: Contact your service representative to obtain a new license that will work with the installed application.

1d020005 486670341 netls_no_families : No matching socket families found (network license server/library)

EXPLANATION: A fatal error occurred in trying to find the socket family on the client host.

ACTION:

1. Verify that TCP/IP is installed and configured on both the client machine and the license server machine.
2. Verify that NCS is installed properly on both the client machine and the license server machine.

1d020006 486670342 netls_no_svrs_fnd : No servers available for this vendor (network license server/library)

EXPLANATION: No license servers were found that have licenses for the application.

ACTION:

1. Check the cell configuration of the machines running the applications and license servers. The machines should be in the same cell.
2. Use **ls_admin** to verify that the licenses needed are installed on a license server in the cell.
3. Install licenses for the application.

1d020007 486670343 netls_invalid_vid : Vendor id is invalid (network license server/library)

EXPLANATION: The vendor ID specified does not conform to the format of a standard UUID.

ACTION:

1. Verify that the vendor ID used as input matches the vendor ID that was supplied with the license.
2. Contact your software vendor to get a new license with a correct vendor ID.

1d020008 486670344 netls_bad_param : Bad parameter (network license server/library)

EXPLANATION: One of the parameters to this call was not correct.

ACTION:

1. Verify that the parameters used in the command are specified correctly.
2. The order that the parameters are specified may be significant if the command line is used. See “netlsd Command” on page A-37 for more information on command usage.

1d020009 486670345 netls_invalid_job_id : Job id is invalid (network license server/library)

EXPLANATION: The job ID used by the application to verify itself to the license server is not valid.

ACTION:

1. Restart the application.
2. Call your software vendor.

1d02000a 486670346 netls_duplicate_vnd : Duplicate vendor id (network license server/library)

EXPLANATION: The vendor ID used as input exists already in the **netlsd** database.

ACTION: Normally, no action is required. The error simply means you have already added the vendor ID. You can simply proceed to adding the product ID. If you think you may not have added this vendor ID, do the following:

1. Check your input data against the existing database entries.
2. Check your input data for the vendor against the data supplied with the license.

3. Check the vendor name used.

1d02000b 486670347 netls_cant_create_nl : Cannot create nodelock file (network license server/library)

EXPLANATION: The file `/usr/lib/netls/conf/nodelock` could not be created.

ACTION: Make sure the directory `/usr/lib/netls/conf` has write permission set for the root user.

1d02000c 486670348 netls_no_server_handle : No valid server handle exists, cannot log message (network license server/library)

EXPLANATION: The license server is unable to identify a server name to use when logging messages.

ACTION:

1. Verify that TCP/IP and NCS are properly installed and configured on the client and license server machines.
2. Stop and restart the license server (`netlsd`).

1d030001 486735873 netls_decode_bad_version : Product version incorrect (network license server/tools)

EXPLANATION: The vendor information supplied does not match the information encrypted in the license password.

ACTION:

1. Check the input data specified against the information supplied with the license.
2. Contact your software vendor to obtain a new license with correct vendor information.

1d030002 486735874 netls_bad_password : Password contains invalid characters (network license server/tools)

EXPLANATION: The given password contains invalid characters.

ACTION:

1. Check your input value for the password against the value supplied with the license.
2. Contact your software vendor to obtain a new license and password.

1d030003 486735875 netls_wrong_target : Incorrect target in password (network license server/tools)

EXPLANATION: The user is attempting to install a license on a machine, but the license was created for a different machine.

ACTION:

1. Use the `ls_targetid` command to check the target ID of the machine on which the license is being installed. The target ID of the machine must be the same as the target ID specified in the information supplied with the license.
2. Contact your software vendor to reorder the license for the correct machine.

1d030004 486735876 netls_bad_pword_ver : Invalid password or vendor incorrect (network license server/tools)

EXPLANATION: The vendor information supplied does not match the information encrypted in the license password.

ACTION:

1. Check the input data specified against the information supplied with the license.
2. Contact your software vendor to obtain a new license with correct vendor information.

1d030005 486735877 netls_fatal_err : Internal error (network license server/tools)

EXPLANATION: A unexpected problem occurred with the license server.

ACTION:

1. Use **ls_rpt -f** to show all logged fatal events.
2. Stop and restart the license server (**netlsd**).
3. Contact your service representative.

1d030006 486735878 netls_not_bundled : 1.0 or 1.1 license not bundled (network license server/tools)

EXPLANATION: The license server was not installed completely.

ACTION: Deinstall and reinstall the license server software.

1d030007 486735879 netls_cant_add_nl : Nodelocked license cannot be added to server database (network license server/tools)

EXPLANATION: It is impossible to add licenses of the type "nodelock" to the license database.

ACTION: You add nodelocked licenses to the **nodelock** file (**/usr/lib/netls/conf/nodelock**), not to the license server.

Chapter 5. Network Computing System (NCS)

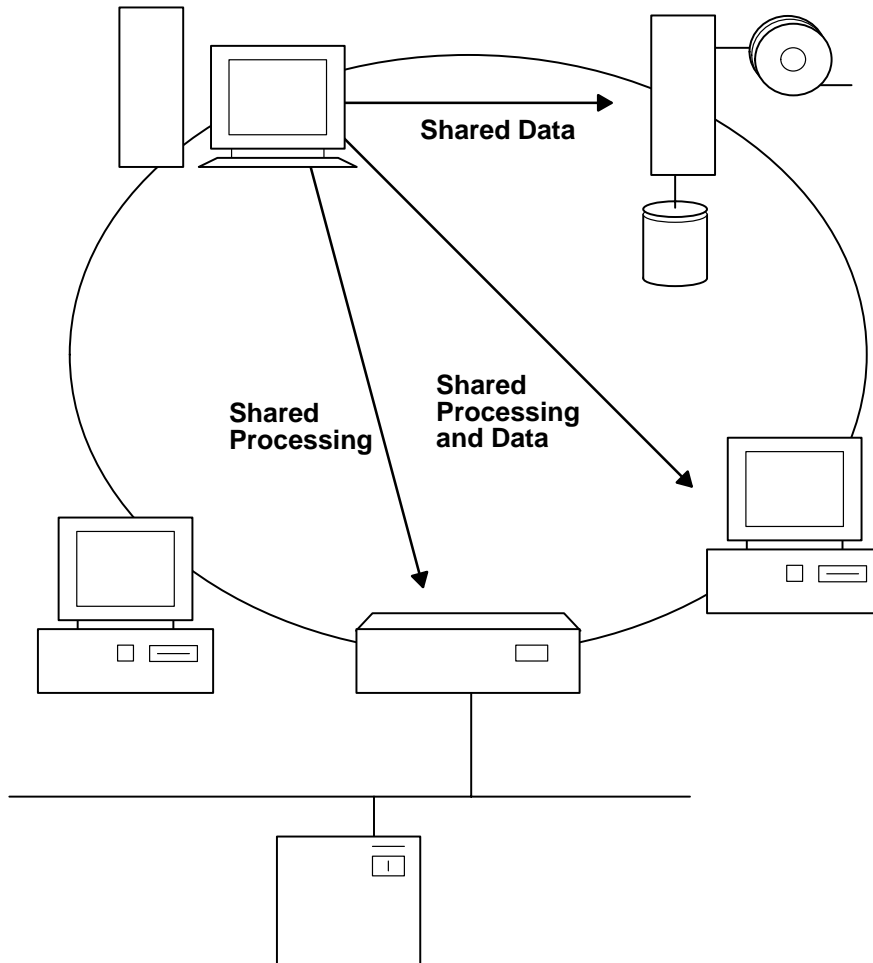
This chapter describes the Network Computing System, which enables distribution of computing tasks across an internet.

Network Computing System (NCS) Introduction

The Network Computing System (NCS), an implementation of the Network Computing Architecture, is a set of tools for distributing computer processing tasks across resources in either a network or several interconnected networks (an internet). These networks and internets can include a variety of computers and programming environments. Programs based on the Network Computing Architecture take advantage of computing resources throughout a network or internet, with different parts of each program executing on the computers best suited for those specific tasks. NCS attempts to provide networked computing services while making the computer network transparent to end users.

The figure "Network Computing Architecture" shows how the Network Computing Architecture shares processing and application data.

To enable application programs to access network computing resources, NCS provides the remote procedure call (RPC) run-time library and location brokers. Together these two components make up the Network Computing System (NCS), which contains all the software required to run a distributed application.



NCS Components

NCS consists of the following components.

Remote Procedure Call (RPC) Run-Time Library

The RPC run-time library provides library routines that enable local programs to run procedures on remote hosts. These routines transfer requests and responses between clients (the programs calling the procedures) and servers (the programs executing the procedures).

Location Brokers

A *location broker* provides information about the network or internet resources to clients. It maintains a database containing the identities and locations of objects in the network. A

local location broker (LLB) manages information about resources on an individual host. A *global location broker* (GLB) manages information about resources available on all hosts.

Configuring the `glb_site.txt` File

Each NCS client on the network must have an `/etc/ncs/glb_site.txt` file that specifies the daemon to be accessed by that client. The `glb_site.txt` file consists of one or more lines of the format:

```
ip:HostName  
ip:#ipaddress
```

where each line indicates a host to check for an active global location broker daemon. The entries in the file are processed sequentially until an active daemon is found or the end of the file is reached. To configure the `glb_site.txt` file, do the following:

1. Open the `glb_site.txt` file with your favorite editor.
2. Add a line (in the format described above) for each NCS server on the network where a `glbd` daemon is running.
3. Save and close the file.

Manually Configuring the Network Computing System

This section describes the procedure for manually performing the steps normally performed through `netls_config` and `netls_first_time`.

Prerequisites

Please refer to “Prerequisites” on page 2-4 for a full list of prerequisites.

Procedure

The first step is to determine if in fact NCS is already running on your network, in which case you may not have to do anything at all.

1. Log in to a node on the network as root.
2. Invoke the `lb_find` command (located in `/usr/lib/ncs/bin`).
3. If the NCS is installed and running on the network, the display should be similar to that shown below:

```
lb_find  
sent to broadcast address 192.92.110.255  
waiting for replies  
received response from glb daemon at ip:daffy(192.92.110.4)port  
1243  
received response from glb daemon at ip:stimp(192.92.110.43)port  
1110  
received response from glb daemon at ip:apollo(192.92.110.12)port  
1024  
...  
replicable ip:daffy alternate_1  
54bdad9a4000.0d.00.01.83.0f.00.00.00
```

```
replicatable ip:stimpay alternate_1
54bdad9a4000.0d.00.01.83.0f.00.00.00
```

```
replicatable ip:apollo alternate_1
54bdad9a4000.0d.00.01.83.0f.00.00.00
```

The example output shown above indicates that there are three global location brokers running, that they all belong to the same cell, and that the cell is not the default cell.

4. If the display looks similar to the one shown below, it means that the NCS is not installed and running on the network:

```
lb_find
sent to broadcast address 192.92.110.255
waiting for replies
received response from glb daemon at ip:daffy(192.92.110.4)port
1243
received response from glb daemon at ip:stimpay(192.92.110.43)port
1110
received response from glb daemon at ip:apollo(192.92.110.12)port
1024
.....
```

For more information on the **lb_find** command, see the *AIX Version 4.1 Commands Reference*.

Having determined the status of the NCS, you have three options:

- If there is no NCS running on your network and you intend to use the default GLB cell, then follow the procedure on page 5-4 for installing the first global location broker daemon.
- If you intend to establish a new GLB cell, then follow the procedure on page 5-5 on establishing a new NCS cell.
- If NCS is running on your network and you want to create an additional **glbd** then follow the procedure on page 5-7 on installing additional global location broker daemons.

Installing a Global Location Broker Daemon in the Default Cell

To install a global location broker daemon in the default cell:

Note: Before you install a global location broker daemon in the default cell, be sure there is not one there already.

1. Choose the node to run the global location broker daemon, **glbd**.
2. Log in to that node as root.
3. Use this syntax to start the **llbd**, running in the background:

```
cd /usr/lib/ncs/bin
startsrc -s llbd
```

4. Initialize the first global location broker daemon in the NCS cell (or in the default cell that you get if you do not explicitly create your own cell) with this syntax:

```
startsrc -s glbd -a "-create -first -family ip"
```

This command creates the global location database, tells the global location broker daemon that it is the first one in the network, and tells it to use the TCP/IP network protocol.

5. Make sure that the global location broker daemon is running with the **lssrc** command.
6. Verify that the global location broker daemon is communicating properly by starting the **drm_admin** utility:

```
drm_admin
```

The **drm_admin** facility will prompt you for further actions. With the **set** command, you set the object you want to administer to **glb**, as well as the host to the system you are working on:

```
drm_admin: set -o glb -host ip:HostName
```

drm_admin reports the status of the global location broker on the host. Its state should be *in service*. Use the quit subcommand to quit **drm_admin**.

7. Now make sure that **llbd** and **glbd** are correctly started when you reboot the system. To do this, edit the file **/etc/rc.ncs**. Find the lines that are commented out that start the **llbd** and **glbd** programs, and delete the comment characters from the beginning of the executable lines. The final result should look like this:

```
<preceding lines>
#Uncomment the following line to start the Local Location Broker
startsrc -s llbd
#Uncomment the following line to start the Global Location Broker
startsrc -s glbd
<more lines>
```

8. Reboot the system and verify that **llbd** and **glbd** were started automatically.
9. Try **drm_admin** again to make sure that the **glbd** is in service.
10. If you plan to install the iFOR/LS license server daemon on this system, see "Setting Up the iFOR/LS License Server" on page 5-8.

Establishing a New NCS Cell

It is important to understand the impact of joining or creating a cell when configuring your iFOR/LS license server. Remember that a cell is a logical partitioning of servers and clients on the network. Key points to keep in mind about cells are:

- A license request for a licensed application can be granted only from either a nodelock file or a license server in the same cell.
- A license server can only talk to licensed applications and other servers that belong to the same cell.
- Cell configuration is per machine. A machine and the applications and servers that run on it can not be in more than one cell at the same time.

Normally you would want to run your server only in the default cell. This is the simplest configuration and allows the general community to use the installed licenses without any special configuration. You should work with your system administrator when starting your server in the default cell since you will be impacting other systems and users. If your machine is not on a network, but you still need to run the license server for an installed application, it also makes sense to run in the default cell.

You should configure your license server in an alternate cell if your environment is unstable. For instance, you may be testing hardware or software, and your machine or network may not be stable enough for you to always perform a clean shutdown of your license server

before it goes offline. In this case, you would want to protect the stability of the default cell by running your server and applications in an alternate cell.

You may also want to run your system in an alternate cell if the default cell is known to be unstable. This may occur when many inexperienced users are all starting and stopping their servers in the default cell without properly cleaning up after themselves. By running your system in an alternate cell, you isolate it from unpredictable changes to the default cell.

1. Choose the node to run the global location broker daemon, **glbd**.
2. Log in to that node as root.
3. Generate a universal unique identifier (UUID) for the cell and put this UUID into a special file called **glb_obj.txt** in the **/etc/ncs** directory. UUIDs are generated with the **uuid_gen** program supplied. Use the following syntax:

```
cd /usr/lib/ncs/bin
uuid_gen >/etc/ncs/glb_obj.txt
```

The content of the **glb_obj.txt** file created with **uuid_gen** is a string somewhat like the following:

```
54c7874546ae.02.81.87.92.34.00.00.00
```

Since the string is generated from the unique system ID of the workstation and from a time stamp, the number is uniquely different from the number that anyone else would generate with **uuid_gen**, and different from the number that would be generated with **uuid_gen** at a different time. The **glb_obj.txt** file uniquely identifies an NCS cell.

4. Copy the **glb_obj.txt** file that was just generated into the **/etc/ncs** directory of every node that is part of the cell. This includes both iFOR/LS license servers and nodes that have the licensed product installed but are not set up to be iFOR/LS license servers.
5. Use this syntax to start the **llbd**, running in the background:

```
cd /usr/lib/ncs/bin
startsrc -s llbd
```

6. Initialize the first global location broker daemon in the NCS cell (or in the default cell that you get if you do not explicitly create your own cell) with this syntax:

```
startsrc -s glbd -a "-create -first -family ip"
```

This command creates the global location broker database, tells the global location broker daemon that it is the first one in the network, and tells it to use the TCP/IP network protocol.

7. Make sure that the global location broker daemon is running with the **lssrc** command:

```
lssrc -s glbd
```

8. Verify that the global location broker daemon is communicating properly by starting the **drm_admin** utility:

```
drm_admin
```

The **drm_admin** facility will prompt you for further actions. With the **set** command, you set the object you want to administer to **glb**, as well as the host to the system you are working on:

```
drm_admin: set -o glb -h ip:HostName
```

drm_admin reports the status of the global location broker on the host. Its state should be *in service*.

- Now make sure that **llbd** and **glbd** are correctly started when rebooting the system. To do this, edit the file **/etc/rc.ncs**. Find the lines that are commented out that start the **llbd** and **glbd** programs, and delete the comment characters from the beginning of the executable lines. The final result should look like this:

```
<preceding lines>
#Uncomment the following line to start the Local Location Broker
startsrc -s llbd
#Uncomment the following line to start the Global Location Broker
startsrc -s glbd
<more lines>
```

- Reboot the system and verify that **llbd** and **glbd** were started automatically.
- Try **drm_admin** again to make sure that the **glbd** is in service.
- If you plan to install the iFOR/LS license server daemon on this system, you may want to do that before continuing. See "Setting Up the iFOR/LS License Server" on page 5-8.

Installing a Global Location Broker Daemon in the Alternate Cell

If you decide to run more than one global location broker daemon on the network, use the following procedure to install each additional **glbd**:

- If you set up a new NCS cell, copy the **/etc/ncs/glb_obj.txt** file from the first **glbd** node to the next one rather than generating a new file.
- Use the **date** command to find the date on the node:
- Use the **drm_admin** command to find the date on nodes where other **glbd** servers are running:

```
drm_admin
drm_admin: set -o glb -h ip:OtherglbHost
```

drm_admin will show the dates on other global location broker servers.

- If the dates differ by more than two minutes, restore the dates of all **glb** servers to within 2 minutes of each other using the **date** command on systems whose dates must be changed.
- If the local location broker daemon, **llbd**, is not running, start it running in the background:

```
startsrc -s llbd
```

- Initialize the global location broker daemon with the command:

```
startsrc -s glbd -a "-create -from ip:OtherglbHost"
```

This command creates the global location database and creates a replica of the contents of the database found on node *OtherglbHost*.

- Make sure that **glbd** is running with the **lssrc** command.

```
lssrc -s glbd
```

- Verify that **glbd** is communicating properly by starting the **drm_admin** utility:

```
drm_admin
```

The **drm_admin** facility will prompt for further actions. With this command, set the object you want to administer to **glb**, as well as the host to system you are working on:

```
drm_admin: set -o glb -host ip:HostName
```

drm_admin will tell you the status of the global location broker on the host. Its state should be *in service*.

9. Now make sure that **llbd** and **glbd** are correctly started when rebooting the system. To do this, edit the file **/etc/rc.ncs**. Find the lines that are commented out that start the **llbd** and **glbd** programs, and delete the comment characters from the beginning of the executable lines. The final result should look like this:

```
<preceding lines>
#Uncomment the following line to start the Local Location Broker
startsrc -s llbd
#Uncomment the following line to start the Global Location Broker
startsrc -s glbd
<more lines>
```

10. Reboot the system and verify that **llbd** and **glbd** were started automatically.

Setting up the iFOR/LS License Server

After setting up one or more global location brokers on the network, one or more iFOR/LS license server daemons can be enabled as follows:

1. Log in as root.
2. Determine if the local and global location broker daemons, **llbd** and **glbd**, are running on the system using the **lssrc** command:

```
lssrc -s llbd
lssrc -s glbd
```

3. If the brokers are running, go to the next step. If not, refer to “Installing the First Global Location Broker Daemon” on page 5-4 for information on getting them started.
4. Now start the iFOR/LS daemon, **netlsd**:

```
startsrc -s netlsd
```

5. Make sure that **llbd** will be started when rebooting the system. If you have followed the directions to set up the node to be a global location broker, **llbd** is already enabled on bootup, so you can skip to step 7.

Otherwise, edit the file **/etc/rc.ncs**. Find the lines that are commented out that start **llbd** and delete the comment characters at the beginning of the executable lines:

```
<preceding lines>
#Uncomment the following line to start the Local Location Broker
startsrc -s llbd
#Uncomment the following line to start the Global Location Broker
startsrc -s glbd
<more lines>
```

6. Make sure that **netlsd** will be started when the system is rebooted. To do so, you need to make changes to **/etc/rc.netls**. Find the commented-out lines that start **netlsd**, and delete the comment characters at the beginning of the executable lines.

An example follows of an entry that starts the license server without logging checkin, vendor, product, timeout, or message events.

```
#####
#
# rc.netls -- iFOR/LS startup file
#
# Note: This system must run a llbd in order to run a netlsd
#
#
# See also: /etc/rc.ncs, llbd(1M), glbd(1M).
#####
```

```
#####
# Start the iFOR/LS daemon
#####
if /usr/lib/ncs/bin/lb_find -q
then
echo "Starting the netlsd daemon"
startsrc -s netlsd -a "-no cvptm" && echo "starting up the
iFOR/LS Daemon"
else
echo "The GLB server is not responding;"
echo "iFOR/LS server cannot be started."
fi
```

7. Reboot the computer to make sure that the appropriate daemons are started. If they are not, check the contents of the **/etc/rc.ncs** and **rc.netls** files carefully.

Appendix A: iFOR/LS Commands

Descriptions of license-server commands and tools for iFOR/LS system administration follow; the descriptions appear in alphabetical order.

drm_admin command

glbd daemon

lb_admin command

lb_find command

llbd daemon

ls_admin command

ls_dpass command

ls_rpt command

ls_stat command

ls_tv command

monitord daemon

netlsd daemon

nrglbd daemon

The license-server tools reside in the **/usr/lib/netls/bin** directory.

drm_admin Command

Purpose

Administers servers based on the Data Replication Manager (DRM), such as **glbd**, the replicated version of the global location broker (GLB).

Syntax

```
— drm_admin — [ — -version — ]
```

Description

The **drm_admin** tool administers servers based on the Data Replication Manager (DRM) such as **glbd**, the replicated version of the global location broker (GLB).

With **drm_admin**, you can inspect or modify replica lists, merge databases to force convergence among replicas, stop servers, and delete replicas.

The role of **drm_admin** is to administer the replication of databases, not to change the data they contain. For instance, you can use **drm_admin** to merge two replicas of the GLB database, but you must use **lb_admin** to add a new entry to the database. Also, although **drm_admin** can stop or delete a GLB replica, you must invoke **glbd** directly if you want to start or create a replica.

Once invoked, **drm_admin** enters an interactive mode, in which it accepts the commands described below.

Flags

-version Displays the version of NCS that this **glbd** belongs to, but does not start the daemon.

Subcommands

Most **drm_admin** commands operate on a default object (*DefaultObj*) at a default host (*DefaultHost*). Together, *DefaultObj* and *DefaultHost* 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. We identify this host as *OtherHost*.

The host name you supply as a *DefaultHost* or an *OtherHost* takes the form *Family:Host*, where the host can be specified either by its name or by its network address. For example, *ip:jeeves*, *ip:bertie*, and *ip:#192.5.5.5* are acceptable host names.

addrep *OtherHost*

Adds *OtherHost* to the replica list at *DefaultHost*. The replica at *DefaultHost* will propagate *OtherHost* to all other replica lists for *DefaultObj*.

chrep **-from** *OtherHost* **-to** *NewOtherHost*

Changes the network address for *OtherHost* in the replica list at *DefaultHost* to *NewOtherHost*. The replica at *DefaultHost* will propagate

this change to all other replica lists for *DefaultObj*. The **chrep** command will fail if a replica of *DefaultObj* is running at *OtherHost* or if *OtherHost* is not on the replica list at *DefaultHost*.

delrep *OtherHost*

Deletes the replica of *DefaultObj* at *OtherHost*. The **delrep** command tells the replica at *OtherHost* to:

1. Propagate all of the entries in its propagation queue.
2. Propagate a delete request to all other replicas, causing *OtherHost* to be deleted from all other replica lists for *DefaultObj*.
3. Delete its copy of *DefaultObj*.
4. Stop running.

The **delrep** command returns you immediately to the **drm_admin** prompt, but the actual deletion of the replica can take a long time in configurations that are not stable and intact. You can check whether the daemon for the deleted replica has stopped by listing the processes running on its host.

info Gets status information about the replica for *DefaultObj* at *DefaultHost*.

lrep [-d] [-clocks] [-na]

Lists replicas for *DefaultObj* as stored in the replica list at *DefaultHost*.

- d** Lists deleted as well as existing replicas.
- clocks** Shows the current time on each host and indicates clock skew among the replicas.
- na** Lists the network address of each host.

merge {-from | -to} *OtherHost*

Copies entries in the *DefaultObj* 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 timestamp.

A merge does not cause entries to be propagated. The database and replica list at the origination are not changed.

The **-from** option copies entries from the *DefaultObj* database and replica list at *OtherHost* to the *DefaultObj* database and replica list at *DefaultHost*.

The **-to** option copies entries from the database and replica list at *DefaultHost* to the database and replica list at *OtherHost*.

A **merge -from** followed by a **merge -to** causes the replicas at the two hosts to converge.

merge_all

Uses *DefaultHost* as the hub for a global merge of all replicas for *DefaultObj*. For each host on the replica list at *DefaultHost*, a **merge_all** first does a **merge -from**, then does a **merge -to**. All replicas of *DefaultObj* are thereby forced into a consistent state. The **merge_all** operation does not cause any entries to be propagated.

You should do a **merge_all** when:

- A replica is purged.
- A replica is reset.
- A replica has been inaccessible for two weeks or more.

drm_admin

A replica has become physically inaccessible (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 *DefaultObj* every n minutes and to report any clock skews or nonanswering replicas. If you do not specify **-r**, the period is 15 minutes.

purgerep *OtherHost*

Purges *OtherHost* from the replica list at *DefaultHost*. The replica at *DefaultHost* then propagates a delete request to the replicas at the hosts remaining on its list, thereby removing *OtherHost* from all other replica lists for *DefaultObj*. The delete request is not sent to *OtherHost*.

A **purgerep** can cause data to be lost and should only be used when a replica has become physically inaccessible. You should do a **merge_all** operation after the **purgerep** to prevent the remaining replicas of the *DefaultObj* database from becoming inconsistent. If the purged replica is still running, it should be reset.

We recommend that you use **chrep** (rather than **addrep** and **purgerep**) to change entries on the replica list.

quit Quits the **drm_admin** session.

reset *OtherHost*

Resets the replica of *DefaultObj* at *OtherHost*.

The **reset** command tells the replica at *OtherHost* to delete its copy of *DefaultObj* and to stop running. It does not cause *OtherHost* to be deleted from any other replica lists. This command can cause data to be lost unless a successful **merge_all** is done first.

set [-o *ObjName*] -h *HostName*

Sets the default object and host. All subsequent commands will operate on *ObjName*. Subsequent commands that do not specify a host will be sent to *HostName*. If you do not specify the **-o** option, **drm_admin** keeps the current *DefaultObj*.

If you use **set** with the **-o** option, **drm_admin** checks the clocks at all hosts with replicas of the specified object.

stop Stops the server for *DefaultObj* that is running at *DefaultHost*.

Examples

The following example starts **drm_admin**, sets the default object to GLB, and sets the default host to mars:

```
/etc/ncs/drm_admin drm_admin: set -o glb -h dds:mars
Default object: glb default host: dds:mars
state: in service
Checking clocks of glb replicas
dds:mars 1987/04/09.17:09
dds:pluto 1987/04/09.17:09
dds:mercury 1987/04/09.17:07
```

Implementation Specifics

This command is part of the Network Computing System filesset.

Related Information

The **lb_admin** command.

The **glbd** (NCS) daemon

TCP/IP must be configured and running on your system before starting the **glbd** daemon. The **llbd** daemon must also be started and running before you start the **glbd** daemon.

Flags

- create** Creates a replica of the GLB. This option creates a GLB database in addition to starting a broker process. It must be used with either **-first** or **-from**.
- first** Creates the first replica (that is, the very first instance) of the GLB on your network or internet. This option can be used only with the **-create** option.
- family *FamilyName***
Specifies the address family that the first GLB replica will use to identify itself on the replica list. This option can be used only in conjunction with the **-first** option. Any subsequently created replicas must use this family to communicate with this replica. Currently, *FamilyName* can be either **dds** or **ip**. If this option is not used, the replica will be identified on the replica list by its DDS address.
- from *HostName***
Creates additional replicas of the GLB. This option can be used only with the **-create** option. A replica of the GLB must exist at *HostName*. The database and replica list for the new replica are initialized from those at *HostName*. The replica at *HostName* adds an entry for the new replica to its replica list and propagates the entry to the other GLB replicas.
- A *HostName* takes the form family:host, where the host can be specified either by its name or by its network address. For example, **ip:jeeves**, **ip:bertie**, and **ip:#192.5.5.5** are acceptable host names.
- The new replica will use the same address family as *HostName* in identifying itself on the replica list. For example, if *HostName* is an IP address, the new replica will be listed by its IP address on the replica list.
- change_family *FamilyName***
Changes the address family of every GLB replica. Use this option only if network reconfigurations require that you make such a change. Currently, *FamilyName* can be either **dds** or **ip**.
- listen *FamilyList***
Restricts the address families on which a GLB listens. Use it only if you are creating a special configuration where access to a GLB is restricted to a subset of hosts in the network or internet.
- The *FamilyList* is a list of the address families on which the GLB will listen. Names in this list are separated by spaces. Possible family names include **dds** and **ip**.
- The GLB will always listen for requests from the family by which it is listed on the replica list, even if that family is not specified in *FamilyList*.
- If **glbd** is started without the **-listen** option, the GLB will listen on all address families that are supported both by NCS and by the local host. On

glbd (NCS)

Apollo systems, this set of families always includes **dds** and may also include **ip**. On most other systems, **ip** is currently the only family.

-version Displays the version of NCS that this **glbd** belongs to, but does not start the daemon.

Files

/etc/ncs/glb_log Contains diagnostic output from **glbd**.
/etc/rc.ncs Contains commands to start the NCS daemons.

Examples

1. Create and start for the first time the first replica of the GLB on this network or internet:

```
/etc/ncs/glbd -create -first -family ip &
```
2. Start for the first time a subsequent replica of the GLB, initializing its database from host jeeves:

```
/etc/ncs/glbd -create -from ip:jeeves &
```
3. Restart an existing replica of the GLB:

```
/etc/ncs/glbd &
```

Implementation Specifics

This command is part of the Network Computing System fileset.

Related Information

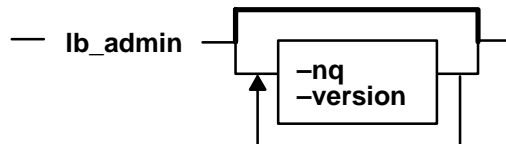
The **drm_admin** command, **lb_admin** command, **startsrc** command.
The **llbd** daemon.

lb_admin Command

Purpose

Administers the registration of NCS-based servers in location broker databases.

Syntax



Description

The **lb_admin** tool 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. The **lb_admin** tool can be used to look up information, add new entries, and delete existing entries in a specified database.

The **lb_admin** 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 **lb_admin** to manually remove its entry from the GLB database.

When accepting input or displaying output, **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
```

where each x is a hexadecimal digit. Descriptive textual names are associated with UUIDs in the **uuidname.txt** file.

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.

Flags

-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.

Subcommands

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**.

lb_admin

c[lean] Finds and deletes 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** tries to look 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.

There are two situations in which it is likely that a database entry should be deleted:

- The server does not respond. **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.
- Server responds, but its UUIDs do not match the entry in the database. The server that responded 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, there are four ways to respond. 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[ele]te Synonym for unregister.

h[elp] [*Command*] or ? [*Command*]
Displays a description of the specified *Command* or, if none is specified, list all of the **lb_admin** commands.

l[ookup] *Object Type Interface*
Looks up and displays all entries with matching *Object*, *Type*, and *Interface* fields in the current database. An asterisk can be used as a wildcard for any of the arguments. If all the arguments are wildcards, **lookup** displays the entire database.

q[uit] Exits the **lb_admin** session.

r[egister] *Object Type Interface Location Annotation [Flag]*
Adds the specified entry to the current database. Use an asterisk to represent the nil 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. Possible values for *Family* include **ip**. A leading **#** can be used to indicate that a host name is in the standard numeric form. For example, `ip:vienna[1756]` and `ip:#192.5.5.5[1791]` are acceptable location specifiers.

The *Annotation* is a string of up to 64 characters annotating the entry. Use double quotation marks to delimit 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] [*BrokerSwitch*] *Host*

Sets the host for the current LLB or GLB. If specifying global as the *BrokerSwitch*, **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. Possible values for *Family* include **ip**. A leading **#** can be used to indicate that a host name is in the standard numeric form. For example, `ip:prague` and `ip:#192.5.5.5` are acceptable host specifiers.

Issue **use_broker**, not this command, to determine if subsequent operations will access the LLB or the GLB.

set_t[imeout] [**short** | **long**]

Sets the timeout period used by **lb_admin** 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 timeout value.

u[nregister] *Object Type Interface Location*

Deletes 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. Possible values for *Family* include **ip**. A leading **#** can be used to indicate that a host name is in the standard numeric form. For example, `ip:vienna[1756]` and `ip:#192.5.5.5[1791]` are acceptable location specifiers.

An asterisk can be used 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] [*BrokerSwitch*]

Selects the type of database that subsequent operations will access, GLB or LLB. The *BrokerSwitch* is either global or local. If a *BrokerSwitch* 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.

Implementation Specifics

This command is part of the Network Computing System filesset.

Related Information

The **drm_admin** (NCS) command

The **glbd** (NCS) daemon, **llbd** (NCS) daemon, **nrglbd** (NCS) daemon.

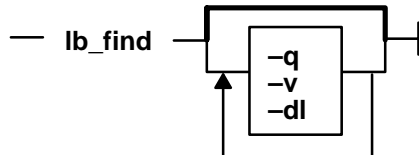
lb_find

lb_find

Purpose

Gets a list of global location broker (GLB) server daemons and their attributes.

Syntax



Description

lb_find sends out inquiries to the NCS location broker daemons and gathers the responses. The results are analyzed to determine whether the global location broker is replicatable, and which cell each daemon serves. After ten seconds, the results are summarized, showing the GLB broker type, the server host's network address, a cell name of either *default* or *alternate_N*, and the cell's UUID.

Flags

- q** Queries for a GLB server, using the standard RPC mechanism. At most, one GLB server is printed, and only servers in the current machine's cell are searched. The program exits with a status of 0 if a GLB server is found; otherwise the status is nonzero.
- v** Prints out the NCS version string.
- dl** Turns on RPC debugging while searching for GLB servers.

Examples

A network contains one **glbd** in each of two NCS cells and one **nrglbd** in a third cell.

```
/etc/ncs/lb_find
sent to broadcast address 192.92.110.255
waiting for replies
received response from glb daemon at ip:stimp(192.92.110.43)
port 1072.
received response from glb daemon at ip:oscar(192.92.110.16) port
1168.
received response from glb daemon at ip:vmess(192.92.110.21) port
1114.
.....
replicatable        ip:stimp            default            333b91c50000.0d.0
0.00.87.84.00.00.00
replicatable        ip:oscar            alternate_1        54bdad9a4000.0d.0
0.01.83.0f.00.00.00
```

```
non_replicatable ip:vmess      alternate_2      5c0e4acb8fa7.02.c  
0.5c.6e.15.00.00.00
```

Implementation Specifics

This command is part of the Network Computing System fileset.

Related Information

The **lb_admin** command.

The **glbd** (NCS) daemon, **llbd** (NCS) daemon, **nrglbd** (NCS) daemon.

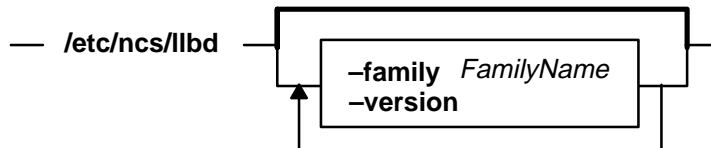
llbd (NCS)

llbd (NCS) Daemon

Purpose

Manages the information in the local location broker database.

Syntax



Description

The **llbd** daemon 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 the **llbd** daemon to support the location broker forwarding function or to allow remote access (for example, by the **lb_admin** tool) to the LLB database. In general, any host that runs an NCS-based server program should run an **llbd** daemon, 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 daemon (**glbd**).

The **llbd** daemon is started in one of two ways:

- Through the System Resource Controller (the recommended method), by entering on the command line:

```
startsrc -s llbd
```

- By a person with root user authority entering on the command line:

```
/etc/ncs/llbd &
```

TCP/IP must be configured and running on your system before you start the **llbd** daemon. (You should start the **llbd** daemon before starting the **glbd** or **nrglbd** daemon.)

Flags

-listen *FamilyList*

Restricts the address families on which an LLB listens. Use it only if you are creating a special configuration where access to an LLB is restricted to a subset of hosts in the network or internet.

The *FamilyList* is a list of the address families on which the LLB will listen. Names in this list are separated by spaces. Possible family names include **ip**.

If **llbd** is started without the **-listen** option, the LLB will listen on all address families that are supported both by NCS and by the local host.

-version

Displays the version of NCS that this llbd belongs to, but does not start the daemon.

Implementation Specifics

This command is part of the Network Computing System filesset.

Files

/etc/rc.ncs Contains commands to start the NCS daemons.

Related Information

The **lb_admin** command, **startsrc** command..

The **glbd** (NCS) daemon, **nrglbd** (NCS) daemon.

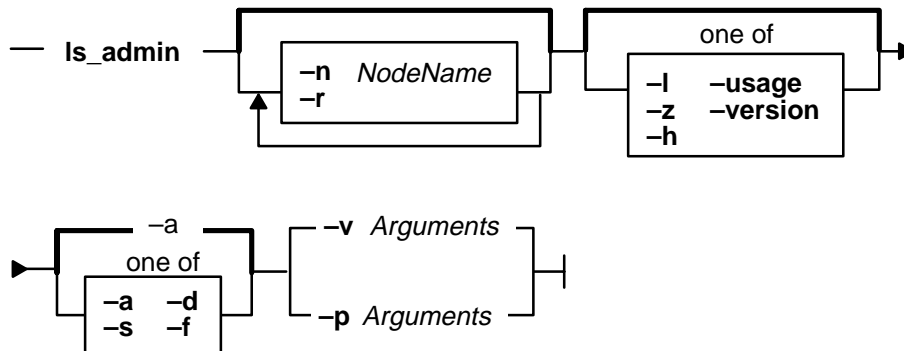
“Distributed Computing with NCS” on page 1-6.

Is_admin Command

Purpose

Displays and edits the license server database.

Syntax



Description

The **Is_admin** command allows you to examine and edit license databases. This command description describes the command line interface to **Is_admin**. The graphic interface is explained separately in Related Information.

Flags

- n *NodeName*** Indicates the server at which the license database to be edited or displayed resides. (Optional; the default value is the name of the license server node at which the command is executed.)
- r** Specifies a version of a product to be operated upon.
- l** Specifies the license annotation.
- z** Debugging flag. (Prints RPC debugging information.)
- h** Displays command usage information. (Same as **-usage**)
- usage** Displays command usage information. (Same as **-h**)
- version** Displays command version information.
- a** Adds a new vendor, a new product (and licenses), or more licenses for an existing product to the license database. This is the default value.

If adding a vendor, specify (as arguments to the **-v** option) the vendor name, vendor ID, and vendor password.

If adding a new product and licenses, specify (as arguments to the **-p** option) the vendor name, product name, product password, version text, and license annotation (if there is one) as arguments. (Do not use the **-r** option in this case).

You must have previously added the vendor in order to add its product, and you may not establish a vendor and product licenses simultaneously in a

single command line. If adding new licenses for an established version of a product, you may not specify a license annotation unless the established version had an annotation.

The same annotation must be used in all licenses for a given product (identified by the product ID and version).

The options **-a**, **-d**, and **-s** are mutually exclusive.

-s Shows information about the specified license server, vendor or product. To show information about a license server, use the **-n** option with the node name as the argument. To show information about a vendor, use the **-v** option with the name of the vendor as the argument. To show information about all vendors at a license server, use the **-v** option without an argument. To show information about a product version, use the **-r** option with the version text as the argument followed by the **-p** option with the vendor name and product name as arguments. To show information about all versions of a product use the **-r** option without an argument, followed by the **-p** option with the vendor name and product name as arguments. To show information about all versions of all products of a vendor, use the **-p** option, giving the vendor name as the only argument.

The options **-a**, **-d**, and **-s** are mutually exclusive.

-d Deletes a vendor or product from the license database. To delete a vendor, use the **-v** option with the vendor name as the argument. You may not delete a vendor unless you have previously deleted all versions of all products of the vendor at the current server, nor may you delete more than one vendor at a time. To delete a product, use the **-p** option with the vendor name and product name as arguments, followed by the license timestamp. You may not delete use-once licenses nor compound passwords that have not expired, nor may you delete more than one version of a product at a time. Use the **-s** and **-p** options to get the timestamp of the specified product licenses.

The options **-a**, **-d**, and **-s** are mutually exclusive.

-f Copies a vendor (specified with the **-v** option) from the server specified in the **-f** option to the server specified in the **-n** option, or to the default server if no **-n** server is specified.

-v Specifies the vendor to be operated upon. **-v** or **-p** and their arguments must appear last on the command line.

-p Specifies the product to be operated upon. **-p** or **-v** and their arguments must appear last on the command line.

Examples

In the following examples, argument items represented by terms such as *VendorName* and *ProductName* must appear in the command line separated by spaces. If a given argument item contains spaces, it must be enclosed in double quotes (""). For example, a *VendorName* like Acme Firmware, Inc., must appear in the actual command line as "Acme Firmware, Inc." Also, vendor and product names must be capitalized correctly.

To add a vendor:

```
ls_admin [-n NodeName] -a -v VendorName VendorID VendorPassword
```

ls_admin

To add a product or additional licenses:

```
ls_admin [-n NodeName] -a [-l Annotation] -p VendorName  
ProductName ProductPassword ProductVersion
```

Note: The `-l Annotation` parameter must be included for those products having annotations.

To show servers:

```
ls_admin [-n NodeName] -s
```

To show vendors:

```
ls_admin [-n NodeName] -s -v VendorName
```

Note: If *VendorName* is not specified, this command shows all vendors at the specified server. If no server is specified, all vendors at the default server (the one on the node the command is run from) are displayed.

To show all products for a single vendor at the specified server:

```
ls_admin [-n NodeName] -s -p VendorName
```

To show all licenses for all versions of a specified product of a specified vendor:

```
ls_admin [-n NodeName] -s -p VendorName ProductName
```

To show a specified version of a specified product of a specified vendor:

```
ls_admin [-n NodeName] -r Version -s -p VendorName ProductName
```

To copy a vendor from another server:

```
ls_admin -f NodeName -v VendorName
```

To delete a vendor:

```
ls_admin [-n NodeName] -d -v VendorName
```

Note: You cannot delete a vendor who has products listed (that is, you must delete all the products first).

To delete a product:

```
ls_admin [-n NodeName] -d -p VendorName ProductName TimeStamp
```

Note: Products must be deleted one at a time and are distinguished by their timestamps.

Implementation Specifics

This command is part of the License System Client Utilities fileset.

Related Information

The `drm_admin` (NCS) command

The `glbd` (NCS) daemon, `llbd` (NCS) daemon, `nrglbd` (NCS) daemon.

Information on Graphic Interface

The following describes the options on the graphical user interface version of `ls_admin`.

MENUS AND BUTTONS

Exit Button Exits from `ls_admin`.

Operate On: Menu

This menu lists the license server objects you can operate on: **Server**, **Vendor**, **Product**, and **License**.

- Server** Select **Server** to display a list of servers. After you select a server, you can select **Vendor** to display a list of vendors for that server, or you can select an operation to perform on the selected server from the **Operations:** menu.
- Vendor** Select **Vendor** to display a list of vendors for the server selected in the Servers list. After you select a vendor, you can select **Product** to display a list of products for that vendor, or you can select an operation to perform on the selected vendor from the **Operations:** menu.
- Product** Select **Product** to display a list of products for the selected server and vendor. After you select a product, you can select **License** to display a list of licenses for that product, or you can select an operation to perform on the selected product from the **Operations:** menu.
- License** Select **License** to display a list of license records for the selected server, vendor, and product. Each record shows the number, type, and terms of the licenses. Select a license record and select an operation from the **Operations:** menu.

Operations: Menu

This menu lists the license database operations you can perform. The contents of this menu vary depending on the object (**Server**, **Vendor**, **Product**, or **License**) selected in the **Operate On:** menu.

OPERATIONS ON SERVERS

Check user file Verifies that the format of the file **user_file**, located in the **/usr/lib/netls/conf** directory, is valid.

Update server list

Updates server and/or license database information. The information displayed is current, so it is generally unnecessary to use **Update server list** unless a communications failure has been repaired or a new server has been started since you invoked **ls_admin**, or another user is currently editing a license database with **ls_admin**.

Add vendor Adds a vendor to the selected license database. Enter the vendor name, vendor ID, and vendor password on the pop-up; then select **Add vendor**.

Describe Provides detailed information about the selected server, including socket information, target type, and target ID.

OPERATIONS ON VENDORS

Add product Adds a product to the selected vendor at the selected server. Enter the product name, version, product password, and license annotation (if there is one) on the pop-up; then select **Add vendor**. If you add a product to more than one server, be sure to use exactly the same product name at all servers. Note that **Add product** performs two functions: it establishes a new product, and it adds licenses for the product. To add more licenses for an existing product, select the product and then use **Add licenses**.

Rename Renames the selected vendor. Enter the new vendor name on the pop-up. If you rename a vendor at one server, you should also rename it (using the same name) at all servers where that vendor is listed.

Is_admin

- Delete** Deletes the selected vendor at the selected server. Select the **Delete?** pop-up to confirm the operation. Move the cursor off the pop-up to cancel the operation. You cannot delete a vendor that has active licenses for its products.
- Copy vendor** Copies the selected vendor to another server's license database. Select the server to which you want the vendor copied from the pop-up that appears.
- Describe** Provides detailed information about the selected server and vendor, including the vendor ID.

OPERATIONS ON PRODUCTS

- Add licenses** Adds licenses to the selected product. Enter the license password on the pop-up. (Use **Add licenses** only to add more licenses for an existing product. If you are both establishing a new product and adding licenses for the product, use **Add product** rather than **Add licenses**.)
- Rename** Renames the selected product. Enter the new product name on the pop-up. If you rename a product at one server, you should also rename it (using the same name) at all servers where that product is listed.
- Describe** Provides detailed information about the selected server, vendor, and product. Product information displayed includes the ID, annotation string, and the number, type, and date of existing licenses for the product.

OPERATIONS ON LICENSES

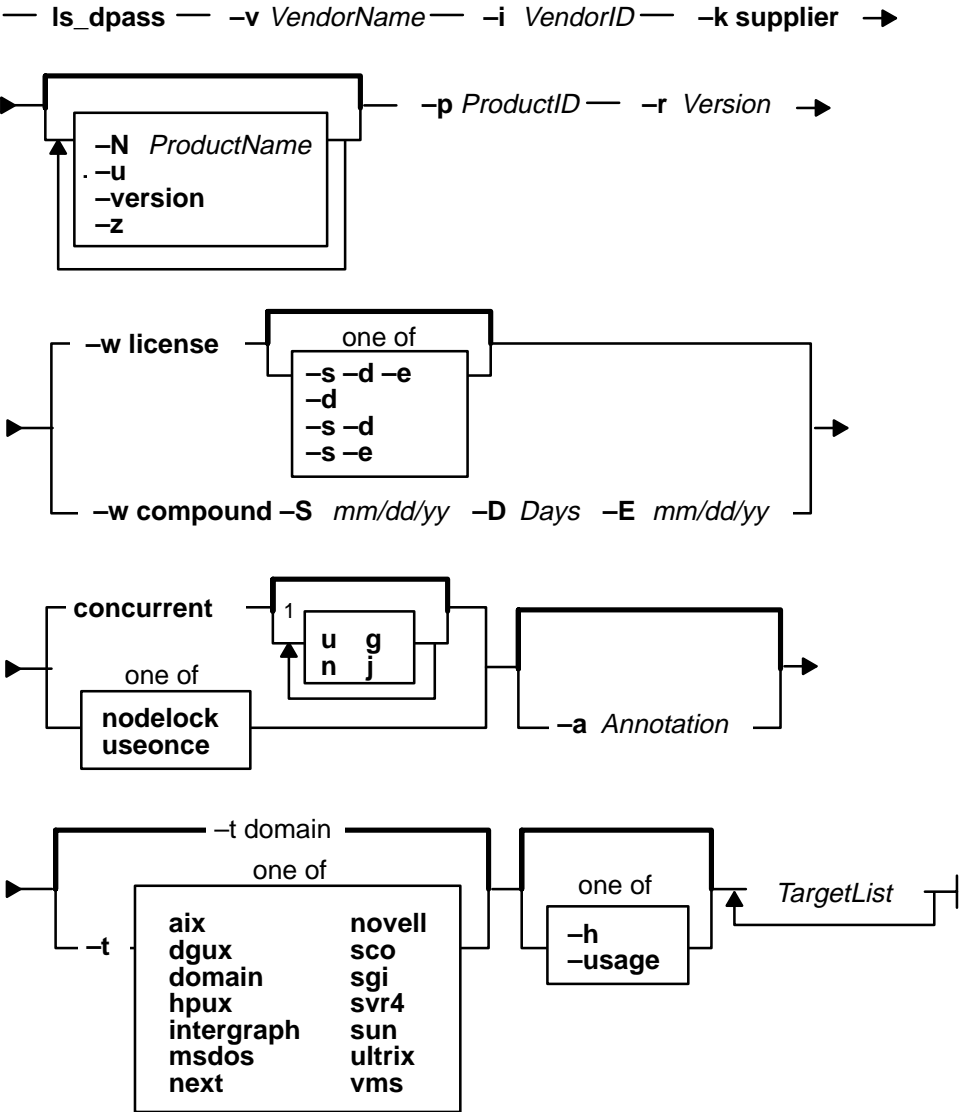
- Delete** Deletes the selected license record. This enables you to get rid of expired licenses. Select the **Delete?** pop-up to confirm the operation, or move the cursor off the pop-up to cancel.
- Describe** Provides detailed information about the selected server, vendor, product, and license record. License information displayed includes the number, type, date, and timestamp.

Is_dp^{ass} Command

Purpose

Create passwords for iFOR/LS-licensed software from compound licenses.

Syntax



1 Do not use a space between these items

Description

The **Is_dp^{ass}** command creates passwords for customers and distributors of enabled software products. **Is_dp^{ass}** features both a command line and graphical interface. If all

ls_dpass

options are omitted, or the **x** argument delimiter is found on the command line, **ls_dpass** invokes a graphical interface. Both interfaces are described here.

Flags

- v** *VendorName* Specifies the vendor name.
- i** *VendorID* Specifies the vendor ID. Supply the vendor ID specified by the provider of the compound licenses.
- k** **supplier** Use the keyword **supplier**: this causes **ls_dpass** to use the supplier's vendor key (which is known to the license server) to encode the passwords.
- N** *ProductName* (Optional) Specifies the product name. If an argument is not supplied, a product name of the form "Product <product ID>" is created by **ls_dpass**.
- p** *ProductID* Specifies the product ID.
- r** *Version* Specifies the product revision text.
- w** Specifies the password type; supply one of the following keywords: **license** or **compound**. If the password type is **compound**, you must also supply the derived start/end dates (**-S**, **-E**) and the aggregate duration in days (**-D**).
- S** *mm/dd/yy* (Compound passwords only). Specifies the derived license start date. This is the date before which no license password derived from the compound password is valid.
- D** *NumberOfDays* (Compound passwords only). Specifies maximum aggregate duration (in days) of all derived passwords.
- E** *mm/dd/yy* (Compound passwords only). Specifies the derived license end date. This is the date after which no license password derived from the compound password is valid.
- l** Specifies the license type; use one of the following keywords:
 - nodelock**
 - useonce**
 - concurrent** If you specify **concurrent**, you may optionally define multiple-use rules for the licenses being created.
- m** (Optional) Specifies the rules whereby multiple invocations of a product require only a single concurrent license. You can specify the rules as any combination of the following arguments: **u** (same user), **n** (same node), **g** (same group), **j** (same job ID).

For example, the specification **-m un** means that, if the user and node are the same as those associated with a previously granted license, then any succeeding invocations of the product will not require any additional concurrent-use licenses.

Note: Arguments to the **-m** option are specified without separating spaces, commas, etc. (**ungj**).

- a Annotation** (Optional) Specifies the license annotation (up to 80 characters).
- s mm/dd/yy** Specifies the start date of the password. If the password type is **license**, then this value specifies the start date of the licenses; if the password type is **compound**, then this value specifies the start date for creating license passwords derived from the compound password.

Note: If this option is omitted, the start date of the password defaults to the current date as the start date. Start dates cannot be “before” the current date.

- d NumberOfDays** Specifies the duration of the password. If the password type is license, then this value specifies the number of days for which the licenses are valid; if the password type is compound, then this value specifies the number of days during which license passwords may be derived from the compound password.
- e mm/dd/yy** Specifies the end date of the password. If the password type is license, then this value specifies the end date of the licenses; if the password type is compound, then this value specifies the end date for creating license passwords derived from the compound password.

Note: Valid combinations of the following **-s**, **-d**, and **-e** options are as follows:

- d** alone **-s** defaults to the current date; **ls_dpass** calculates the expiration date for you.
- s** and **-d** **ls_dpass** calculates the expiration date for you.
- s** and **-e** **ls_dpass** calculates the duration for you.

- t** Specifies the target type; default if omitted: domain. Or supply one of the following keywords: **aix**, **dgux**, **domain**, **hpux**, **intergraph**, **msdos**, **next**, **novell**, **sco**, **sgi**, **svr4**, **sun**, **ultrix**, or **vms**. (See the release notes for the latest list of supported targets).
- u** (Optional) Generates **ls_admin** command lines as part of the **ls_dpass** output. These **ls_admin** command lines can be used to install the passwords generated by **ls_dpass**.
- n** Number of licenses. Supply the total number of licenses over all target IDs on the list.
- h** Command usage information. (Same as **-usage**)
- z** Debugging flag. (Prints RPC debugging information)
- usage** Command usage information. (Same as **-h**)

TargetList This argument must come at the end of the command line. Enter a list of target IDs separated by spaces.

All other target types must specify either a target ID or a date. Enter a date in the following format: *mm/dd/yy*.

Security

Access Control: Only root users have execute (x) access to this command.

Files Accessed: **lic_db**

Event	Information
lic_db	Decrements number of compound licenses available.

ls_dpass

Examples

1. To create a nodelocked password for a single node:

```
ls_dpass -v vendor -i vendor_uuid -k supplier -N product -p 4 \  
-r 4.0 -w license -l nodelocked -s 90/02/07 -d 5 -t ibm/aix -u -n  
1 21a9a
```

2. To create a nodelocked password for multiple nodes:

```
ls_dpass -v vendor -i vendor_uuid -k supplier -N product -p 4 \  
-r 4.0 -w license -l nodelocked -s 90/02/07 -d 5 -t ibm/aix -u -n  
4 21a9a \ 20add fb40 18fa0
```

Note: When creating nodelocked passwords, the total number of licenses specified by **-n** must equal the number of target IDs in the list.

3. To create concurrent-use licenses for a single node:

```
ls_dpass -v vendor -i vendor_uuid -k supplier -N product -p 4 \  
-r 4.0 -w license -l concurrent -s 90/02/07 -d 5 -t ibm/aix -u -n  
1 21a9a
```

Notes:

1. When creating concurrent-use licenses for multiple nodes, the total number of licenses specified by the **-n** switch will be evenly divided among the total number of servers specified in the target list.
2. Use-once passwords work the same way that concurrent-use licenses do.

Implementation Specifics

This command is part of the Network Computing System fileset.

Files

/usr/lib/netls/conf/products

Products database for creating licenses.

Related Information

Information on Graphic Interface

The following describes the options on the graphical user interface version of **ls_dpass**.

MENUS AND BUTTONS

- | | |
|--------------------|--|
| Exit Button | Select this button to exit from ls_dpass . |
| Select Menu | Select an item from this menu to specify the type of object you want to work with (vendor, product, password, or customer). |
| Vendor | Select this button to display a list of vendors in the List box, and a menu of vendor-related commands in the Command box. Select a vendor. Then select either a vendor-related command to operate on the vendor list, or Product to display a list of products for the vendor you selected. |
| Product | After you have selected a vendor, select this button to display a list of the vendor's products in the List box, and a menu of product-related commands in the Command box. Select a product. Then select either a product-related command to operate on the vendor list, or Customer to display a list of customers for the product you selected. |

- Password** After you have selected a vendor, product, and customer, select this button to display information fields related to the creation of passwords for the selected customer.
- Customer** Select this button to display a list of customers in the List box, and a menu of customer-related commands in the Command box.

Vendor-related Commands

Add New Vendor

Select this button to define a new vendor. Enter the vendor name and vendor ID on the form that pops up. Then select **Add Vendor** to establish the vendor, or **Cancel** to cancel the operation.

Note that, as a distributor, before you can create license passwords, you must first use **Is_admin** to install the licensor's vendor password and compound password(s) for the product (the licensor supplies these passwords).

(See "Graphic Interface Data Entry Fields" on page A-26 for a description of the fields in the Add Vendor popup).

- Show Vendor** After selecting a vendor from the List box, select this button to display vendor information, including the vendor's name and ID.
- Delete Vendor** Select this button to delete a vendor from the vendor list. A popup appears prompting you to confirm that you want to perform the delete. Select the popup to delete the Vendor. If you do not want to delete the vendor, move the cursor off the popup and it will disappear from the screen.

Product-related Commands

Add New Product

Select this button to define a new product. If your company is the original licensor of the product, enter the product name, product ID, and version text on the form that pops up. Then select Add product to establish the product, or Cancel to cancel the operation.

(See "Graphic Interface Data Entry Fields" on page A-26 for a description of the fields in the Add Product popup).

- Show product** After selecting a product from the List box, select this button to display product information, including the product name and product ID.
- Delete product** Select this button to delete a product from the product list. A popup appears prompting you to confirm that you want to perform the delete. Select the popup to delete the product. If you do not want to delete the product, move the cursor off the popup and it will disappear from the screen.

Password-related Commands

Password type:

Select the button to the right of the label Password type: to toggle between **License** (default) and **Compound**.

- License type:** Select the button to the right of the label License type: to display a menu of iFOR/LS license types, from which you can choose one. The types are **concurrent**, **use once**, and **node locked**.

Multiple-Use Rules

Use this menu to specify the rules whereby multiple invocations of a product require only a single concurrent-use license. Do not specify

ls_dpss

different rules for passwords for any single version of a product that are destined for installation in the same network environment.

- Same User** Check this item to indicate that only a single concurrent-use license is required for multiple invocations of the product so long as the same user is invoking the product.
- Same Group** Check this item to indicate that only a single concurrent-use license is required for multiple invocations of the product so long as the invocations originate from the same group.
- Same Node** Check this item to indicate that only a single concurrent-use license is required for multiple invocations of the product so long as the product is being invoked at the same node.
- Same Job** Check this item to indicate that only a single concurrent-use license is required for multiple invocations of the product so long as the invocations are associated with the same job ID.
- Exit** Exits from the multiple-use rules menu.
- Target type:** Select the button to the right of the label Target Type: to display a menu of target types from which you can choose one to specify the type of node for which you are creating passwords. The default choice is **AIX**. Other choices include: **DGUX, Domain, HPUX, Intergraph, MSDOS, NeXT, Novell, SCO, SGI, SVR4, Sun, Ultrix, and VMS**.
- Next target** Select this button to switch to the next target.

Create Passwords

Creates passwords based on the product/vendor data specified. If you have used the Output file option, this information is saved in the file you specify. Note that when you create license passwords, **ls_dpss** decrements the number of compound licenses available according to the type and number of licenses specified.

- Create script** **ls_dpss** can output scripts that customers can use to automate the installation of the passwords. The script is appended to the **ls_dpss** transcript. If you want **ls_dpss** to generate the shell script, select the check box.
- Output file** Use this to enter a filename in which you want customer passwords to be saved. (Optional) You must select this button before you select Create Passwords.

Graphic Interface Data Entry Fields

Vendor Information

Vendor name, Vendor ID, Vendor key

Enter the vendor name and vendor ID. If the vendor ID has not already been established, you may use the Create Vendor ID button to generate one.

Product Information

Product name, Product ID, Product Version

Enter the product name, product ID, and the version.

Password Information

Number of targets:

Enter the total number of target nodes on which passwords are to be installed. (Optional; default is 1.)

Number of Licenses (total):

Enter the total number of licenses to be created (that is, the aggregate of all licenses specified by all passwords to be created in this session).

License annotation

Enter an annotation of up to 80 characters for the licenses (optional). The software product defines the annotation, and when licenses are created, **Is_dpss** outputs the annotation along with the passwords. If there is no annotation, leave this field blank. Do not specify different annotations for passwords for any single version of a product that are destined for installation in the same network environment.

Target n of n Indicates the target for which **Is_dpss** is currently displaying password information.

Target id: Enter the target ID. The passwords generated are installable only at the target having the specified ID.

Start: If the password type is **License**, enter the start date for the licenses (the licenses become effective at midnight on the day before the specified date). This date can't be earlier than the current date. (Default is the current date.)

If the password type is **Compound**, enter the start date for the compound password(s) (passwords become effective at midnight on the day before the specified date). This date can't be earlier than the current date. (Default is the current date.)

Duration (days):

If the password type is **License**, enter the duration of the licenses (in days); or skip this field and enter the expiration date instead. The maximum duration of a license is 4096 days. (Default is 0.)

Expiration: If the password type is **License**, enter the expiration date of the licenses in date format (licenses expire at midnight on the specified date). If you prefer, skip this field and enter the duration in days instead. The latest expiration date may be no more than 4096 days after the start date. (Default is today's date, corresponding to a duration of 0 days.)

If the password type is **Compound**, enter the expiration date of the password(s) in date format (passwords expire at midnight pm on the specified date). The latest expiration date may be no more than 4096 days after the start date. (Default is today's date, corresponding to a duration of 0 days.)

Derived license start:

Enter the earliest start date for licenses that are to be derived from a compound password (this item is not applicable to license passwords). The derived licenses may start later, but not earlier, than the date you specify here.

Derived license expiration:

Enter the latest expiration date for licenses that are to be derived from a compound password (this item is not applicable to license passwords). The derived licenses may expire earlier, but not later, than the date you specify here.

ls_dpass

Aggregate duration (days):

Enter the aggregate duration of all licenses that are to be derived from a compound password (this item is not applicable to license passwords). For example, a compound password from which 100 licenses may be derived might have an aggregate duration of 36500 days. From this password may be derived 100 1-year licenses, or 50 6-month licenses and 50 18-month licenses, and so on.

Number of licenses (this target):

Enter the number of licenses to be installed on the current target if this number is different from the default number shown here. (By default, **ls_dpass** divides the total number of licenses to be installed by the number of targets on which the licenses are to be installed.) (This information applies only to concurrent-use and use-once license types; passwords for nodelock licenses are always one per target.)

Customer Information

Customer name, address, address, contact

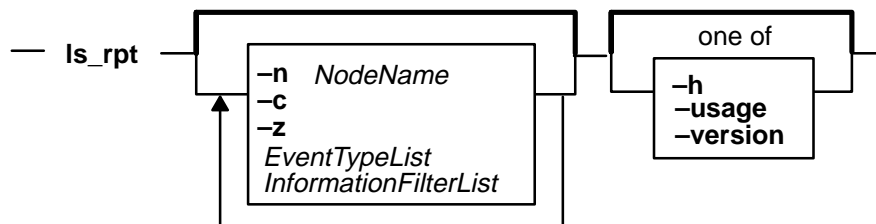
Use these fields to add the name and address of a new customer. The customers file stores customer names, addresses, and contacts.

Is_rpt Command

Purpose

Reports on network license server events

Syntax



Description

The **Is_rpt** command generates reports on license server events. There is no graphic interface for this command.

Flags

- n** Specifies the server node about which the report is to be generated. If you do not specify a node, **Is_rpt** reports on the current server node.
- c** Lists data in 80-column format.
- z** Debugging flag. (Prints RPC debugging information.)
- h** Displays command usage information. (Same as **-usage**)
- usage** Displays command usage information. (Same as **-h**)
- version** Displays command version information.

EventTypeList You can specify any combination of the following event types. Specify **-a** to specify all event types.

- a** Lists all log messages.
- l** Lists all license-related events (product received license, product release license to server, user entered license queue, user exited queue. This is the default option.
- e** Lists all error events
- s** Lists all server start/stop events.
- m** Lists all messages that were logged by a software product or license server.
- f** Lists any fatal error events.
- d** Lists all license database modification messages.

InformationFilterList

You can choose any combination of the following information filters. If no filters are specified the default is all dates, all vendors, all products, all users.

- b mm/dd/yy** Lists events that occurred beginning at the specified date.
- t mm/dd/yy** Lists events that occurred up to the specified date.

ls_rpt

- v *VendorName*** Lists events related to the specified vendors.
- p *ProductName*** Lists events related to the specified products.
- u *UserName*** Lists events related to the specified users.
- r 1** Lists, for the specified product, the number of requests for licenses, the number of licenses granted, and the percent of rejected requests.
- r 2** Lists the same information as **-r 1**, but also includes user names and the number of licenses installed.
- x *mm/dd/yy*** Deletes log file entries written on and before the specified date.

Examples

1. List license events on the local server node:

```
ls_rpt
```

2. List errors and fatal errors occurring between August 31 and September 30, 1990 on the server node `plums`:

```
ls_rpt -n plums -e -f -b 08/31/90 -t 09/30/90
```

3. List all messages logged at `mars` by the vendor `XYZ`:

```
ls_rpt -n mars -m -v xyz
```

4. Delete all log entries created on or before May 1, 1994 on the server `mars`:

```
ls_rpt -n mars -x 5/1/94
```

Implementation Specifics

This command is part of the License System Client Utilities fileset

Related Information

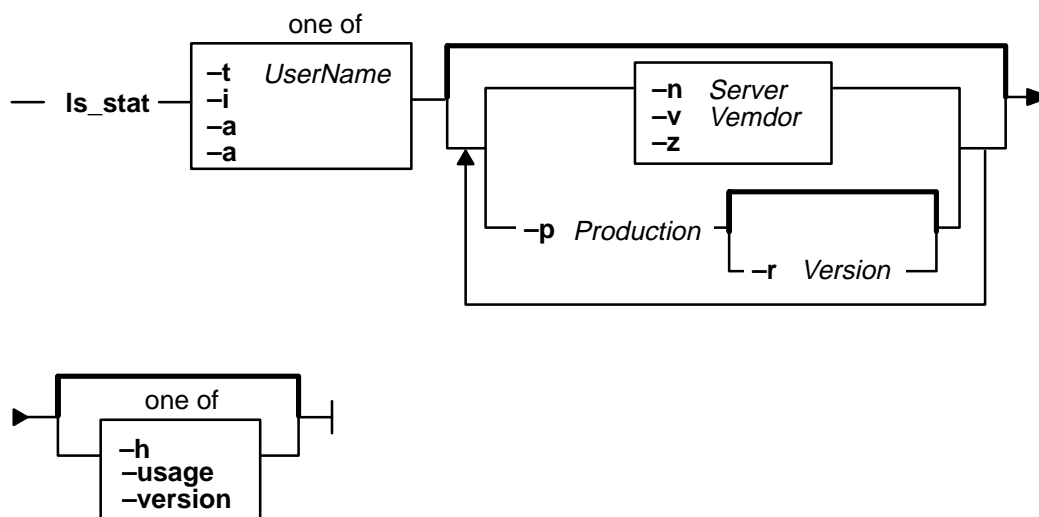
The `netlsd` daemon.

Is_stat Command

Purpose

Displays the status of the license server system.

Syntax



Description

The **Is_stat** command provides status information on network licenses (that is, all license types except nodelocked). End users as well as system administrators may find **Is_stat** useful for finding out the status of licenses. This command description explains the command line interface of **Is_stat**. The graphic interface is explained separately in Related Information.

Flags

- t** Displays a table of total license usage compared to installed licenses; all servers and all products are listed by default.
- i** Displays installed licenses; all servers and all products are listed by default.
- a** Displays information about all concurrent-use license users; all servers and all products are listed by default.
- u *UserName*** Displays licenses being used by the specified user.

Note: One of the options listed above must be included in all **Is_stat** command lines.

- n *Server*** Displays licenses located at the specified server.
- v *Vendor*** Displays licenses of the specified vendor; if the vendor string contains spaces, it must be delimited by single or double quotes.
- p *Product*** Displays licenses for the specified product; if the product string contains spaces, it must be delimited by single or double quotes.

ls_stat

<code>-r</code>	<i>Version</i>	Displays licenses for the specified revision of a product specified by <code>-p</code> ; if the version string contains spaces, it must be delimited by single or double quotes.
<code>-z</code>		Debugging flag. (Prints RPC debugging information.)
<code>-h</code>		Displays command usage information. (Same as <code>-usage</code>)
<code>-usage</code>		Displays command usage information. (Same as <code>-h</code>)
<code>-version</code>		Displays command version information.

Examples

1. To display all licenses installed for all products on all servers:

```
ls_stat -i
```

2. To display licenses in use from the server `park`:

```
ls_stat -a -n park
```

3. To display licenses installed and currently in use for the product Kwik-Draw, Version 2.1:

```
ls_stat -a -i -p Kwik-Draw -r 2.1
```

4. To display licenses installed on `park` for the vendor `Apollo`.

```
ls_stat -i -v Apollo -n park
```

Implementation Specifics

This command is part of the License System Client Utilities fileset.

Related Information

Information on Graphic Interface

The following describes the options on the graphical user interface version of `ls_stat`.

MENUS AND BUTTONS

Exit Button Select this button to exit from `ls_stat`.

License Information Menu

This menu contains three buttons: **Installed**, **Usage**, **All Users**, and **User**. After you have selected a server and product from the **Server** and **Product** lists, select these buttons to display information about users, installed licenses, and usage of the selected server and product.

Installed Button

Displays information, listed by vendor, product and server, about product licenses installed at selected servers, including number of active licenses, their start and end dates, their type, the number of licenses currently in use, and the length of the queue of users waiting for licenses.

Usage Button Displays information, listed by vendor, product and server, about the usage of products, including number of licenses in use, total number of licenses, and number of licenses available.

All Users Button

Displays information, listed by vendor, product and server, about current users of licensed products, including user ID, node name, group, number of licenses held, and start time.

User Button Displays information, listed by vendor, product and server, about a specific user of licensed products, including user ID, node name, group, number of licenses held, and start time. After the User button is selected, a pop-up dialog is displayed in which you may enter a user ID.

Server List Box

This list box, directly to the right of the License Information menu, displays the server list. At the top of this box is the All Servers (Update) button (see below). At the left of the box is a scroll bar that you can use to scroll the list.

All Servers (Update) Button

Select this button to poll the network and update the server list. When you select this button, a check mark appears in the box at its left. A check mark in this box indicates that:

- All existing servers are displayed in the **Server List** box.
- The vendors and products listed in the **Product List** box are the vendors and products existing at the server currently selected in the Server List box.
- After updating the server list, select a server to display (in the Product List box) the products it administers. Next, select a product (or **All**) from the list of products; then select **Users, Installed, or Usage**.

Product List Box

This list box, directly to the right of the **Server List** box, displays the server list. At the top of this box is the **All Products (Update)** button (see below). At the left of the box is a scroll bar that you can use to scroll the list.

All Products (Update) Button

Select this button to poll the network and update the product list. When you select this button, a check mark appears in the box at its left. A check mark in this box indicates that:

- All existing vendors and products are displayed in the **Product List** box.
- The servers listed in the **Server List** box are the servers that hold licenses for the product currently selected in the **Product List** box.

After updating the product list, select a product to display (in the Server List box) the servers holding licenses for the product. Select a server (or **All**) from the list of servers; then select **Users, Installed, or Usage**.

Status Message Field

This field, across the bottom of the window, describes the information currently displayed in the **Server List** box and the **Product List** box.

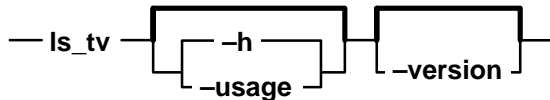
ls_tv

ls_tv

Purpose

Verifies that license servers are working.

Syntax



Description

The **ls_tv** tool requests a concurrent-use license from a license server and prints a list of active license servers.

If you can run **ls_tv** successfully but are still having a problem with a license product, the problem is probably with the licenses, or possibly with the product itself: in this case, talk to the vendor of the licensed software product.

If you cannot run **ls_tv** successfully and receive one of the error messages listed below, use the explanation of the error to fix the problem. Then try running **ls_tv** again.

If you cannot run **ls_tv** successfully and receive an error that is not listed below, it means there is a problem with the software on which the license server is layered (for example, TCP/IP or NCS), or a hardware problem.

Flags

- | | |
|-----------------|---|
| -h | Displays command usage information. (Same as -usage .) |
| -usage | Displays command usage information. (Same as -h .) |
| -version | Displays command version information. |

Error Messages

netls_request_license

Communication failure(networking computing system/RPC run time)

netls_no_svrs_found

No servers available for this vendor.

netls_license_not_found

License not found in the database.

netls_not_authorized

The user is not authorized to use this product.

netls_bad_timestamp

Time disparity too large.

For recovery information relating to these error messages, please refer to "ls_tv Errors" on page 4-4.

Example

To run the license-server test-and-verification tool:

ls_tv

The system responds with something similar to the following:

```
LS-TV Version 2.0 -- iFOR/LS Test and Verification Tool
Copyright 1991, Hewlett-Packard Company, All Rights Reserved
Copyright 1991, Gradient Technologies, Inc. All Rights Reserved
Completed license transactions on node 1f9a4 running iFOR/LS
Active iFOR/LS Servers:
    altair (AIX) running iFOR/LS
    cobweb (AIX) running iFOR/LS
```

Implementation Specifics

This command is part of the License System Client Utilities fileset.

Related Information

The **netlsd** daemon.

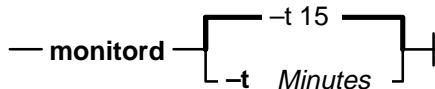
monitord Daemon

Purpose

Communicates with the iFOR/LS server and requests an AIX Version 4.1 concurrent-use license for each countable login.

Syntax

monitord [**-t** *Minutes*]



Description

The AIX Version 4.1 BOS has multiple ways to access the system, and each of them has a different behavior upon exit. The **monitord** daemon provides a common interface to the iFOR/LS **netlsd**. **monitord** communicates with the iFOR/LS server and requests an AIX Version 4.1 concurrent-use license for each countable login.

Note: The iFOR/LS licensing mechanism is used only if the system has the *floating license mode* enabled.

After user logout, **monitord** requests **netlsd** to release the specific AIX Version 4.1 license the user was using, in order to make it available for further logins.

monitord is started when the **chlicense -f on** command is used to enable the *floating license mode*. When the *floating license mode* is enabled, **monitord** is started upon system startup via an entry in **/etc/inittab**. The default (invoked without **-t** option) is an interval of fifteen minutes.

The entry in **/etc/inittab** looks like the following:

```
monitord:2:once:/usr/sbin/monitord >/dev/console 2>&1
```

Flags

-t *Minutes* Sets the value in minutes of the heartbeat interval. A value of 0 sets an infinite interval. The default is fifteen minutes.

Implementation Specifics

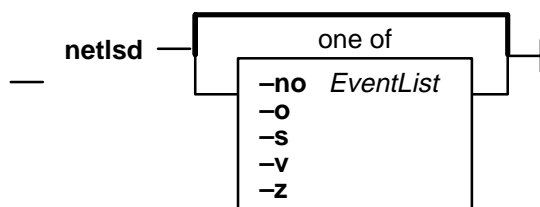
This command is part of the License Management fileset.

netlsd Daemon

Purpose

Starts an iFOR/LS license server on the local node. There is no graphical interface for this command.

Syntax



Description

netlsd starts an iFOR/LS license server on the local node. There is no graphical interface for this command.

Flags

- no** Turns off logging of the events specified in *EventList*. 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** License-grant and license-release events.
 - c** License check-in events. (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** Error events.
 - 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 may run after it has lost contact with the license server.
 - m** Message events.
 - s** License server start/stop events.
- o** Overrides the in-use flag at a license server database. While a license server is running, its database is flagged as being in use to prevent more

netlsd

than one server from running on the same node. When a license server stops running, the flag is reset. However, if a license server exits abnormally, the flag may not be reset, which prevents the server from restarting. In overriding the in-use flag, this option allows the server to be restarted. Do not use **-o** unless you are sure the license server is not running.

- v** iFOR/LS library verbose mode.
- s** When the **-s** flag is present, **netlsd** ignores database-manipulation requests that are not local. Otherwise, remote requests to change the license database are accepted, provided the remote user is authorized.
- z** Debugging flag. (Prints RPC debugging information.)

Examples

1. Start a license server; do not log check-in, vendor, product, timeout, or message events:

```
netlsd -no cvptm
```

2. Start a license server, overriding the in-use flag:

```
netlsd -o
```

Files

lic_db	Contains vendor, product and license details. (Encrypted binary file)
lic_db.bak	Backup of lic_db . Used for recovery. (Encrypted binary file)
cur_db	Run-time cache of current license status. (Encrypted binary file)
log_file	Log of license server events. (Binary file)
user_file	List of users who may and may not obtain licenses. (ASCII file)

Implementation Specifics

This command is part of the License System Server fileset.

Related Information

The **glbd** (NCS) daemon, **llbd** (NCS) daemon.

"Distributed Computing with NCS" on page 1-6.

nrglbd Daemon (NCS)

Purpose

Manages the global location broker database.

Syntax

```
— /etc/ncs/nrglbd — 
  -version
```

Description

The **glbd** daemon manages the global location broker (GLB) database. The GLB database, part of the Network Computing System (NCS), helps clients to locate servers on a network or internet. The GLB database stores the locations (that is, the network addresses and port numbers) of servers on which processes are running. The **glbd** daemon maintains this database and provides access to it.

There are two versions of the GLB daemon, **glbd** and **nrglbd**. You should run only one **nrglbd** on a network or internet, and you should not run a **nrglbd** and a **glbd** on the same network or internet.

The **nrglbd** daemon is typically started in the background; it can be started in one of two ways:

- By a person with root user authority entering on the command line:


```
/etc/ncs/nrglbd &
```
- Through the System Resource Controller (SRC), by entering on the command line:


```
startsrc -s nrglbd
```

TCP/IP must be configured and running on your system before starting the **nrglbd** daemon. The **llbd** daemon must also be started and running before you start the **nrglbd** daemon.

Flags

-version Displays the version of NCS that this **nrglbd** belongs to, but does not start the daemon.

Files

/etc/rc.ncs Contains commands to start the NCS daemons.

Implementation Specifics

This command is part of the Network Computing System fileset.

Related Information

The **lb_admin** command.

The **llbd** daemon.

nrglbd

Appendix B. iFOR/LS Files

glb_obj.txt

Purpose

Specifies the object UUID of the global location broker.

Syntax

`/etc/ncs/glb_obj.txt`

Description

The global location broker (GLB) is an object identified by a universal unique identifier (UUID). This UUID has a default value. The **glb_obj.txt** file allows the default value to be overridden by specifying a different GLB object UUID for a particular host.

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 hosts do not need to have a **glb_obj.txt** file.

If a host has a **glb_obj.txt** file, the UUID in the file identifies the GLB object to which that host will direct lookups and updates. If the host runs a GLB daemon (**glbd** or **nrglbd**), the UUID also identifies the GLB object managed by that daemon, and the daemon will accept lookups and updates only for that object. By specifying different GLB object UUIDs on different sets of hosts, the network or internet can be partitioned into location broker “cells.”

Location broker cells have independent GLB databases. Each cell can be serviced by one **glbd**, one **nrglbd**, or a set of **glbd** replicas. All hosts in a cell use the same GLB object UUID. Cells need not correspond in any way to physical or logical network topology.

A **glb_obj.txt** file consists of one line containing the textual representation of a UUID, such as is generated by the **uuid_gen** utility. The contents of **glb_obj.txt** are identical on all hosts in a cell.

If a host does not have a **glb_obj.txt** file, it will use the default value for the GLB object UUID.

Example

The following is a sample of what a **glb_obj.txt** file would contain:

```
437f28e72000.0d.00.00.fb.40.00.00.00
```

Related Information

The **glbd** (NCS) command.

glb_site.txt

Purpose

Lists possible global location broker sites.

Syntax

`/etc/ncs/glb_site.txt`

Description

The **glb_site.txt** file lists the network addresses of hosts where a global location broker (GLB) daemon may be running.

There are two versions of the GLB daemon: **glbd** and **nrglbd**. AIX Version 4.1 supports both.

Ordinarily, programs contact a GLB by broadcasting on the local network. However, some systems do not support broadcasting. Also, in certain internet configurations, not every network can have a GLB. (This typically occurs in internets that use **nrglbd**, but it can also occur in an internet that uses **glbd** if not all networks include a host that can run a **glbd**.) For hosts that cannot locate a GLB via broadcast, the **glb_site.txt** file provides a list of addresses where the host can try to directly contact a GLB.

Each line in **glb_site.txt** contains a network address where a GLB may be running. Hosts that have a **glb_site.txt** file try these addresses in order. Each address has the following form:

```
Family:Host
```

The *Family* is the textual name of an address family. Possible values include **ip** and **dds**.

The *Host* is a host name. A leading **#** can be used to indicate that the host name is in the standard numeric form (for example, `#192.9.8.7` or `#515c.111g`).

Blank lines and lines beginning with **#** are ignored.

If a host has a **glb_site.txt** file but does not find a GLB at any of the addresses listed in the file, the host then tries to locate one via broadcast.

Example

The following are sample **glb_site.txt** files for the IP family:

```
ip:piglet
ip:#192.9.8.7
```

Related Information

The **glbd** (NCS) command.

The **nrglbd** (NCS) command.

ARK Log File

For system administrators who wish to write their own ARK report generators, this appendix gives examples of **log.h**, the license server log file, and **log_imp.idl**, the network interface definition language source file. Each file is located in the **/usr/lib/netls/bin** directory.

Format of the Log File

The ARK Log File has the following format:

Version (4 bytes) int log_version
Header record (8 bytes) short type short length
Log record Format depends on record type
Header record (8 bytes) short type short length
Log record Format depends on record type

Sample Log File – log.h

```
/* version of log file */  
  
#define LOG_VRSN -1005  
  
/* user stipulation of what goes into log file */  
  
#define LOG_LIC_BIT    0x0001 /* license grant/release */  
  
#define LOG_CHK_BIT    0x0002 /* license check */  
  
#define LOG_WQU_BIT    0x0004 /* wait queue */  
  
#define LOG_VND_BIT    0x0008 /* vendor/db */  
  
#define LOG_PRD_BIT    0x0010 /* product/db */  
#define LOG_ERR_BIT    0x0020 /* error */  
  
#define LOG_TIO_BIT    0x0040 /* license timed out */  
  
#define LOG_MSG_BIT    0x0080 /* message */  
  
#define LOG_SVR_BIT    0x0100 /* server start/exit */  
  
/* command line letters for above events */  
  
#define log_stip_events "lcwvpetms"  
  
static short log_bits_array[] = {
```

```

        LOG_LIC_BIT,

        LOG_CHK_BIT,

        LOG_WQU_BIT,

        LOG_VND_BIT,

        LOG_PRD_BIT,

        LOG_ERR_BIT,

        LOG_TIO_BIT,

        LOG_MSG_BIT,

        LOG_SVR_BIT

};

/* ACTIONS for LS - Used in 'action' field of log_action_rec_t*/
/* and log_action_rec_ll_*/

#define LIC_GRNT      1

#define LIC_RLS 2

#define LIC_CHK 3

#define WAITING 4

#define WAIT_GNT      5

#define WAIT_RMV      6

#define LIC_GRNT_MULTI  7

#define LIC_RLS_MULTI   8

static char *action_name[] = {

    " ",

    "G R A N T E D",

    "R E L E A S E D",

    "C H E C K E D",

    "W A I T I N G",

    "G R A N T / W A I T",

    "W A I T   R E M O V E D",

```

```

        "M U L T I P L E   G R A N T",
        "M U L T I P L E   R E L E A S E"
};

/* ACTIONS for DB */

#define VEND_ADD 1
#define VEND_DEL 2
#define VEND_REN 3
#define PROD_ADD 4
#define PROD_DEL 5
#define PROD_REN 6

static char *db_action_name[] = {
    " ",
    "V E N D O R   A D D E D",
    "V E N D O R   D E L E T E D",
    "V E N D O R   R E N A M E D",
    "P R O D U C T   A D D E D",
    "P R O D U C T   D E L E T E D",
    "P R O D U C T   R E N A M E D"
};

/*****/
/***** PROTOTYPES *****/
/*****/

long init_log();
void close_log();
void cleanup_log(
#ifdef __STDC__
long begin_date,
long *status

```

```

#endif
);

void log_action(
#ifdef __STDC__
uuid__t *vid,
long prod_id,
char *prod_name,
char u_id[32],
char n_id[32],
char g_id[32],
char vrsn[ls_VLEN],
long amt,
trans_id_t trans_id,
char mach_type[32],
ls_job_id_t *job_id,
char multi_use_flag,
long action
#endif
);

void log_db_change(
#ifdef __STDC__
uuid__t *vid,
long action,
char *a1,
char *a2,
char *a3,
char *a4,
char *a5,

```

```

char    *a6,
char    *a7
#endif

);

void log_error(
#ifdef __STDC__
ls_job_id_t *job_id,
uuid__t *vid,
long prod_id,
char *prod_name,
char u_id[32],
char n_id[32],
char g_id[32],
char vrsn[ls_VLEN],
long amt,
long status,
#endif
);

void log_fatal_error(
#ifdef __STDC__
char *msg
#endif
);

void log_lic_to(
#ifdef __STDC__
uuid__t *vid,
long prod_id,
char *prod_name,

```



```

char u_id[32],

char n_id[32],

char g_id[32],

char vrsn[ls_VLEN],

long amt,

trans_id_t trans_id,

ls_job_id_t *job_id

#endif

);

void log_msg(

#ifdef __STDC__

char *vname,

char *msg

#endif

);

void log_svr_crsh();

void log_svr_start();

```

Format of the User File

Below is the format of the file **user_file**. “Creating and Maintaining User Files” on page 3-18 contains an example of a user file.

```

[CRAWL n]
%comment
VENDOR 1stVendorName {ALL | 1stProductName}
{ALLOW | DISALLOW}
1st_user_name [-P1 | -P2]
nth_user_name [-P1 | -P2]
VENDOR 1stVendorName {ALL | nthProductName}
{ALLOW | DISALLOW}
1stUserName [-P1 | -P2]
nthUserName [-P1 | -P2]
VENDOR nthVendorName {ALL | 1stProductName}
{ALLOW | DISALLOW}
1st_user_name [-P1 | -P2]
nth_user_name [-P1 | -P2]

```

The **user_file** resides in the **/usr/lib/netls/conf** directory. You can specify any number of vendors with associated products and user authorizations, and any number of user names

(or none) may follow the **ALLOW** or **DISALLOW** keywords. Keywords may be uppercase or lowercase, but mixed-case keywords are not allowed. Single or double quotes must delimit vendor or product names that contain spaces.

Following are explanations of the keywords in a user file:

- CRAWL** Causes the priority of users to be changed to the next higher priority after the application has polled the queue *n* times. Crawling may be specified only once in a user file. This keyword only applies to products that use queuing.
- VENDOR** Specifies a vendor of licensed products. Either **All** or the name of a single product must follow the declaration of a vendor.
- ALL** Specifies that all of the vendor's products have the same user authorizations. The keyword **ALL** and the name of a single product are mutually exclusive.
- ALLOW** Specifies that the user names following this keyword are allowed to use the products. If no user names follow this keyword, it means no users are allowed to use the products. **ALLOW** and **DISALLOW** are mutually exclusive.
- DISALLOW** Specifies that the user names following this keyword are not allowed to use the products. If no user names follow this keyword, it means all users are allowed to use the products. **ALLOW** and **DISALLOW** are mutually exclusive.
- P1** Specifies a priority-1 user (highest priority). Priority-1 users may displace lower-priority users already waiting in a queue. Default if omitted: priority 3 (lowest priority). This keyword only applies to products that use queuing.
- P2** Specifies a priority-2 user. Priority-2 users may displace priority-3 users already waiting in a queue. Default if omitted: priority 3. This keyword only applies to products that use queuing.

Appendix C. RLM Compatibility

Q1: What is RLM? Does it work with iFOR/LS?

The Resource License Manager is an older network license management program based on an earlier version of iFOR/LS called NetLS. iFOR/LS can service license requests by RLM-enabled client programs in many configurations. This document describes configurations for setting up RLM-enabled clients with iFOR/LS servers.

Q2: What are the recommended steps for setting up RLM and iFOR/LS?

Follow these steps when setting up both RLM and iFOR/LS:

Note: Further detail on these steps can be found in other answers below.

1. Install the iFOR/LS server on a machine that is not running the RLM server; *or* shutdown the RLM server and setup the iFOR/LS server on the same machine.
2. Setup the iFOR/LS server in the default cell.
3. Enter your RLM licenses (keys) and iFOR/LS keys into the iFOR/LS server.

Your iFOR/LS server can now serve both RLM and iFOR/LS clients.

Q3: Can an RLM server and a iFOR/LS server run on the same machine?

No, they cannot. See related Question #7.

Q4: What kind of cell should I join/create when setting up iFOR/LS to work with RLM clients?

You have to be in the default cell because NCS 1.1 (which supported RLM) does not recognize the concept of cells.

Both RLM and iFOR/LS use the Network Computing System to implement client/server communications. Unfortunately, RLM uses NCS 1.1 while iFOR/LS uses a newer version, NCS 1.5.1. The old NCS 1.1 does not support the concept of cells, which can be used to logically partition your network into NCS subnets. RLM keys can work with iFOR/LS, but only if the iFOR/LS server containing the RLM keys is part of the default cell. When an iFOR/LS server joins an alternate cell, or creates a new alternate cell, the RLM client (compiled with NCS 1.1) cannot “see” it.

If a client machine joins an alternate cell, the RLM applications on that client will not be able to get their keys serviced by iFOR/LS.

Q5: Should I run **glbd**, or **nrglbd**? Can multiple **glbds** or **nrglbds** be run?

You can run **glbd** or **nrglbd**, but not both. NCS needs at least one global location broker (**glbd**) or one non-replicable global location broker (**nrglbd**) daemon running in the default cell where the iFOR/LS server resides. It is recommended to run one or more **glbds**.

If you use multiple **glbd**'s, you can use the `/etc/ncs/glb_site.txt` file to prioritize the search for **glbds**, but any client in the cell can see and be serviced by any of the **glbds** running there.

You *cannot* run both **glbd** and **nrglbd** in the same cell. If you try to join the default cell when setting up **glbd** on a server, and a machine in the same cell is already running **nrglbd**, **glbd** will fail to run (it will come up and then terminate within 30 seconds or so). This means that you need to be familiar with every machine in the default cell, since you must insure that they are all running the same type of global location broker. Therefore, if you have multiple

global broker daemons in a network, you must determine the type (**glbd** or **nrglbd**) of each daemon.

Q6: What versions of iFOR/LS work with RLM?

iFOR/LS versions GRI 1.1.2a or above can handle RLM-enabled clients and RLM keys. Previous versions (including the old HP versions) *cannot* handle RLM keys.

Q7: Are the RLM and iFOR/LS servers compatible with each other?

The RLM daemon and the iFOR/LS daemon (**rlmd** and **netlsd**) can coexist on the same network but not on the same machine. The **rlmd** daemon cannot service iFOR/LS keys. Both the RLM server and its clients must be in the default cell.

iFOR/LS Glossary

ADK. The component of the iFOR/LS system that is used by software developers to define and create licenses for software products.

alternate cell (isolated cell). An NCS cell that restricts access to a partial group of nodes in the network. This cell type is usually used to confine iFOR/LS license transactions to a particular set of nodes in the network.

annotation. See *license annotation*.

ARK. Administrator Runtime Kit. The run-time environment for licensed software products. It consists of the network license server daemon (**netlsd**) and its associated tools: **ls_admin**, **ls_rpt**, and **ls_stat**.

compound license. A type of license that allows a system administrator to generate their own license passwords for a given number of licenses. Such a license is valuable when an administrator needs a certain number of licenses, but does not yet know what machines or who will use them. A compound license can generate either nodelocked or non-nodelocked licenses, not both.

concurrent-use license. A type of license administered by the license server that can be used by different users at any node that is connected to a license server node. Concurrent-use licenses allow as many users to use a software product concurrently as there are licenses.

CPU lock. See *nodelocked license*.

default cell (global access). An NCS cell that allows access from any node in the network. This is the most common cell used in iFOR/LS configurations since it allows all other iFOR/LS servers in the default cell to communicate freely.

dynamic license. A license that specifies multiple nodelocked licenses. The license is installed at a license server, and then the license server derives license passwords (each of which specifies a single nodelocked license) from the dynamic license. The server automatically installs the individual licenses at the user nodes from which the licensed product is invoked.

floating license. See *concurrent-use license*.

glbd. The global location broker daemon.

global location broker (GLB). A service that manages the location of all NCS services in the NCS network. The GLB uses one or more daemons (**glbds**) to maintain this service.

iFOR/LS. A run-time license-management application based on Gradient Technologies' Version 2.0.1 (Version 1.1.2a) of the Network Licensing System. The system allows software vendors to bundle compliance mechanisms with their software. In tracking license usage, iFOR/LS allows customers to easily comply with their software license agreements.

iFOR/LS Test Product. The product used by the **ls_tv** tool to verify that license servers are working properly.

license. An instance of permission to use a licensed software product or service. Sometimes, a user needs more than one license to use a product.

license annotation. A special data string that modifies the use of a license in a manner defined by the vendor of the software product.

license database. The database of licenses maintained by a license server. The license database file—**lic_db**—resides in the **/usr/lib/netls/conf** directory.

license information. The information that describes licenses. This information consists of the product name, the product version, the number of licenses, the license type, the start and end dates of the licenses, the target type, the target ID, and a time stamp.

license password. A string encoded with license information for a software product.

license server daemon. A software program that administers licenses for software products, invoked with the command **netlsd**. The **netlsd** command can be found in the **/usr/lib/netls/bin** directory.

licensed product. A software product that has been enabled by a software vendor for use with the iFOR/LS system. Enablement allows a vendor to enforce end-user compliance to their license agreement.

llbd. The local location broker daemon.

local location broker (LLB). A service that provides an interface to the global location broker from the iFOR/LS server. The LLB daemon (**llbd**) has no information about network-wide services. It runs continuously in the background to intercept and forward information to the **glbd**. See also *global location broker*.

log file. The text file that records messages and errors from the license server, and sometimes from licensed products as well, resides in the **/usr/lib/netls/conf** directory.

ls_admin. The software program used to modify a license server database, invoked with the command **ls_admin**, which is located in the **/usr/lib/netls/bin** directory.

ls_rpt. The software program that reports on the history of license server events, invoked with the command **ls_rpt**, which is located in the **/usr/lib/netls/bin** directory.

ls_stat. The software program that reports on the status of licenses, invoked with the command **ls_stat**, located in the **/usr/lib/netls/bin** directory.

ls_tv. The network license server daemon test and verification tool, invoked with the command **ls_tv**, located in the **/usr/lib/netls/bin** directory.

NCS. The underlying communications protocol used by iFOR/LS to transmit licensing transactions between clients and servers. Messages are broadcast from clients to the NCS-managed network.

NCS cell. A logical concept of grouping together one or more machines in an NCS network. Any node belonging to an alternate cell may only have their license requests satisfied by iFOR/LS servers in that cell. Nodes outside the cell may not make license requests to servers in another cell. Two types of cells, default and alternate, are used to provide two ways of accessing iFOR/LS servers.

NetLS. See *iFOR/LS*.

netlsd. The command used to invoke the network license server daemon.

Network License System. See *iFOR/LS*.

nodelocked license. A type of license locked to a specific node so that the product may only be used at that node. The license server does not administer nodelocked licenses.

nodelock file. The text file at a user node (rather than at a license server node) where nodelocked

licenses are added. The **nodelock** file is located in the **/usr/lib/netls/conf** directory.

password. A string encoded with information about a software vendor (vendor password) or about a software product (product password).

product ID. An integer that identifies a vendor's licensed software product; by means of product IDs, the license server distinguishes among products of the same vendor.

product password. A string encoded with information about licenses for a software product. Product passwords are of two types: license passwords and compound passwords.

revision text. See *version identifier*.

target. The node at which a password is to be installed. If the password specifies a single nodelocked license, the target is the node licensed to run the product. If the password specifies multiple nodelocked licenses (that is, a compound password for nodelocked licenses), or licenses of any other type, then the target is a node running the license server daemon.

timestamp. An integer that describes the date and time at which a set of licenses was created.

universal unique identifier (UUID). A string used to identify an NCS cell. The string, generated by the **uuid_gen** tool, is completely unique, having been created based on the unique system ID of the workstation and a time stamp. Once generated, the string is placed in the **glb_obj.txt** file.

use-once license. A type of license administered by the license server that can be used for a single instance of invoking a product or of using a service. The license server decrements the number of use-once licenses each time the product is used.

user file. A text file that specifies the users who may (or may not) use licensed software products.

vendor ID. The identifier of a vendor of licensed products. By means of vendor IDs, license servers can distinguish among any number of vendors established in a network. Vendor IDs are an iFOR/LS-specific usage of Network Computing System universal unique identifiers (UUIDs).

vendor password. A string encoded with information about a vendor that, together with a vendor ID, establishes the vendor of a licensed product in a license database.

version identifier. A string that identifies a version of a product; by means of version identifiers, the license server distinguishes among different versions of a product.

INDEX

Symbols

#, in glb_site.txt file, B-2
-P1 keyword, B-9
-P2 keyword, B-9

A

add, command in lb_admin, A-9
adding a license, to a nodelock file, avoiding errors
 when, 3-3
ADK (Application Developer's Toolkit), 1-1
administrative considerations, vendor specific, 1-3
Administrative Runtime Kit (ARK). *See* license
 server
all keyword, 3-19, 3-20, B-9
allow keyword, B-9
alternate cell
 installing a global location broker daemon in,
 5-7
 when to use, 5-5
annotation, license, 1-4
Application Developer's Toolkit (ADK), 1-1
ARK log file, B-3

C

cell, 1-7
 alternate. *See* alternate cell
 default. *See* default cell
 establishing new, 5-5
 fixup for, 4-6
 manual fixup for, 4-6
clean, command in lb_admin, A-9
cleaning up location broker databases, 4-7
commands, license server administration, A-1
common problems and solutions, 4-8
communications failures, NCS, 4-5
compatibility, with RLM and NCS Version 1.1, 2-9
components of iFOR/LS, 1-5
concurrent-use licenses
 adding or deleting, 3-6
 definition of, 1-3
 troubleshooting, 4-2
 using ls_admin to add, 3-8
configuration, problems relating to, 4-11
configuring iFOR/LS, post-installation configuration.
 See netls_config shell script
configuring NCS, for concurrent-use licenses, 2-2
configuring the glb_site.txt file, 5-3
NCS (Network Computing System), 5-1–5-9
 location brokers, 5-2
 Network Computing Architecture, 5-1
 Network Computing Kernel (NCK), 5-1

remote procedure call run-time library. *See*
 RPC
RPC (remote procedure call), run-time library,
 5-2
configuring the network computing system,
 manually, 5-3
crawl keyword, 3-19, B-9

D

daemons
 glbd, A-6
 llbd, A-14
 maintaining, 3-18
 monitord, A-36
 netlsd, A-37
 nrglbd, A-39
default cell, installing global location broker
 daemon in, 5-4
delete, command in lb_admin, A-10
deleting stale log entries, 3-18
diagnostics, running on iFOR/LS license server,
 2-7
directory structure, 2-7
disallow keyword, 3-20, B-9
distributed computing with NCS, 1-6
drm_admin command, A-2
drm_admin utility, 3-18
dynamic nodelocked licenses, 3-2
dynamic nodelocking, 3-3

E

error logging, 2-7
error messages
 monitord, 4-13
 netlsd, 4-17
errors, ls_tv, 4-4
establishing a new NCS cell, 5-5
events, choosing logged, 3-17
example
 of compiler configuration, 2-10
 of installation, 2-1
 of ls_admin session, 3-8
 of ls_stat session, 3-14
 of user file, 3-18
 of vendor concurrent-use license, 3-7
 of vendor license, 3-4

F

file
 ARK log, B-3
 glb_obj.txt, B-1

- glb_site.txt, B-2
- recording history of target_ids, 2-7
- user, format of, B-8

floating license. *See* concurrent use license

G

- glb_obj.txt file, B-1
- glb_site.txt file, B-2
- glbd (global location broker daemon)
 - description of, A-6
 - installing of in alternate cell, 5-7
 - installing of in default cell, 5-4
 - nonreplicable, 2-3
 - planning for, 2-2

- global location broker
 - administrative tools for, A-9
 - list of possible sites for, B-2
 - locating without broadcasting, B-2
 - object UUIDs of, B-1

H

- handling NCS for iFOR/LS, 4-5

hints

- on adding a nodelocked license, 3-3
- on handling NCS for iFOR/LS, 4-5

I

- ID, vendor, X-2

- iFOR/LS (information for Operation Retrieval/License System)

- components of, 1-5
- installing and configuring, 2-1
- interaction with NCS, 1-7
- relation to NetLS, 1-1
- setting up license server for. *See* netlsd
- upgrading to from NetLS, 2-9
- walkthrough of, for non-nodelocked licensed application, 1-5

- iFOR/LS license server. *See* license server
- installation considerations, with RLM and NCS Version 1.1, 2-9

installation examples

- an existing network, 2-10
- compiler configuration, 2-10
- initial installation, 2-10

- installation procedure, 2-5

- installing and configuring iFOR/LS, 2-1

installing glbd

- in alternate cell, 5-7
- in default cell, 5-4

- invoking licensed products, non-nodelocked, 1-5

K

keywords

- P 2, B-9
- P1, B-9
- all, 3-19, 3-20, B-9
- allow, B-9
- crawl, 3-19, B-9

- disallow, 3-20, B-9
- vendor, B-9

L

- lb_admin

- add command, A-9
- clean command, A-9
- delete command, A-10
- help command, A-10
- lookup command, A-10
- quit command, A-10
- register command, A-10
- set_broker command, A-11
- set_timeout command, A-11
- unregister command, A-11
- use_broker command, A-11
- command description of, A-9

- lb_find command, A-12

- license database, 1-5

license server

- adding and deleting nodelocked licenses, 3-2
- administrative commands, A-1
- choosing a, 2-3
- cleaning up after stopping, 4-7
- display list, A-19, A-33
- getting information about, A-19
- ls_admin operations on, A-19
- managing, 3-11
- reports using ls_rpt. *See* ls_rpt
- setting up, 2-1, 5-8
- starting and stopping properly, 2-8
- typical installation of, 1-1
- update list, A-19, A-33
- verifying operation of, 2-7

licenses

- adding, A-20
- administering
 - using commands, 3-2
 - using SMIT, 3-1
- annotation of, 1-4
- concurrent use
 - adding or deleting, 3-6
 - troubleshooting, 4-2
- deleting, A-20
- enforcement of, 1-4
- getting information on, A-20, A-32
- in use, checking, 3-16, A-32
- installed, checking, A-32
- list of, displaying, A-19
- ls_admin operations on, A-20
- nodelocked
 - adding or deleting, 3-2
 - installing dynamic, 3-3
 - troubleshooting, 3-5
- status information, how to get, 3-12
- troubleshooting
 - concurrent use, 4-2
 - nodelocked, 4-1
 - use once, 4-2

- types of, 1-1
- use once, 1-3
 - adding or deleting, 3-6
 - troubleshooting, 4-2
- llbd (local location broker daemon), 2-2
 - description of, A-14
- location broker, 5-2
 - administrative tools for, A-9
 - lookup of, A-10
 - registering, via lb_admin, A-10
 - unregistering, via lb_admin, A-11
- location broker databases, cleaning up, 4-7
- log entries, stale, 3-18
- log file, ARK, B-3
- logged events, choosing, 3-17
- lookup, command in lb_admin, A-10
- ls_admin
 - Add licenses button, A-20
 - Add product, 3-10
 - Add product button, A-19
 - Add vendor, use of, 3-9
 - Add vendor button, A-19
 - Check user file button, A-19
 - command description for, A-16
 - command line interface, A-16
 - Copy vendor button, 3-11, A-20
 - Delete button, A-20
 - Describe button, 3-8, A-19, A-20
 - Exit button, A-18
 - Operate On menu, A-19
 - Operations, Menu, A-19
 - Product button, A-19
 - Rename button, A-19, A-20
 - sample session, 3-8
 - Server button, A-19
 - startup display, example of, 3-8
 - Update server list button, A-19
 - Vendor button, 3-9, 3-11, A-19
- ls_dpass command, A-21
- ls_rpt, command description of, A-29
- ls_stat
 - All Products (Update) button, 3-14, A-33
 - All selection, 3-15
 - All Servers (Update) button, 3-13, A-33
 - command description for, A-31
 - Exit button, 3-13, A-24, A-32
 - graphical interface, description, 3-12
 - Installed button, 3-13, A-32
 - License Information menu, 3-13, A-32
 - Menus and Buttons, A-24, A-32
 - Product List box, 3-13, A-33
 - sample session of, 3-14
 - Server List box, 3-13, A-33
 - startup display, 3-12
 - Status Message field, 3-14, A-33
 - Usage button, 3-15
 - use of, 3-12
 - Users button, 3-13, A-32
- ls_targetid tool, for obtaining nodelocked license, 1-2

- ls_tv command, A-34
- ls_tv errors, 4-4

M

- managing iFOR/LS, 3-1
- managing license servers, 3-11
- menus and buttons, ls_admin operations on, A-18
- monitord daemon, A-36

N

- NCS (network computing system)
 - See also* distributed computing with NCS
 - basic terminology for, 1-7
 - cell fixup for, 4-6
 - configuring for concurrent-use licenses, 2-2
 - configuring of manually, 5-3
 - distributed computing with, 1-6
 - installation considerations with, 2-9
 - interaction with iFOR/LS, 1-7
 - manual cell fixup for, 4-6
 - problems and solutions, 4-5
- NCS daemons
 - glbd, A-6
 - llbd, A-14
 - nrglbd, A-39
- NetLS (network license system)
 - relation to iFOR/LS, 1-1
 - upgrading to iFOR/LS, 2-9
- netls_config shell script, 2-5
 - and GLB database, 2-5
 - inquiries sent by, when GLB database does not exist, 2-6
 - report generated by, 2-6
- netls_first_time shell script, 2-5
- netlsd
 - description of, A-37
 - enabling, 5-8
- Network Computing System. *See* NCS (Network Computing System)
- nodelock file, adding a license to, 3-3
- nodelocked licenses
 - adding or deleting, 3-2
 - administering, 3-1
 - dynamic. *See* dynamic licenses
 - file for, 1-2
 - obtaining, 1-2–1-4
 - single license, 3-2
 - troubleshooting, 3-5, 4-1
- nodelocking, dynamic, 3-3
- nrglbd
 - description of, A-39
 - implications of, 2-3

O

- object UUIDs, of global location broker, B-1

P

- password, 1-4
 - for nodelocked licenses, 3-2

- use of, for dynamic nodelocked licenses, 3-2
- planar ID. *See* target_id
- platforms, on which iFOR/LS runs, 1-8
- post-installation configuration
 - when a GLB daemon exists, 2-5
 - when no cells exist, 2-6
 - when no GLB database exists, 2-6
- problems, relating to configurations, 4-11
- products
 - adding, A-19
 - displaying list of, A-19, A-33
 - getting information about, A-24, A-32
 - ls_admin operations on, A-20
 - renaming, A-20

Q

quit, command in lb_admin, A-10

R

register, command in lb_admin, A-10

Resource License Manager (See RLM), 2-9

RLM (resource license manager)

- compatibility with iFOR/LS, C-1
- installation considerations with, 2-9

S

scenario

- for initial installation, 2-10
- for installing in existing network, 2-10

set_broker, command in lb_admin, A-11

set_timeout, command in lb_admin, A-11

shell script

- netls_config, 2-5
- netls_first_time, 2-5

shutting down a license server, cleaning up after, 4-7

single-license nodelocked license, 3-2

SMIT, use of for iFOR/LS, 3-1

stale log entries, deleting, 3-18

starting and stopping the iFOR/LS server properly, 2-8

stopping a license server, cleaning up after, 4-7

T

target_id, effect on when planar ID is changed, 2-7

test and verification utility, errors generated by. *See* ls_tv utility

timeout setting in lb_admin, A-11

troubleshooting iFOR/LS, 4-1

U

unregister, command in lb_admin, A-11

use-once licenses

- adding or deleting, 3-6
- description of, 1-3
- troubleshooting, 4-2

use_broker, command in lb_admin, A-11

user files

- P 1 keyword, B-9
- P 2 keyword, B-9
- all keyword, B-9
- allow keyword, B-9
- crawl keyword, B-9
- creating and maintaining, 3-18
- disallow keyword, B-9
- format of, B-8
- maintaining, 3-20
- vendor keyword, B-9
- verifying format of, A-19
- writing and integrating, 3-20

V

vendor ID, X-2

vendor key, 1-4

vendor keyword, B-9

vendor license, nodelocked, example of, 3-4

vendor-specific administrative considerations, 1-3

vendors

- adding, A-19
- copying, A-20
- deleting, A-20
- displaying list of, A-19
- getting information about, A-20
- ls_admin operations on, A-19
- renaming, A-19

verifying operation, of license server, 2-7

