SC23-2666-00

AIX Version 4.1

iFOR/LS Tips and Techniques

Printed in the U.S.A.

First Edition (August 1994)

Trademarks and Acknowledgements

Contents

About This Book

AIX Version 4.1 iFOR/LS Tips and Techniques provides information on planning, designing, and implementing the iFOR/LS licensing system in both new and existing system environments. Extensive information is provided on the interaction of the Network Computing System (NCS) with iFOR/LS to help system administrators and technical-support personnel in understanding and identifying problems related to the licensing system, as well as determining what needs to be done to enable an existing system environment for iFOR/LS. For this purpose, *AIX Version 4.1 iFOR/LS Tips and Techniques* provides an in-depth discussion of various iFOR/LS configurations, along with advanced administration topics.

This book also includes step-by-step procedures to help migrate system environments that use Resource License Manager (RLM) to iFOR/LS.

The tips and samples included in this book apply, for the most part, to both AIX Version 4.1 and AIX Version 3.2. Any version-specific differences are explicit marked.

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

This book is intended to help plan and implement iFOR/LS in new system environments, to enable existing environments to use iFOR/LS, and to help system administrators perform iFOR/LS system administration tasks.

This book is also intended to help technical support personnel perform problem determination and debugging tasks on system environments that use iFOR/LS as the licensing system.

Readers of this book are expected to know the AIX Base Operating System, the TCP/IP subsystem, and their related commands.

For the iFOR/LS advanced reader, information about system coexistence, compatibility, and migration will be useful. The iFOR/LS novice reader will find detailed explanation about each step performed during configuration.

How to Use This Book

Shell scripts are included as guidance on how administration and operation tasks could be automated or tested. These shell scripts are not for general purpose and have not been submitted to any formal test. You will need to adapt the scripts to match your system environment.

Overview of Contents

This book contains the following chapters and appendixes:

- Chapter 1, “iFOR/LS General Concepts,” introduces the concepts, daemons, and files used within the Information For Operation Retrieval/License System (iFOR/LS). This chapter also includes a figure showing the environment used to test the sample configurations discussed in this book. Refer to this figure when reading the samples.
- Chapter 2, “Planning for iFOR/LS,” discusses information that is vital when planning for your system and provides hardware and software requirements.

- Chapter 3, “Login Licensing Model for the AIX Version 4.1. Base Operating System,” discusses the definitions and implementation of the AIX Version 4.1. BOS enablement for iFOR/LS.
- Chapter 4, “Configuring an iFOR/LS Environment,” contains step-by-step instructions for configuring the system for iFOR/LS.
- Chapter 5, “iFOR/LS License Management,” describes how to manage the log data collected by iFOR/LS servers and how to locate important information within log files.
- Chapter 6, “RLM and iFOR/LS Coexistence,” deals with RLM and iFOR/LS compatibility.
- Chapter 7, “NCS Version 1.5.1 vs NCS Version 1.1,” provides information on the differences between NCS Version 1.5.1 and NCS Version 1.1.
- Chapter 8, “iFOR/LS Performance and Availability,” describes the reasons for performance and availability impacts during iFOR/LS usage.
- Chapter 9, “iFOR/LS Security Features,” describes the the security features provided by iFOR/LS.
- Appendix A, “Configuring iFOR/LS within an HACMP Environment,” provides a shell script that may be used instead of the **netls_config** and **netls_first_time** shell scripts to set up iFOR/LS in an HACMP environment. The shell script may need modification for a specific HACMP environment.
- Appendix B, “Testing the NCS Configuration,” provides a shell script that can be used to ensure the definitions in the NCS environment are set up properly.

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.
Italics	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. This book is shipped as part of entitlement with the base operating system.

AIX Version 4.1 iFOR/LS System Management Guide, Order Number SC23-2665, 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).

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 SC23-2666.

Chapter 1. iFOR/LS General Concepts

This chapter introduces the concepts, daemons, and files used within the Information For Operation Retrieval/License System (iFOR/LS) license management mechanism to aid system administrators. It supplements *AIX Version 4.1 iFOR/LS System Management Guide*, SC23-2665.

This chapter also contains a figure, "iFOR/LS Network Example" on page 1-6 that shows the network environment used as reference for all the examples included in this book. Readers may find it useful to have a copy of the figure when reviewing the examples.

iFOR/LS is a software-license management application for flexible network-wide license sharing. iFOR/LS uses the Network Computing System (NCS) distributed computing technology, the industry standard for remote procedure call (RPC), within a client/server model.

iFOR/LS is divided into two parts to support both application developers and end users. Application developers need the Application Developer's Kit (ADK) to enable their applications; system administrators use the Administrator Runtime Kit (ARK) to install and maintain the license server, the location broker system, and the licenses. As only the ARK portion is shipped with AIX, this book only covers the ARK portion of iFOR/LS.

ARK consists of management tools and report tools. The tools provide the capacity to manage licenses in the network, as well as information about software and license usage in the network. The license server, used to maintain and grant licenses, and the license administration tool, used to insert licenses into the license server, are complemented by reporting tools, allowing a network administrator to monitor and report on the usage of applications across the network. These reports can then be used to track software, charge departments based on usage, and justify purchasing additional licenses and machines.

Set Up an iFOR/LS Environment

Following is a list of concepts a system administrator must know when planning to set up or implement an iFOR/LS environment:

- Network Computing System on page 1-1
- Remote Procedure Call on page 1-2
- Location Broker on page 1-2
- Cells on page 1-2
- Universal Unique Identifier on page 1-3
- Objects, Types and Interfaces on page 1-3
- Sockets on page 1-3
- Commonly Used License Types on page 1-3

Network Computing System

Network Computing System (NCS) is an implementation of the Network Computing Architecture, an architecture for distributing software applications across heterogeneous collections of computers, networks, and programming environments. Programs based on NCS can take advantage of computing resources throughout a network or internet, with different parts of each program running on computers best suited for the tasks. For example, one program might perform graphical input and output on a workstation while performing intense computation on a supercomputer.

Remote Procedure Call

Remote Procedure Call (RPC) is a protocol that provides the high-level communications paradigm used in the operating system. RPC presumes the existence of a low-level transport protocol, such as TCP/IP or UDP, for carrying the message data between communicating programs. RPC implements a logical client-to-server communications system designed specifically for the support of network applications.

The RPC runtime library is the backbone of NCS. It provides the calls that enable local programs to run procedures on remote hosts. These calls transfer requests and responses between clients (the programs calling the procedures) and servers (the programs running the procedures).

Location Broker

The location broker enables clients to locate specific objects such as databases or specific interfaces such as data-retrieval interfaces. NCS provides two different kinds of location brokers:

- **Local Location Broker (LLB)**, an RPC server that maintains a database of information about objects and interfaces located on the local host. The LLB provides access to its database for application programs and also provides the location broker forwarding service. An LLB must run for any host that runs RPC servers. The LLB runs as the daemon program, **llbd**.
- **Global Location Broker (GLB)**, an RPC server that maintains information about objects and interfaces throughout the network or internet. There are two versions of the GLB daemon: **glbd** and **nrglbd**. The two versions provide the same functionality to clients, but the **glbd** is replicatable, while the **nrglbd** is not. A replicatable **glbd** maintains a replicated GLB database (several copies of the same database on different hosts). This increases availability of the information database for the case of hardware or network failure. The two GLB versions cannot interoperate on the same network or internet. It is recommended that the **glbd** be used if possible.

Cells

A location broker cell is a subset of a network or internet, which can be serviced by a **glbd** daemon. Cells have disjoint and independent GLB databases.

A network or internet can be logically partitioned into several cells by using the **glb_obj.txt** configuration file, which contains the Universal Unique Identifier (UUID). NCS considers hosts with identical UUIDs in their **glb_obj.txt** file as being part of the same cell. There is no correspondence between cells and topology. Any number of hosts can be from any network or internet in a cell. This means that two hosts physically next to each other can belong to different cells while two hosts in different networks and hundred of miles apart can be part of the same cell.

A workstation can only belong to a single cell.

There are two different kind of cells:

- **Default cell**: uses a default (in NCS hardcoded) identifier for the GLB object. Hosts in the default cell do not need the **glb_obj.txt** configuration file.
- **Alternate cell**: uses the UUID stored in the **glb_obj.txt** configuration file to identify hosts belonging to a specific cell. When a host is configured to belong to an existing alternate cell, the alternate cell’s UUID (delivered by the **lb_find** tool) is stored in the host’s **glb_obj.txt**. When a new alternate cell needs to be created, the **uuid_gen** tool is issued to generate a new UUID.

Universal Unique Identifier

The Universal Unique Identifier (UUID) is a 128-bit value used for identification. NCS uses UUIDs to identify interfaces, objects, and types.

Objects, Types, and Interfaces

An **object** is an entity accessed through well-defined operations. A file, a serial line, a printer, and a processor can all be objects.

Every object has a **type.** Programs can access any object of a given type through one or more **interfaces**. Each interface is a set of operations that can be applied to any of these objects. For example, several printer queues can be classified as objects of one type. Any of these objects can be accessed through a directory interface that includes operations to add, delete, and list jobs in the queues.

An object can be replicated. Replicas are copies of an object that all have the same UUID. The global-location-broker database, for example, can be a replicated object. Replicating an object can ensure the availability of information or services despite hardware or network failure.

Sockets

In NCS, a socket is a port on a specific host; a communications end point that is accessible through a protocol family's addressing mechanism. An RPC server listens on one or more sockets where it receives any messages directed to it.

Each socket is identified uniquely by a socket address. A socket address is a data structure that specifies the following:

Address family
Also called the protocol family. Determines the communications protocol used to deliver messages.

Network address
Given the address family. the network address uniquely identifies a host and contains information sufficient to establish communication with the host.

Port number Specifies a port within a host. The terms *port* and *socket* are synonymous, but *port number* and *socket address* are not. A port number is one of three parts in a socket address.

Commonly Used License Types

The most commonly used licenses are summarized in the following table. (See "Types of iFOR/LS Licenses" in *AIX Version 4.1 iFOR/LS System Management Guide* for a detailed definition of nodelocked, concurrent-use, and use-once licenses. License annotation, passwords and vendor keys, and license databases are also described under this heading.)

LIcense Type	**Uses iFOR/LS Server?**	**Application Access**	**Usage**	**May Be Re-used?**
Nodelocked	No	Limited to ma-chine	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.

Summary of iFOR/LS Daemons

Following is a list of daemons involved within the licensing mechanism implemented by iFOR/LS:

llbd	Implements the NCS local-location-broker daemon.
glbd	Implements the replicatable global-location-broker daemon. This program is located in the directory **/etc/ncs**.
nrglbd	Implements the nonreplicatable global-location-broker daemon. This program is located in the directory **/etc/ncs.**
netlsd	Implements the iFOR/LS license-server daemon. This program is located in the directory **/usr/lib/netls/bin**.
monitord	Acts as common interface between the different, countable AIX login programs (**getty**, **telnetd**, **rlogind**, CDE), and the iFOR/LS daemon **netlsd**. This program is located in the directory **/usr/sbin**.

Summary of iFOR/LS Files

Following is a list of files involved within the licensing mechanism implemented by iFOR/LS:

/usr/lib/netls/conf/lic_db
The license database stores vendor, product, and license information.

/usr/lib/netls/conf/lic_db.bak
A backup database for **lic_db**. If **lic_db** is missing when **netlsd** is started, it automatically uses **lic_db.bak** to create a new **lic_db** database.

/usr/lib/netls/conf/cur_db
This database contains information about the status of concurrent use licenses that are held by users.

/usr/lib/netls/conf/log_file
A log file where the license server daemon **netlsd** keeps a history of license-server events.

/usr/lib/netls/conf/nodelock
An ASCII file used to store nodelock licenses. Nodelock licenses are specific to the target ID of the node and the licensed software product. Since entries in this file are encrypted, it is a good idea to add a comment line to each entry that includes the full product name, the version, and the license expiration date (if any).

/etc/ncs/glb_obj.txt
An ASCII file that allows a network or internet to be partitioned into several

location broker cells, each served by its own **glbd** (the **glbd** may be replicated within a cell). The **glb_obj.txt** file is created only on hosts configured for *alternate* cells and contains the UUID for an specific *alternate* cell. Hosts within the *default* cell do not need this file.

/etc/ncs/glb_site.txt
An ASCII file that contains a list of possible hosts (**ip** names or **ip** addresses) where a **glbd** may be running. This file is useful on networks not supporting broadcasting (which is the usual method of locating the **glbd**).

/etc/ncs/glb_log
An ASCII file containing diagnostic output for the **glbd**.

/etc/ncs/glb.e The GLB database. It contains entries for all objects and interfaces registered by the GLB. The identifying keys in the registration information are the network address of the server host, the object UUID, the type UUID, the interface UUID, and a time stamp.

/etc/ncs/glb.p The database implements the *propagation queue* used by the data replication manager (DRM) within the global location broker to immediately propagate GLB updates to GLB replicas at all other hosts on the replica list. If any entries remain in the propagation queue (for example, due to unavailability of some GLB replicas), the DRM retries the propagation every 15 minutes.

/etc/ncs/uuidname.txt
An ASCII file that associates textual names with UUIDs. Names in this file are used by the **lb_admin** NCS administrative tool to identify objects, types, and interfaces.

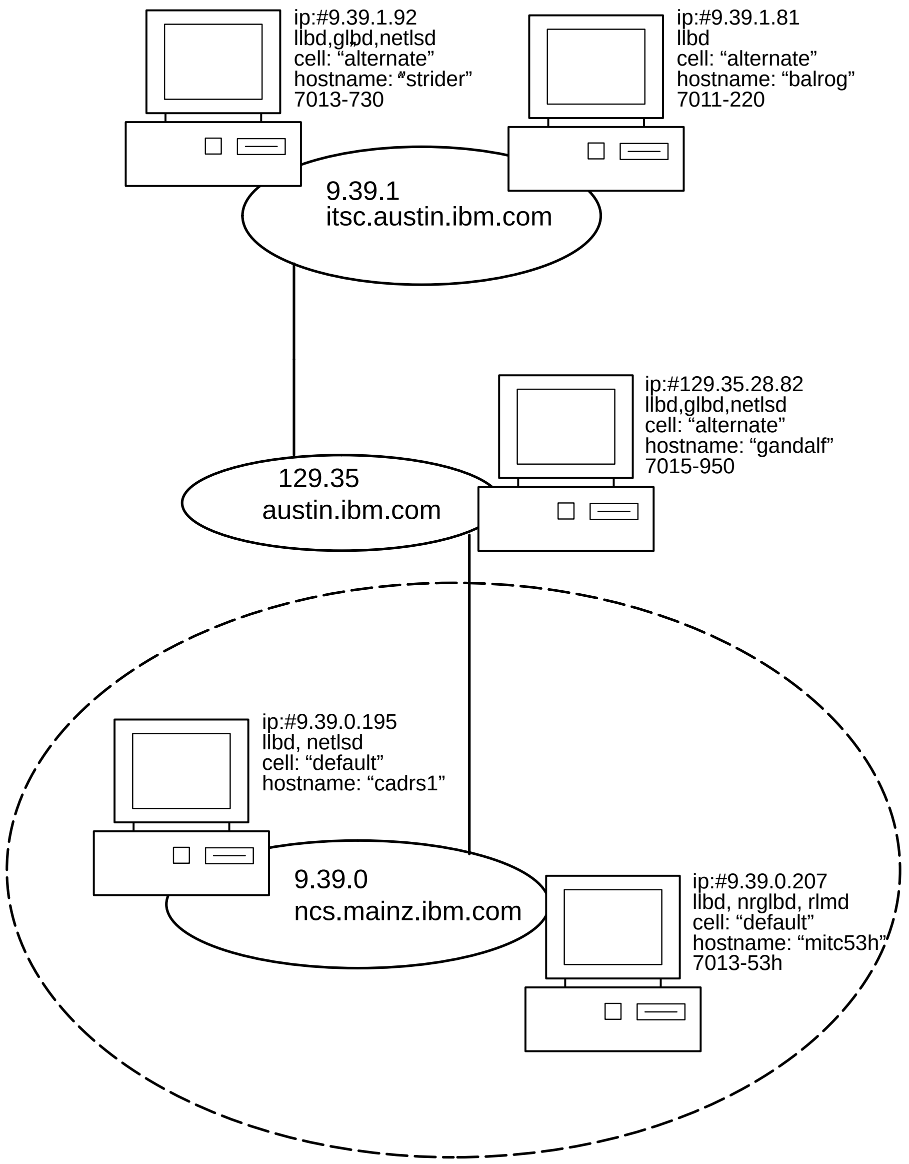
/etc/ncs/uuid_gen
This tool generates a UUID for the cell and puts it into the **glb_obj.txt** file.

/etc/rc.ncs The startup file for NCS daemons.

/etc/rc.netls The startup file for the **netlsd** daemon.

iFOR/LS Network Example

To ensure readability, all examples included in this book correspond to the network environment shown in the figure below. Readers might find it useful to keep a copy alongside when reviewing the examples.



Note: The cell names are symbolic only, the UUIDs are the real descriptors. In this sample environment, the following cells were used:

Default cell UUID	333b91c50000.0d.00.00.87.84.00.00.00
Alternate cell UUID	657cab79f66f.02.81.23.1c.51.00.00.00

Chapter 2. Planning for iFOR/LS

iFOR/LS and NCS provide several ways to satisfy customer needs for a licensing system. Designing the licensing environment that provides the best solution for your business requires careful and thoughtful planning. A well-planned iFOR/LS licensing environment is easier to install and configure and requires less maintenance than a poorly planned one.

It is recommended to include enough time for planning, specially when implementing iFOR/LS in large networks or across subnets. This chapter should help system administrators to be aware which topics are important when planning the licensing system and to find the software options they need for their requirements.

Hardware and Software Requirements for iFOR/LS

The following sections describe the system requirements for installing iFOR/LS.

Hardware Requirements

iFOR/LS is supported on all systems running AIX Version 3.2.3 Extended or higher as the base operating system. The product itself does not have any detailed hardware requirements.

AIX Version 3.2 Software Requirements

- You must have installed the AIX Base Operating System (BOS) Runtime (**bos.obj**). The AIX BOS is part of the AIX licensed program.
- If you want to use the iFOR/LS graphical-user-interface (GUI) administration tools, you must install AIXwindows (**X11rte.obj**) Version 1.2.3, level X11R5, or later.
- You must have installed TCP/IP (**bosnet.tcpip.obj**), included in the operating system.

AIX Version 4.1 Software Requirements

- You must have installed the AIX Base Operating System Runtime (**bos.rte**).
- Install the AIX Windows Runtime Environment (**X11.base**) if the iFOR/LS GUI administration tools should be used.
- The TCP/IP client package (**bos.net.tcp.client**) must be installed. Optionally, you may install TCP/IP SMIT Support (**bos.net.tcp.smit**).

iFOR/LS Filesets: Packaging and Requirements

Starting with AIX Version 4.1, the iFOR/LS-related software is distributed in four filesets contained in two packages. Each of the filesets is separately installable and deinstallable.The filesets are packaged as follows:

1. **bos.ifor_ls** package, including the three following filesets:
   - **bos.ifor_ls.client.rte**
   - **bos.ifor_ls.client.utils**
   - **bos.ifor_ls.server**
2. **bos.net** package, including the following iFOR/LS related fileset:
   - **bos.net.ncs**

The following table shows the filesets and their sizes:

Package	Fileset	Size (MB)
bos.ifor_ls	bos.ifor_ls.client.rte	0.9
	bos.ifor_ls.client.utils	1.2
	bos.ifor_ls.server	0.5
bos.net	bos.net.ncs	1.7
Total		**4.3**

The bos.ifor_ls Package

This package contains the iFOR/LS client and server software in three different filesets:

Client Runtime (bos.ifor_ls.client.rte)

This fileset contains the Network Computing Kernel (NCK) shared libraries (both the NCS V1.1 **libnck.a**, and the NCS Version 1.5.1 **libshrnck.a** libraries) and the iFOR/LS (**libnetls_shr.a**) shared library.

This fileset is required by applications that link with iFOR/LS shared libraries and which request license(s) during execution.

Client Utilities (bos.ifor_ls.client.util*s*)

This fileset includes the iFOR/LS administrative tools and utilities (**ls_admin**, **ls_tv, ls_rpt**, **ls_stat**, and **ls_targetid**), the SMIT screens, and the X11R5 library stubs.

This fileset is required to install and administer vendors, products, and licenses, and to generate license usage reports. An application does not need this fileset during execution.

Server (bos.ifor_ls.server)

This fileset includes the **netls_config** configuration shell script, the iFOR/LS daemon **netlsd**, and the **ls_dpass** tool.

This fileset is needed if the machine is being configured as an iFOR/LS server. The **ls_dpass** tool is used to access and manipulate compound licenses.

The bos.net Package

This package contains many filesets related to the network software including the NCS fileset required by the iFOR/LS server.

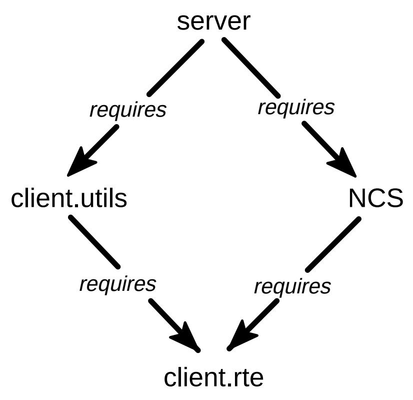
NCS (bos.net.ncs)

This fileset includes the NCS daemons and administrative tools (**llbd**, **glbd**, **nrglbd**, **lb_admin**, **lb_find**, **drm_admin**, **uuid_gen**, and **stcode**).

This fileset is an optional program product (OPP) and must be separate to allow users to install the NCS Version 1.5.1 over old NCS Version 1.1 software. NCS is required on a network which has an iFOR/LS server installed. Although NCS clients do not require this fileset during execution, you may consider installing it to allow for debugging activities.

Package Dependencies

During installation of AIX V4.1 BOS, the fileset **bos.ifor_ls.client.rte** is installed automatically. The following figure shows the dependencies among the different filesets:



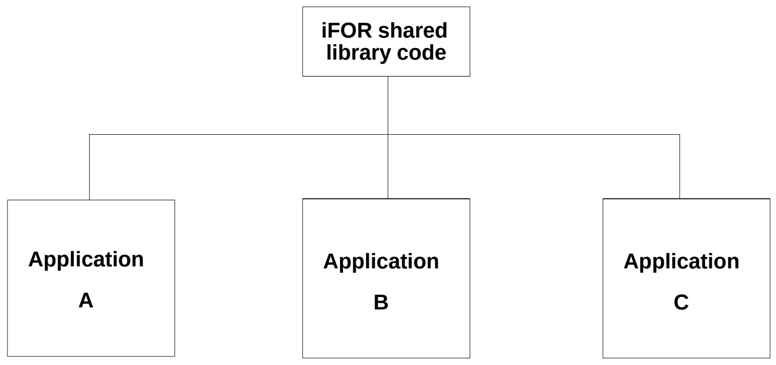
Package Dependencies

iFOR/LS Shared Libraries

Shared libraries allow applications to share commonly used code in memory, thus reducing memory requirements while providing a good model for binary compatibility upgrades.

Applications enabled for iFOR/LS require support provided by the iFOR libraries. This support was provided by statically binding in the library code with the application. With the shared library, multiple applications can use the same copy of this shared code allocated in memory. In this case, the applications must have been compiled using the iFOR shared libraries.

The following figure illustrates this by indicating memory contents for three different simultaneously running applications.



Using Shared Libraries

In this model the first application (A) that uses iFOR library code will load the code into memory. Other applications (B and C) can share that code. Greater memory savings are realized as more applications are run.

Requirements and Sample Scenarios

The following sample scenarios should help administrators to clarify which iFOR/LS filesets they need for their environments.

Case 1: Standalone Machine Running AIX

In this case, no filesets 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 filesets installed and configured.

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

Requires the **client.rte** fileset because CSet++ utilizes concurrent-use licenses provided by the iFOR/LS server on another machine. The **client.utils** fileset can be installed to enable this client for license administration and license-usage reports.

Case 4: Networked Machine Acting As iFOR/LS Server

An iFOR/LS server needs all fou*r* filesets. Refer to the package-dependencies figure on page 2-3.

Case 5: NCS Application Not Using iFOR/LS

The **bos.net.ncs**, and the **client.rte** filesets are required.

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

In this case all four filesets are needed because the **ls_dpass** utility within the *server* fileset is needed.

Selecting iFOR/LS Servers

A license server should run on a computer system that is highly available and reliable and that runs in a controlled environment. Computers that are frequently unavailable, unmanaged, or often shut down for testing or other purposes are not good candidates.

Ideally, license servers should be on the same LAN as the majority of computers that will request licenses. Running license servers over a number of bridges, routers or WAN connections is possible but may lead to inconvenient situations. The license service is as critical to your network as is the Domain Name Service (DNS), the Network Information Service (NIS) or the Network File System (NFS).

Computers that function as license server nodes are also suitable for running the licensed product. The network load of the license service depends on the number of managed licenses and the way a certain product sets up a license heartbeat.

Licenses should be distributed among three or more iFOR/LS license servers. When an iFOR/LS Server system is down, the licenses assigned to this specific license server are unavailable until the recovery of that system. Be aware of this when planning your LAN environment and consider having more than one license server in the NCS cell for distributing the licenses. All license servers in the LAN can be managed from a single point.

Nodelock Licenses vs. Concurrent-Use Licenses

Nodelock licenses restrict an application’s use to a particular machine. Having both the application and the license installed on the same machine eliminates the overhead of a license server, since the nodelock license is entered into the local file system. Nodelock licenses allow unlimited usage but only on one machine. When the software product starts up, it checks the file **/usr/lib/netls/conf/nodelock** on the current node to see if an appropriate entry (nodelock license) exists allowing access. The tradeoff is that if the system is unavailable, the application is also unavailable.

Using the nodelock license application on a different system requires the ordering of new product licenses for this CPU-ID. You can not just copy the nodelock file or mount a remote filesystem containing the nodelock file because the license information is checked against the target ID (typically the CPU-planar ID) of the machine where the product was started. For each system in the network, you have to maintain a different nodelock file.

Concurrent-use licenses are *floating* licenses that allow users to retrieve software licenses over a network. The customer purchases and installs the number of licenses expected to be used at peak periods. The software itself can be installed on any system in the network. Administration is done at a single point for all networked systems.

When a software product is started, it communicates with the license server (**netlsd**) in the network to determine if concurrent licenses for that product have been set up and whether any are available.

iFOR/LS requires any software product using a concurrent-use license to periodically verify to the **netlsd** that it is still alive. This prevents products that have halted abnormally from holding licenses they are no longer using. See "monitord Heartbeat" on page 3-5.

It is possible to use the nodelock and network floating licenses mechanisms together on one machine. Plan your environment carefully according to your needs before ordering any licenses.

iFOR/LS Planning Tips

It is strongly recommended that enough time be allowed for planning the implemention of an iFOR/LS environment. The most critical topics to consider are:

- Availability
- Serviceablity
- Maintainability
- Reliability
- License fees
- Type of license

1. Determine the computer or group of computers that use iFOR/LS licensed applications. Based on this, partition the network into *license cells* where mainly the same licensed applications are used.

   Note: The term *license cells* is not a defined name; it is used here only to illustrate how to group machines for the network layout. Machines, installed products, and products used within a license cell influence each other in terms of licensing.

2. Nodelock licenses should be preferred for applications which typically run on dedicated (single-use) workstations.
3. License cells that have computers shared by several users who use different applications should run with floating licenses.
4. Sensitive applications running as daemons or in batch queueing mode should use the floating license method.
5. Calculate the amount of floating and nodelock licenses needed within a license cell.

6. Based on the license cells, you should try to set up different NCS cells.
7. Each NCS cell must be administered by a cell administrator who needs root access.
8. At least one iFOR/LS server is needed within a NCS cell. To avoid losing all licenses in a cell when the iFOR/LS server is unavailable, plan at least two iFOR/LS servers in a cell, and split the licenses between them.
9. Integration of license accounting is easier if NCS cells (for example, on a departmental level) are used.
10. The final number of NCS cells to be set up on the network is also influenced by the following topics:
    - Total amount of computers
    - Amount of computers within each NCS cell
    - Amount of iFOR/LS licensed products used
    - Designated system/cell administrators
    - Licensing accounting.
11. Set up a change management plan for iFOR/LS servers. Changes made to license servers should be implemented in a managed fashion to avoid server unavailability due to user handling errors.
12. Set up a recovery management plan to restore normal license services to the user in the event of system or disk crash. Refer to "iFOR/LS Backup and Recovery Examples" on page 5-18.
13. Determine a place where copies of the *license forms* obtained from the license provider will be kept. This may be crucial for fast recovery in the case that the license database gets corrupted and a backup is not available.
14. Use a planning sheet like the one showed in the following section.

iFOR/LS Server Planning Sheet

The following planning sheet may be used to collect all information needed before configuring the iFOR/LS environment. The values included are extracted from the network environment shown in “Network Example” on page 1-6.

Hostname	IP-Address	glbd	netlsd	glb_site.txt	cell-uuid (glb_obj.txt)
mitc53h	9.39.0.207	y	n	ip:#9.39.0.207 ip:#9.39.0.195	333b91c50000.0d.00.00.87.84.00.00.00
cadrs1	9.39.0.195	n	y	ip:#9.39.0.207	333b91c50000.0d.00.00.87.84.00.00.00
strider	9.39.1.92	y	y	ip:#9.39.1.92 ip:#129.35.28.82	657cab79f66f.02.81.23.1c.51.00.00.00
balrog	9.39.1.81	n	n	ip:#129.35.28.82	657cab79f66f.02.81.23.1c.51.00.00.00
gandalf	129.35.28.82	y	y	ip:#129.35.28.82 ip:#9.39.1.92	657cab79f66f.02.81.23.1c.51.00.00.00

Chapter 3. Login Licensing Model for the AIX Version 4.1 Base Operating System

Every countable nonroot login to an AIX Version 4.1 system requires a user license.The system administrator configures the number of user licenses available on a system. By default, two user licenses are configured. This chapter discusses the definitions and implementation of login licensing in the AIX Version 4.1 Base Operating System.

Countable and Non-Counted Logins

There are several ways to access the AIX Version 4.1 BOS system. The following are ways that a user can access the system that require an AIX Version 4.1 user license:

- Logins provided via a **getty** (from an active, local terminal)
- Logins provided using the **rlogin** or **rsh –l** command
- Logins provided using the **telnet** or **tn** command
- Logins provided through the Common Desktop Environment (visual login CDE)

Any other way of accessing the AIX Version 4.1 BOS system does not require AIX user licenses (for example: **ftp**, **rexec**, **rsh** without the **–l** flag).

AIX Version 4.1 BOS Licenses

The following AIX Version 4.1 BOS licenses can be obtained:

- Default two-user licenses distributed with the software
- Additional fixed login licenses in increments of one for a designated machine (referred to as fixed licenses because they are available only on a specific machine).
- Additional floating user licenses. Floating licenses are licenses that are available to any user on the network where the licenses are installed.

Default Two-User License

The AIX Version 4.1 Base Operating System software is distributed with a two-user license. Thus, by default, two nonroot users are allowed to login to the system. The root user does not consume any AIX Version 4.1 BOS licenses and is allowed unlimited logins.

Additional AIX Version 4.1 BOS Fixed Licenses

You can configure additional fixed login licenses in increments of one for a designated machine. The number of fixed licenses can vary from 2-32767.

The **chlicense** command or SMIT can be used to configure the system for these additional licenses.

Additional AIX Version 4.1 BOS Floating Licenses

You can configure additional floating concurrent-use login licenses in increments of one with no limit on the maximum value. These licenses are not restricted to a designated machine, and the right to use them is allowed to float in a network. The floating license option can be configured in addition to any fixed licenses already configured.

Floating licenses are accessed by a request to a license server. Each license is used to satisfy a unique request on a network, where the license count is not enforced for an individual machine, but instead for an NCS cell on the network. Refer to “Cells” on page 1-2 and to “Configuring NCS Cells” on page 4-9 for more information about NCS cells.

The **chlicense** command or SMIT can be used to enable the system for these additional floating licenses.

Software Requirements for AIX Version 4.1 BOS Licenses

When AIX Version 4.1 BOS is installed, the following iFOR/LS-related package is automatically installed:

- **bos.ifor_ls.client**

The iFOR/LS software is not required to use the default two-user licenses.

File Sets Required for Changing the Number of Licenses on a System

The **bos.sysmgt.loginlic** file set contains the following commands for determining what licenses are available on an AIX system:

chlicense	Sets the number of fixed licenses on an AIX system and enables and disables floating licensing for the system.
lslicense	Shows the number of fixed licenses on an AIX system and whether floating licensing is enabled or disabled.
monitord	Daemon process that interfaces between login processes and a license server to handle floating user license transactions.

The **bos.ifor_ls.server** file set contains the license server that is needed to serve floating user licenses to clients on the network. It also contains the SMIT screens needed to create, install, and update the floating user licenses managed by the server.

The **bos.ifor_ls.server** file set also requires the **bos.net.ncs** file set.

Login Authentication Overview

The authentication mechanism involves a semaphore setup using the **maxlogin** value found in the file **/etc/security/login.cfg**. The semaphore is decremented by each successful login, and at zero, no more logins are allowed. The semaphore is set by using a reset flag, such that whenever a login exits, the count is restored.

When all the semaphores are used up by the logins, at the next login the authentication code checks whether floating licensing is enabled or disabled. If floating licensing is enabled and licenses are available in the network, the login is allowed to continue.

License Validation

AIX Version 4.1 allows two modes for acquiring login licenses:

Fixed License Mode
is selected if the user is installing licenses on a standalone machine. This mode does not use iFOR/LS to track logins.

Float License Mode
is selected for using AIX Version 4.1 BOS concurrent-use licenses floating in the network. This mode utilizes iFOR/LS to satisfy login license requests.

The login license mode can be set with the **chlicense** command. The **lslicense** command displays the current mode and the current maximum number of fixed licenses.

Setting the License Type from the Command Line

The **chlicense** command can be used to set the number of fixed user licenses for a machine. To set the number of fixed users to 25 by using the **chlicense** command, enter:

```
chlicense –u 25
```

The **chlicense** command also has an option to toggle between fixed and floating mode:

```
chlicense –f on|off
```

chlicense –f on performs the following steps:

- Creates the following entry in **/etc/inittab**:
  ```
  monitord:2:once:/usr/sbin/monitord >/dev/console 2>&1
  ```
- Starts the **monitord** daemon explained later in this chapter.
- **monitord** creates the empty file **/etc/security/monitord_lock.**

chlicense –f off performs the following steps:

- Deletes the **monitord** entry in **/etc/inittab**.

To view the current license configuration for a machine, the **lslicense** command is used. The **lslicense** command displays the current licensing mode and maximum number of fixed licenses. For example:

```
root@inti# lslicense
Maximum number of fixed licenses is 7.
Floating licensing disabled.

root@inti# chlicense –on

root@inti# lslicense
Maximum number of fixed licenses is 7.
Floating licensing enabled.
```

The number 7 means two default licenses plus five additional fixed-user licenses.

Setting the License Type by Using SMIT

The number of fixed user licenses for an AIX system can be set by using the **chlicense** command with the **–u** flag followed by an integer that specifies the number of licenses purchased for the system. The number of fixed user licenses can also be set by means of the following SMIT menus:

```
System Management
  System Environments
    Change / Show Number of Licensed Users
```

Floating licensing can be enabled and disabled for an AIX system by using the **chlicense** command with the **–f** flag followed by the word **on** or **off.** Floating licensing can also be enabled and disabled by means of the following SMIT menus:

```
System Management
  System Environments
    Change / Show Number of Licensed Users
```

The licensing configuration for an AIX system can be inspected by using the **lslicense** command with no arguments or by means of the following SMIT menus:

```
System Management
  System Environments
    Change / Show Number of Licensed Users
```

Installing Floating User Licenses on an AIX System

The following steps should be used to install floating user licenses on an AIX system. This procedure can only be used to install licenses. To change the number of licenses installed on the server, follow the procedure “Changing the Number of Floating User Licenses on an AIX System.”

1. Install the **bos.ifor_ls.server** file set.
2. Configure the machine as a network server by running the script **/usr/lib/netls/conf/netls_config.**
3. Install AIX floating user licenses by means of the following SMIT menus:

   ```
   System Management
         System Environments
           Manage AIX Floating User Licenses for This Server
             Register Floating User Licenses
   ```

4. Specify the number of floating user licenses purchased for installation on the server. The licenses will be automatically installed.

Changing the Number of Floating User Licenses on an AIX System

The following steps should be used to change the number of floating user licenses that are installed on an AIX system. This procedure can only be used on a system that already has floating user licenses installed. To initially install floating user licenses, follow the procedure “Installing Floating User Licenses on an AIX System.”

1. Change the number of AIX floating user licenses installed on a system by means of the following SMIT menus:

   ```
   System Management
     System Environments
       Manage AIX Floating User Licenses for This Server
         Change the Number of Installed Floating User Licenses
   ```

2. Select the installed license to change.
3. Specify the new number of floating user licenses to be installed on the system.
4. The old license will be replaced with a new license that has been updated with the new number of users specified.

iFOR/LS Policies Implemented

When AIX Version 4.1 BOS floating licenses are used, the following iFOR/LS policies apply:

- On license request failure: *Hard stop*. If an iFOR/LS license request cannot be satisfied, the login will fail.
- On checkperiod (heartbeat) failure: *Soft stop*. If a checkperiod (heartbeat) for an iFOR/LS license fails, then the login session is allowed to continue.
- The checkperiod (heartbeat) for AIX Version 4.1 BOS floating licenses is 15 minutes by default and is configurable when the **monitord** daemon is started.
- The root user is always able to log in (with password) without a license.

In floating-license mode, the system tracks users by first assigning all of the fixed mode licenses available, then going to **monitord** and the iFOR/LS system for additional licenses if needed.

The monitord Daemon

The AIX Version 4.1 BOS provides 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 license server **(netlsd)**. The **monitord** daemon communicates with the iFOR/LS server and requests an AIX Version 4.1 BOS concurrent-use license for each countable login**.** Refer to “Countable and Non-Counted Logins” on page 3-1 for more information on countable logins.

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 BOS license the user was using in order to make it available for further logins.

The **monitord** daemon keeps track of each login process on the system that is granted a floating user license. **monitord** holds a license for each of these processes and performs ”heartbeats” to keep the licenses. A heartbeat is a communications mechanism that **monitord** uses to tell the iFOR/LS server that a floating license is still being used. If a heartbeat is missed, the license server will assume that the license is no longer in use, and the license is then made available for consumption by another user on the network. The process that loses a license in this manner is allowed to continue execution, but it is no longer a licensed user login.

When a user logs out of a system, **monitord** is told to release a license. If the user that exited had obtained a floating user license, **monitord** returns the license to the license server. If the user that exited had obtained a fixed user license, **monitord** takes no action.

monitord Heartbeat

When a license is checked out from an iFOR/LS license server, a time interval must be specified that defines the time-out period for the license. The time-out period is the maximum amount of time that the license server will wait for usage verification before the license is invalidated and made available again to other users.

monitord requests licenses from a server on behalf of user login processes. Therefore, it is also the responsibility of **monitord** to perform the heartbeats necessary to keep these licenses valid. The heartbeats issued by **monitord** occur at regular intervals based on when the first floating user license is obtained by **monitord**. For example, suppose the first floating user license obtained for a system is checked out from the license server at exactly 12:00 noon, and the heartbeat interval is set to 15 minutes. The following table illustrates when heartbeats will occur for various floating user login licenses obtained for the same machine.

License Number	License Checkout Time	Heartbeat Time
1	12:00pm	12:15pm 12:30pm 12:45pm
2	12:01pm	12:15pm 12:30pm 12:45pm
3	12:10pm	12:15pm 12:30pm 12:45pm
4	12:20pm	12:30pm 12:45pm 1:00pm
5	12:27pm	12:30pm 12:45pm 1:00pm

Note: The heartbeat is actually performed 10 seconds prior to the time-out expiration to allow for network delays.

Heartbeats continue for a license until the login process that obtained the license exits or until communications fail between **monitord** and the license server (invalidating the license).

If **monitord** terminates abnormally and does not release the floating user login licenses that it holds, the licenses will become available again to the network after the time-out period expires.

The default time-out/heartbeat period for monitord is 15 minutes. This value is configurable and can be set by using the **–t** option when starting **monitord.** When choosing a heartbeat interval for **monitord,** you should consider the tradeoff between the following:

1. License Recovery Time
2. Network Traffic

License recovery time (the time-out period) is the amount of time that must pass before a license is made available for re-use in the event of an application failure. In the context of **monitord**, this is the maximum amount of time that a license may be unusable on the network if **monitord** is abnormally terminated. Short time-out periods allow for quick recovery of licenses in the event of **monitord** or communications failure.

Network traffic due to **monitord** heartbeats increases as the number of floating licenses in use on a system increases. If the time-out period used by **monitord** is short, then network traffic will be heavier as all of the heartbeats are performed more frequently for the system.

monitord Syntax

The syntax for the **monitord** daemon is:

```
monitord [–t n]
```

where the **–t** option is used at daemon startup. The default (invoked without the **–t** option) is an interval of 15 minutes. This flag is used to set the value of the *heartbeat* interval:

–t 0	gives an infinite interval
–t *n*	gives *n* minutes

The **–t** flag is used with **monitord** to set the heartbeat interval for communications with the license server. An integer value must be specified with the **–t** flag to define the number of minutes between heartbeats. This interval is also the time-out period for license recovery. If the specified interval is 0, then then the time-out/heartbeat interval will be set to INFINITE; heartbeats will never occur, and floating user licenses will never time out automatically. If the **–t** flag is used, the specified value is written to the file **/etc/security/monitord.cfg**. If the **–t** flag is not used and the **monitord.cfg** file exists and specifies a valid value, then that value is used for the heartbeat/time-out period. If the **–t** flag is not used, and no preconfigured value can be retrieved, then the default value of 15 minutes is used for the heartbeat/time-out period.

monitord Log File

The **monitord** daemon writes license events to the text file **/var/security/monitord_log**. This log file can be used by the system administrator to retrieve auditing information regarding license usage. A system administrator can also use this information to determine the sources of errors that cause license transactions to fail or **monitord** to exit.

monitord Lockfile

There should never be more than one instance of **monitord** running at one time on a system. To prevent this from happening, **monitord** uses a lockfile to determine whether another instance has already been started. The lockfile is called **/etc/security/monitord_lock**. **monitord** looks for this file when it is started on a system. If the lockfile is detected, **monitord** exits immediately. If the lockfile does not exist, **monitord** creates it and continues operating. When **monitord** exits, it removes the lockfile. If **monitord** terminates abnormally, the lockfile remains on the system and prevents **monitord** from restarting. If this occurs, the system administrator should check the log file for any errors, correct the errors, remove the lockfile, and restart **monitord**.

Chapter 4. Configuring an iFOR/LS Environment

This chapter describes the steps to be performed by the iFOR/LS administrator when configuring the iFOR/LS licensing system.

There are two configuration files involved in the iFOR/LS configuration process. The steps performed by both files are explained to help iFOR/LS administrators in understanding the configuration changes made when enabling the system for iFOR/LS. These files are:

- **netls_config**
- **netls_first_time**

Also sample environments have been selected to help system administrators configuring iFOR/LS within existing production environments that might require special configuration processes, files, or methods. Use this information as a quick guide to the most relevant configuration items within the different environments. This information is not intended to be a replacement of the original product documentation.

Refer to “iFOR/LS General Concepts ” on page 1-1 and to the *AIX Version 4.1 iFOR/LS System Management Guide* if more information is required.

Network Licensing with iFOR/LS

The Information For Operation Retrieval/License System (iFOR/LS) design differs considerably from other licensing methods. One or more license servers run on nodes strategically distributed throughout a network. Each license server, maintaining its own license database, is responsible for a portion of the licenses installed for a given product. When a user requests a license to run a software application, the user’s node locates one of the license servers that handle licenses for that application and requests a license. iFOR/LS may be installed on any node in the network, including the node running the application. Different iFOR/LS servers can all be managed from a central point.

iFOR/LS on AIX is implemented using:

- NCS components:
  - Local Location Broker (LLB)
  - Global Location Broker (GLB)
  - Location Broker Client Agent
  - Location Broker Data
  - Location Broker Administrative Tools
- Setup and definition files:
  - iFOR/LS
  - NCS

NCS Components

This section describes the structure and function of the Location Broker software and databases.

Local Location Broker

The Local Location Broker (LLB) is a server that maintains a database of information about objects and interfaces located on the local host. It runs as the daemon **llbd**. The LLB also provides application programs with access to its database and implements the Location Broker forwarding service. An LLB must run on any host that runs NCS servers.

The forwarding facility of the LLB eliminates the need for a client to know the specific port that the server uses and thereby helps to conserve well-known ports. The LLB listens on one well-known port per address family (**llbd** uses UDP port 135 in AIX). It forwards any messages that it receives to the local server that exports the requested object. Forwarding is particularly useful when the client that requests a service already knows the host where the server is running. The server can use a dynamically assigned opaque port (number 1024 to 5000); it needs to register only with the LLB at its local host, not with the GLB. To access the server, the client needs to specify the object, the interface, and the host, but not a specific port.

Global Location Broker

The Global Location Broker (GLB) is a server that maintains information about objects and interfaces throughout the network or internet. Clients typically make lookup calls to the GLB when they do not know on which host an specific server is running.

There are two versions of the GLB daemon: **glbd** and **nrglbd**. The two versions provide the same functionality to clients, but **glbd** is replicatable, while **nrglbd** is not.

Note: These two versions cannot interoperate on the same network or internet. It is recommended that the **glbd** be used if possible. The iFOR/LS software provides the replicatable **glbd**.

Location Broker Client Agent

The Location Broker Client Agent is a set of library routines called by application programs to access LLB and GLB databases. When a program issues any Location Broker call, the call goes to the Client Agent at the local host. The Client Agent performs the actual remote lookup or update of information in the appropriate Location Broker database.

Location Broker Data

Each entry in a Location Broker database contains information about:

- Object
- Interface
- Location of server that exports the interface to the object

Fields in an iFOR/LS Database Entry

The following table describes the fields in a database entry:

Field	Description
Object UUID	Unique ID of the object
Type UUID	Unique ID that specifies the object type
Interface UUID	Indicates whether the object is global or not
Flag	Indicates whether the object is global or not
Annotation	64 characters of user-defined data
Socket Address Length	Size of socket address
Socket Address	Location of the server that exports the interface to the object

Location Broker Administrative Tools

NCS includes utilities that administer the Location Broker: **lb_admin**, the Location Broker administrative tool, and **drm_admin**, the Data Replication Manager administrative tool.

- **lb_admin** allows inspection or modification of the contents of a Location Broker database. It provides lookup, register, unregister, and garbage-collection operations. It can perform these operations on any LLB or GLB database.
- **drm_admin** manages the replication of the GLB database. It can inspect or modify replica lists, merge databases to force convergence among replicas, stop servers, and delete replicas. It does not look up, register, or unregister database entries.

Refer to the *AIX Version 4.1 iFOR/LS System Management Guide* for more information on the syntax and options of the preceding tools.

Setup and Definition Files

Following is a list of files and directories that are part of iFOR/LS and NCS, respectively. Refer to “Summary of iFOR/LS Files” on page 1-4 for more information on the most important NCS and iFOR/LS files.

iFOR/LS Files

/usr/lib/netls/bin/ls_targetid	Reports system target ID
/usr/lib/netls/conf/netls_config	License server config shell script 1
/usr/lib/netls/conf/netls_first_time	License server config shell script 2
/etc/rc.netls	License server startup file
/usr/lib/netls/bin/netlsd	License server daemon
/usr/lib/netls/conf/log_file	Log file
/usr/lib/netls/conf/cur_db	netlsd daemon database file
/usr/lib/netls/conf/lic_db	netlsd daemon database file
/usr/lib/netls/conf/nodelock	Nodelock file
/usr/lib/netls/bin/ls_admin	License administration tool / X11 GUI
/usr/lib/netls/bin/ls_stat	License statistics tool / X11 GUI
/usr/lib/netls/bin/ls_rpt	License report tool
/usr/lib/netls/bin/ls_tv	Tests for active netlsd server(s)
/usr/lib/netls/bin/ls_find_svr	Verifies running netlsd server(s)
/usr/lib/netls/ark/ils/	18n commands used during netls_config
/usr/lib/netls/ark/bin/help/	Help files
/usr/lib/netls/ark/cat1/	Catalog help files
/usr/lib/netls/ark/lib/En_US	Message files

NCS 1.5.1 Files

/etc/rc.ncs	NCS startup file
/usr/lib/ncs/bin/llbd	Local Location Broker daemon
/usr/lib/ncs/bin/glbd	Global Location Broker daemon
/usr/lib/ncs/bin/nllbd	Nonreplicatable Local Location Broker daemon
/usr/lib/ncs/bin/nrglbd	Nonreplicatable Global Location Broker daemon
/tmp/llbdbase.dat	Global Location Broker daemon database
/etc/ncs/glb.e	Global Location Broker database file
/etc/ncs/glb.p	Global Location Broker propagation queue
/etc/ncs/glb_site.txt	Pointer file to Global Location Broker host
/etc/ncs/glb_obj.txt	NCS cell UUID of the local host
/etc/ncs/glb_sites	RLM / NCS 1.1 used this file (Compatibility)
/usr/lib/ncs/bin/lb_find	Location Broker find tool
/usr/lib/ncs/bin/lb_admin	Location Broker administrative tool
/usr/lib/ncs/bin/uuid_gen	UUID generator command
/usr/lib/ncs/bin/drm_admin	Data Replication Manager Administrative tool
/usr/lib/ncs/bin/stcode	Translate tool
/usr/lib/ncs/lib/uuidname.txt	UUID <–> name mapping file
/usr/lib/ncs/cat/files/	Catalog help files
/usr/lib/ncs/cat/utils–a/	Catalog help files
/usr/lib/ncs/perf/	NCS example
/usr/lib/ncs/lib/En_US/	Message files

iFOR/LS Configuration Files

The variety of iFOR/LS license types may be divided into two groups:

- Nodelocked licenses
- Concurrent-use and use-once licenses

When iFOR/LS is configured for nodelocked licenses, encrypted information about the vendor, product, and license must be added to the **/usr/lib/netls/conf/nodelock** file. The iFOR/LS license server does not administer nodelocked licenses; therefore, if only nodelocked licenses are used on one system, iFOR/LS does not need to be configured for that system. However, if the iFOR/LS enabled application uses the iFOR/LS shared libraries, the client runtime (**client.rte**) option of the **bos.ifor_ls** package must be installed. Refer to the *AIX Version 4.1 iFOR/LS System Management Guide* for more information on installing iFOR/LS nodelocked licenses.

When concurrent-use licenses or use-once licenses are to be used, iFOR/LS has to be configured following the steps described in installing iFOR/LS for use with concurrent-use and use-once licenses in *AIX Version 4.1 iFOR/LS System Management Guide*.

The iFOR/LS software provides two configuration shell scripts **/usr/lib/netls/conf/netls_config** and **/usr/lib/netls/conf/netls_first_time** to help iFOR/LS administrators configure and start license services with iFOR/LS.

Warning: netls_config and **netls_first_time** are suited for general configurations only. When configuring already existing specific NCS environments, you might need to modify these shell scripts or perform some preliminary setup before starting them.

The following sections describe the execution steps performed by both the **netls_config** and the **netls_first_time** shell scripts.

netls_config Steps

The steps performed by the **/usr/lib/netls/conf/netls_config** shell script provided with the iFOR/LS software are as follows:

1. Entries are added to **/etc/inittab** to automatically start **/etc/rc.ncs** after reboot.
2. The **/etc/rc.ncs** shell script is the place where the local and global location brokers are started during boot time. The default **/etc/rc.ncs** file contains the following:

```
#!/bin/sh
####################################################################
#       rc.ncs  ––      NCS startup file
####################################################################
#–– 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
```

Note: The **llbd** is always started before the **glbd**. This is required because the global location broker will register itself to the local location broker. The LLB uses the well-known TCP/IP port 135. This port is not defined in **/etc/services;** it is compiled into the executables. The **glbd** uses a run-time-assigned TCP/IP port number.

3. Entries are added to **/etc/inittab** to start the **/etc/rc.netls** shell script during boot phase.

4. The **/etc/rc.netls** shell script is the place where the iFOR/LS server daemon, **netlsd**, is started during boot time. The default **/etc/rc.netls** file contains the following:

```
#!/bin/sh
#####################################################################
#       rc.netls  ––    NetLS startup file
#####################################################################
if /usr/lib/ncs/bin/lb_find –q
then
     echo ”Starting the netlsd daemon”
     startsrc –s netlsd  && echo ”Starting up the NetLS Daemon”
else
     echo ”The GLB server is not responding;”
     echo ”NetLS server cannot be started.”
fi
```

Note: The **netlsd** daemon is only started if the return code from the **lb_find** command is zero. It means that the **lb_find** command received a response from a **glbd**. Otherwise, an error message is shown on the screen. The **lb_find –q** command is used to query for a GLB using the RPC mechanism. The GLB can run anywhere in the network.

The **netlsd** daemon uses a runtime-assigned TCP/IP port number. It needs a global location broker because it has to register itself with host name and port number to this GLB before offering services. Applications looking for **netlsd** servers contact the **GLB** daemon to find their way to these servers.

5. A test is made to see if there is already an initialized database for the GLB. This is done by testing for the existence of the files **/etc/ncs/glb.e** and **/etc/ncs/glb.p**. These files are the GLB databases. If these files exist on the local system, **netls_config** asks if they should be used for the ongoing NCS configuration. If the user chooses not to use the already initialized database, the following files are removed:

   /etc/ncs/glb.e

   /etc/ncs/glb.p

   /etc/ncs/glb_log

   /etc/ncs/glb_site.txt

   /etc/ncs/glb_obj.txt

Warning: The **glb_site.txt** file is used to directly contact hosts running GLB daemons. This file is required if the GLB server cannot be contacted via broadcasting. If the **glb_site.txt** file is removed here, the **lb_find** command being run later will not find GLB daemons running in other subnets (**lb_find** uses broadcasting). To get around this problem just delete the **glb.e** and **glb.p** files before starting **netls_config** if you want to overwrite the existing GLB definitions without losing internet definitions.

6. The **lb_find** command is run to find running global location broker daemons in the network. The output of this command is parsed to gather information about the NCS cell environment.

Warning: The **lb_find** command uses the network broadcast addresses of the local system to find running GLB daemons. It is absolutely necessary to set up the **/etc/ncs/glb_site.txt** file before running **netls_config** when the local host will be running in an NCS cell which spans multiple networks (internet).

The **/etc/ncs/glb_site.txt** might contain several entries that are checked in order from top to bottom. The following example shows a **glb_site.txt** file with two entries. The first entry specifies an **ip** host name, while the second entry uses an **ip** address.

Example:

```
# cat  /etc/ncs/glb_site.txt

  ip:mybox.austin.ibm.com
  ip:#128.33.22.11
```

Refer to the *AIX Version 4.1 iFOR/LS System Management Guide* for more information on the **glb_site.txt** file format.

Notes:

1. In addition to the **/etc/ncs/glb_site.txt** file, it is also necessary to set up the **/etc/ncs/glb_obj.txt** file with the UUID of the NCS cell if the cell spans different networks and if it is not the NCS default cell. Use **ftp** or **rcp** to copy the file from the host where the **glbd** is running. If there is no **glb_obj.txt** file on the host where the **glbd** is running, this host is a member of the NCS default cell.

   Example 1:

   ```
   # cat   /etc/ncs/glb_obj.txt           (default NCS cell)

           333b91c50000.0d.00.00.87.84.00.00.00
   ```

   Example 2:

   ```
   # cat  /etc/ncs/glb_obj.txt           (alternate NCS cell)

            65e543fd89ac.02.09.03.01.5c.00.00.00
   ```

2. If your fully qualified host name contains the string **default** or **alternate**, then the shell script will not work properly.

3. An NCS location broker configuration ordinarily contains one or more **glbds** belonging to one GLB cell. The names of the alternate cells can differ among hosts in the local network. Use only the UUID to find the right NCS cell. For example, the alternate_1 cell on host A may show up as alternate_2 on host B. The reason for this is that the **lb_find** command is responsible for the naming. Hosts with more than one active network interface may see more **glbds** than others. An NCS cell does not really have a name; it consists of all hosts with the same entry in the **/etc/ncs/glb_obj.txt** file or of all hosts without this file. Hosts without an **/etc/ncs/glb_obj.txt** file are in the default NCS cell and use the following UUID:

   ```
   333b91c50000.0d.00.00.87.84.00.00.00
   ```

7. Depending on the selected option, the **netls_config** shell script will do one of the following:

   a. Exit without any further action, to be started again later.

b. Set up the local host in the default NCS cell. The **/etc/ncs/glb_obj.txt** file is removed and the following two cases are covered:
   - Set up as the first **glbd** in the default cell, or
   - Set up as an additional **glbd** in the default cell. In this case, a remote **glbd** is used as a reference at the first-time startup.

c. Set up the local host in a new alternate NCS cell. In this case the **netls_config** shell script executes the **/usr/lib/ncs/bin/uuid_gen** command to generate a new UUID and writes it into the **/etc/ncs/glb_obj.txt** file.

d. Set up the local host in an already existing alternate NCS cell. The script displays all NCS cells that can be found in the network and names them alternate_1, alternate_2, and so on. Please verify the UUID, which is also displayed, for the selection of the alternate-cell environment.

8. If the **llbd** daemon is not running, remove its database file **/tmp/llbdbase.date**.
9. Create the **/usr/lib/netls/conf/netls_first_time** shell script. This shell script contains the commands needed to initialize NCS and start up the NCS and iFOR/LS daemons the first time. After that, the daemons will start automatically (from **/etc/inittab***)* whenever the system is booted.

netls_first_time Steps

The steps performed by the **/usr/lib/netls/conf/netls_first_time** shell script created with the **netls_config** shell script are as follow:

1. Start the **llbd** using the command:

   ```
   # startsrc –s llbd
   ```

2. If an existing GLB database at the local host was selected for use, then start the **glbd** using the command:

   ```
   # startsrc –s glbd
   ```

   In addition to that, the leading # for the GLB entry in the file **/etc/rc.ncs** is removed.
3. If the local system should run the first **glbd** daemon in the cell (either default or alternate), start the **glbd** daemon using the following command:

   ```
   # startsrc –s glbd –a ”–create –first –family ip”
   ```

   In addition to that, the leading # for the GLB entry in the file **/etc/rc.ncs** is removed.
4. If the local system should run a **glbd** daemon in a cell that is already set up and running (either default or alternate), then start the **glbd** daemon using the following command:

   ```
   # startsrc –s glbd –a ”–create –from reference_glbd_hostname”
   ```

   In addition to that, the leading # for the GLB entry in the file **/etc/rc.ncs** is removed. The remote host *reference_glbd_hostname* is used to replicate all entries in the remote GLB database (**glb.e**) on the local system.
5. Start the iFOR/LS daemon (**netlsd**) using the command:

   ```
   # startsrc –s netlsd
   ```

Configuring NCS Cells

This section lists the tasks to be performed by NCS cell administrators when configuring NCS cells. A detailed explanation about each of these tasks can be found in the iFOR/LS configuration files section on page 4-6 and the chapter on Network Computing System in the *AIX Version 4.1 iFOR/LS System Management Guide*.

The tasks to be performed are:

- Install, configure, and start the TCP/IP daemons needed.
- The Network Computing Kernel (NCK) must be installed on all nodes. On AIX Version 4.1, this software is provided with the client runtime (**client.rte**) option of the **bos.ifor_ls** package.
- The **llbd** daemon must run on every node that runs an iFOR/LS license server (**netlsd**). It is recommended to start the **llbd** on every node in the network. Start the **llbd** daemon with the following command:

  ```
  # startsrc –s llbd
  ```

- When the first system within an alternate cell is configured, the **/etc/ncs/glb_obj.txt** file is needed. This file contains the UUID for the cell and can be generated manually with the command:

  ```
  # /etc/ncs/uuid_gen > /etc/ncs/glb_obj.txt
  ```

  or through the **netls_config** shell script provided with the iFOR/LS software. Any successive member added to this cell needs to remote copy the **/etc/ncs/glb_obj.txt** file from the first system. This can also be done using the **netls_config** shell script.

Note: Since this is the reason for most NCS problems, ensure the content of the **glb_obj.txt** file is the same on all nodes within a cell. Refer to the iFOR/LS configuration files section, on page 4-6, for more information concerning configuring alternate cells.

When the **default** cell is used, the **/etc/ncs/glb_obj.txt** file is not needed.

- If you are using the **/etc/ncs/glb_site.txt** file, make sure it is pointing to a valid **glbd** server.
- When started for the first time, the first **glbd** daemon in a newly created cell has to be started using the following command:

  ```
  # startsrc –s glbd –a ”–create –first –family ip”
  ```

  or through the **netls_first_time** shell script generated by **netls_config**. This is needed only the first time the **glbd** daemon is started. Any successive start of **glbd** may be done using the following command:

  ```
  # startsrc –s glbd
  ```

- At least one **glbd** daemon must be running in a cell. The node running the **glbd** daemon must have started the **llbd** daemon before the **glbd**.
- If running more than one **glbd** daemon within a cell (GLB replicas), the system time of all **glbd** hosts must be within 2 minutes of each other. It is good practice to merge the databases on all **glbd** servers every week using the **drm_admin** tool.

Managing Global Location Broker Replicas

To avoid losing access to the GLB information whenever the **glbd** server is unreachable, NCS version 1.5.1 provides the ability to maintain replicated copies of the cell’s GLB database. The GLB replicas are maintained and synchronized by the data replication Manager (DRM), which is part of the **glbd** daemon.

A replicated **glbd** might be running on any number of nodes within a cell. GLB replicas are created through the **netls_first_time** shell script or by using the following command:

```
# startsrc –s glbd –a ”–create –from ip:hostname”
```

When this command is used, the following database files are remotely copied from the reference machine specified with the *hostname* parameter.:

/etc/ncs/glb.e	Contains information about servers, vendors, and products within the cell.
/etc/ncs/glb.p	Used by the DRM to propagate any updates made to the **glb.e** database.

The preceding command is needed only the first time the replicated **glbd** is started. Any successive start of the **glbd** may be done by using the following command:

```
# startsrc –s glbd
```

What to do if glbd –create –from ip:<host> doesn't work ?

1. Enter the command from the command line.
2. Check that the files **/etc/ncs/glb.p** and **/etc/ncs/glb.e** have been created:

   If they have been created, check the content of the **glb_log** file.
3. If the content looks similar to the following, the clocks between both machines differ by more than 2 minutes:

   ```
   (DRM) 1994/03/17.14:36:38  creating replica of object glb.
   (DRM) 1994/03/17.14:36:39  [1c030012] unable to create replica of
   object glb.
   ?(GLB) cannot create replica – clocks skewed (network computing
   system/DRM) (GLB) exiting
   ```

   Synchronize the clocks using:

   – The **date** command.

   – One machine as the referenced time provider. Start the **setclock** command by using the **cron** facility on every **glbd** server. The following example shows the **crontab** entry on the `cadrs1` server when node `mitc53h` is used as the referenced time provider:

   ```
   # crontab –l

   0  11 * * * /usr/bin/errclear –d S,O 30
   0  12 * * * /usr/bin/errclear –d H 90
   30 22 * * * /usr/bin/setclock mitc53h    1>/dev/null
   2>/dev/null
   ```

   – Set up the **timed** daemon on all networked machines that need to be synchronized.

   – Use an external time provider, such as DCF77, in combination with programs based on the Network Time Protocol (see RFC 1305 NTP Version 3 Specification for more details).
4. If **no** then:

   a. Check the TCP/IP communication.
   b. Check for available disk space on the file system containing the **/etc/ncs** directory by using the following command:
   ```
   # df /etc/ncs
   ```
   c. Check the error log entries using the **errpt –a** command.
   d. Check for possible hardware problems, and start normal problem determination procedures.

Setting Up iFOR/LS Servers

This section lists the tasks to be performed by iFOR/LS administrators when configuring iFOR/LS license servers. A detailed explanation about each of these tasks can be found in the iFOR/LS configuration files section, on page 4-6, and the chapter on installing and configuring iFOR/LS in the *AIX Version 4.1 iFOR/LS System Management Guide*.

The tasks to be performed are:

- Ensure that the TCP/IP software is installed and configured and the appropriate daemons have been started.
- Ensure that the NCS software is installed and configured and the appropriate daemons have been started. The iFOR/LS server should have the **llbd** daemon running. At least one **glbd** daemon must be running within the cell. On AIX Version 4.1, the NCS software is provided as an optional program product (OPP) within the **bos.net** package.
- Install the iFOR/LS software using SMIT or the **installp** command.
- Protect the iFOR/LS administrative tools and databases by setting the proper file permissions located in the following directories:
  - **/usr/lib/netls/bin**
  - **/usr/lib/netls/conf**
- Ensure that the clocks of all iFOR/LS clients are synchronized within 24 hours with the iFOR/LS server clock. Nodes whose clocks are not synchronized may not be able to acquire licenses.
- Start the iFOR/LS server daemon (**netlsd**). There might be several **netlsd** daemons running within a cell.

Note: It is recommended that several iFOR/LS servers be operational within a cell and have the licenses split among them to avoid losing access to all licenses, which would happen if all licenses were managed by one iFOR/LS server and this server became unreachable. Refer to “iFOR/LS Planning Tips” on page 2-5 for more information.

The **netlsd** daemon must be running before most iFOR/LS tools can be used.

- Execute the **ls_stat –i** command to verify that the iFOR/LS installation was successful.
- Optionally, create and maintain a user file to protect license usage. For more information on creating and maintaining user files, refer to the *AIX Version 4.1 iFOR/LS System Management Guide*.
- Use the **ls_admin** tool (SMIT, command line or GUI) to add licenses to each iFOR/LS server by entering passwords for vendors and products (these are provided by the software vendor). In the case of nodelocked licenses the **/usr/lib/netls/conf/nodelock** file must be used. Each iFOR/LS license server decodes the password to establish the vendor, product, and license information in its database.
- Use the **ls_stat** and **ls_rpt** tools to obtain license status information and to generate appropriate reports, respectively.
- Run the **ls_tv** test and verification tool to verify that all iFOR/LS license servers within the cell are running properly.
- In case of problems, refer to the *AIX Version 4.1 iFOR/LS System Management Guide*, for information about troubleshooting.

Getting the Target ID for License Generation Request

Before an application enabled for iFOR/LS can be used, licenses for this application must be obtained from the application vendor. Customers will need to contact the vendor and supply him with the target ID of the iFOR/LS license server (machine where **netlsd** is running) that is going to manage these licenses.

There are two possibilities for getting the target ID (typically the CPU planar ID) from a running system:

1. Using the iFOR/LS **ls_targetid** command.
2. Using the standard AIX Operating System **uname** command.

Note: Both commands retrieve the target but in a different format. In case customers have not yet installed iFOR/LS but want to request license keys for a specific machine, it is possible to use the **uname** output.

Example of ls_targetid Output:

```
# /usr/lib/netls/bin/ls_targetid

PREFERRED NetLS Target ID (CPU Planar ID)
-----------------------------------------
             1093710

ALTERNATE Target ID (Link Level Address)
---------------------------------------------
             903015C
```

Example of uname –m Output:

```
# uname –m

000109371000
```

Converting uname Output into ls_targetid Format

Converting the **uname** output into **ls_targetid** format is done by using the following procedure:

- Remove leading zeros and two trailing zeros from the **uname** output.

```
000<–– 1093710 ––>00    ===> 1093710
```

Setting Up iFOR/LS Clients

This section lists the tasks to be performed by iFOR/LS administrators when configuring iFOR/LS client machines. A detailed explanation about each of these tasks can be found in the iFOR/LS configuration files section, on page 4-6, and the chapter on Network Computing System in the *AIX Version 4.1 iFOR/LS System Management Guide*.

- Ensure the TCP/IP software is installed and configured and the appropriate daemons have been started. Test the TCP/IP communication with the IFOR/LS server machine.

- Install the Network Computing Kernel (NCK) if not already done. In AIX Version 4.1, this software is provided with the client run-time (**client.rte**) option of the **bos.ifor_ls** package and is installed automatically during BOS installation.

Note: Some applications try to contact the **llbd** daemon on the local host first. If the **llbd** is not running on the local machine, the application waits a timeout period before using the broadcasting mechanism to contact any running **glbd** daemon. This may cause a long startup period for an application. If this is the case for your application, you may consider installing the NCS software on the iFOR/LS client and having the **llbd** daemon running. On AIX Version 4.1, the NCS software is provided as an OPP within the **bos.net** package.

- At least one **glbd** daemon must be running within the cell. It may be running on any machine within the cell.
- Should this iFOR/LS client join an alternate cell, then the **/etc/ncs/glb_obj.txt** file must be copied remotely from another member of the cell.
- When you compare the content of the remotely copied **/etc/ncs/glb_obj.txt** file (UUID) with the **glb_obj.txt** content on the **glbd** server machine and the **glb_obj.txt** content on the iFOR/LS server machine, make sure the iFOR/LS client is a member of the correct NCS cell. The UUID must be the same within a cell.
- When working within the default cell, the **/etc/ncs/glb_obj.txt** file is not required.
- You may create and use the **/etc/ncs/glb_site.txt** file to contact **glbd** servers in a specific order. **glbd** servers can be specified either by host name or by **ip** address as the following example shows:

```
# cat /etc/ncs/glb_site.txt

ip:gandalf
ip:9.39.1.92
```

  In the preceding example, the client machine first tries to contact the **glbd** daemon running on the machine with the hostname `gandalf`. If the attempt fails, it tries to contact the **glbd** daemon running on the machine specified by the **ip** address `9.39.1.92`.

- If the NCS software is installed, use the **lb_find** command to test the NCS communication.
- Test the applications which require iFOR/LS licenses.
- In case of problems, refer to the section on troubleshooting in the *AIX Version 4.1 iFOR/LS System Management Guide*.

iFOR/LS and DCE for AIX

This section describes topics the iFOR/LS administrator must be aware of when implementing iFOR/LS within a DCE environment; for example, the conflicts that might appear due to the fact that DCE and NCS both use Remote Procedure Calls (RPC) but implement the local location broker (LLB) in a different way.

Overview of AIX DCE

Distributed Computing Environment (DCE) for AIX is a layer between the AIX operating system and network on the one hand, and the distributed application on the other. DCE provides services that allow a distributed application to interact with a collection of possibly heterogeneous computers, operating systems, and networks as if they were a single system. Several technology components work together to implement the DCE layer. For more detailed information, refer to the DCE documentation in InfoExplorer.

DCE Remote Procedure Call (RPC)

Network Computing System (NCS) 2.0 was selected by OSF as the base technology for the DCE Remote Procedure Call (RPC). The AIX DCE RPC facility consists of both a development tool kit and a run-time service. The development tool kit consists of a programming language and its compiler. It supports the development of distributed applications following the client/server model. It automatically generates code that transforms procedure calls into network messages. The run-time service implements the network protocols by which the client and server sides of an application communicate. DCE RPC also includes software for generating unique identifiers, which are useful for identifying service interfaces and other resources.

Using iFOR/LS in a DCE RPC Environment

NCS applications can coexist with DCE on the same machine. The DCE RPC daemon (**rpcd**) is used to provide the services of the NCS **llbd** daemon and is run instead of **llbd**.

To run NCS 1.5.1 applications, for example, iFOR/LS applications, within a DCE environment, do the following:

1. Install the NCS and iFOR/LS software if not already on the system. Refer to the *AIX Version 4.1 iFOR/LS System Management Guide* for information on installing iFOR/LS.
2. Set up the NCS system manually or use the **/usr/lib/ncs/config/netls_config** shell script.

Note: After the **netls_config** shell script is used, the system displays a message asking you to run the **/usr/lib/ncs/config/netls_first_time** shell script. *Do not* run the shell script yet. This will be done in step 5.

3. Edit the **netls_first_time** shell script, and change the line with the command:

   ```
   startsrc –s llbd
   ```

   to:

   ```
   /etc/rc.dce rpc
   ```

4. Change the file **/etc/rc.ncs** as follows:

```
#!/bin/sh
######################################################################
#       rc.ncs  ––      NCS startup file
######################################################################
#–– Uncomment the following line to start the Local Location Broker
#startsrc –s llbd

#–– Uncomment the following line to start the rpcd daemon
/etc/rc.dce rpc

#–– Uncomment the following line to start the Global Location Broker
#startsrc –s glbd
```

5. Start the **netls_first_time** shell script, or do the configuration manually.

Notes:

1. The two daemons **/usr/lib/ncs/bin/llbd** and **/opt/dcelocal/bin/rpcd** cannot run on the same machine. The **/opt/dcelocal/bin/rpcd** daemon provides the same functionality as the Local Location Broker Daemon (**llbd**) along with the functionality for DCE.
2. Normally the **/etc/rc.dce** script starts **rpcd**. The script checks whether **rpcd** is running or not, and if **rpcd** is already running, then the script will not attempt to start another daemon.
3. The **lb_admin** command is still used to administer the NCS 1.5.1 end-point mappings. The DCE **rpccp** command is used to administer the DCE RPC end-point mappings. The **lb_admin** command will not show the DCE RPC end-point mappings and the DCE **rpccp** command will not show the NCS 1.5.1 end-point mappings.

Remote Procedure Call Daemon

The Remote Procedure Call daemon (**rpcd**) is a server that provides the end-point map service for the local host. The daemon typically resides on disk as **/opt/dcelocal/bin/rpcd**. It starts at boot time and runs in the background on any host that runs RPC-based servers.

The primary function of **rpcd** is to provide the end-point map service, enabling clients to reach the particular end points on which servers are listening. (For some readers, port is a more familiar synonym for end point.) To support RPC applications based on NCS Version 1.5.1, **rpcd** also implements the registration and lookup functions of the NCS LLB.

For the most part, the LLB and end-point map components of the **rpcd** are independent of each other. They are different logical databases and in fact have separate physical database files (names) and administrative utilities. The **rpccp** tool administers only the end-point map component; the NCS 1.5.1 **lb_admin** program administers only the LLB component.

Both components are used together only when the **rpcd** performs the Endpoint Map mapping function or the LLB forwarding function. This is to allow NCS 1.5.1 clients to use compatible DCE servers, and DCE clients to use compatible NCS 1.5.1 servers. The **rpcd** first searches the end-point map for a compatible server and, if none is found, the LLB database is then searched. When the IP protocol is used, the **rpcd** process listens on port 135.

The end-point map typically resides on disk in the **/opt/dcelocal/var/rpc/rpcdep.dat** file. After a system reboot, RPC-based servers restart and register again with the end-point map service, so the database file has to be deleted before **rpcd** starts.

Note: **rpcd** also creates the NCS 1.5.1 LLB file, which resides on disk in the **/opt/dcelocal/var/rpc/rpcdllb.dat** file. Like **rpcdep.dat**, the **rpcdllb.dat** database needs to be deleted at system boot time.

Where to Run rpcd

An **rpcd** process must run on every host that runs an RPC-based server program. For example, an **rpcd** must run on any host that runs the DCE Distributed File Service (DFS) Protocol Exporter, the server side of any other DCE service, or the server side of any RPC-based user application.

To simplify planning and configuration, you can run the **rpcd** process on every host. However, **rpcd** is mandatory only on hosts that run RPC-based servers. Only one **rpcd** can run on a host at a time.

How to Start rpcd

Before **rpcd** starts, any underlying network services on which RPC depends must be available. On most UNIX systems, for example, network interfaces and routing services must be enabled. Once these transport layer services are established, you can start **rpcd**. After **rpcd** starts, RPC-based servers can start.

Because **rpcd** listens in some address families on privileged or reserved end points, it must be started with root privileges. Typically, **rpcd** starts each time a host boots. Usually, a **/opt/dcelocal/etc/dce.rc** file is created with a link from **/etc/dce.rc** and with **/etc/rc** or **/etc/rc.local** invoking **/etc/dce.rc**. The **/opt/dcelocal/etc/dce.rc** file is responsible for deleting the database files and starting the **rpcd**.

Note: You can kill and restart the **rpcd** after the system has booted. The restarted **rpcd** will not lose any previously registered server bindings. It simply reloads its internal databases from the **rpcdep.dat** and **rpcdllb.dat** files.

iFOR/LS and AIX High-Availability Cluster Multi-Processing

AIX High Availability Cluster Multi-Processing (HACMP) environments are complex systems designed to offer services in a highly available fashion. Due to the configuration complexity involved with HACMP, and due to the special HACMP network design, it is necessary to carefully plan and analyze before adding iFOR/LS to the system. Additional time must be planned to test the environment before enabling it for production.

This section should help HACMP administrators in planning the selection of NCS cells, in understanding how the HACMP network design (**service**, **standby**, and **boot** ip addresses) might affect the NCS broadcasting process, and in executing the configuration steps.

The sample configuration shell script on page A-1 may be used to enable most HACMP environments for iFOR/LS.

AIX HACMP Overview

HACMP is the availability control system for AIX. HACMP services include automatic fallover and recovery/restart for those applications deemed critical by the customer in his or her HACMP system design.

The HACMP software allows the customer to shut down a system for scheduled maintenance and restart the system with automatic resynchronization of application and data from a backup or fallover system. The high-availability attribute allows critical business processes to remain in operation during component or subsystem failure or during maintenance downtime.

Availability is enhanced (transparently to end users) by using redundant hardware components. Incremental system facilities are provided by HACMP to existing application bases for four-way scalability through clustering.

HACMP utilizes industry-standard TCP/IP communication protocols as the transport mechanism between server machines in a highly available cluster, as well as between servers and client machines that require services from these servers. HACMP can utilize multiple TCP/IP interfaces on current adapters, specifically Ethernet, Token Ring, FDDI, and Serial Optical Channel Converter.

HACMP provides cluster monitoring and automatic fallover to backup processors if a server failure has been detected. The SNMP feature within HACMP can generate SNMP alerts and send them to any network manager, for example NetView, reporting on cluster status if an out-of-service condition occurs. This can also be monitored in the HACMP cluster console service if NetView is not used.

The HACMP fallover scripts can be extended and customized for configuration and application use by system administrators or application programmers.

iFOR/LS Guidelines in an HACMP Environment

Generally HACMP servers are stable systems planned and configured to offer services in a highly available fashion, so it’s a good idea to install and set up the iFOR/LS daemons on these systems as well.

Since the iFOR/LS license passwords are typically tied to the system’s CPU planar ID, it is not possible to set up license takeover with an HACMP configuration. To avoid losing all licenses if one server goes down or is unreachable, the number of licenses should be split among two or more servers (refer to “Selecting iFOR/LS Servers” on page 2-4, for more information on selecting iFOR/LS servers). The number of available licenses in the network will decrease if one or more iFOR/LS servers are down or unreachable, by the number of licenses managed by the failed iFOR/LS servers.

With a two-server HACMP configuration, it might be useful to allow HACMP clients to have their key requirements satisfied by either one of the servers. In this case, the following is a convenient set up:

- Both systems should run on the same NCS cell.
- To avoid a single point of failure, each system should run the GLB daemon.
- Each system should run its own iFOR/LS server daemon.
- The number of licenses should be split equally between both servers running iFOR/LS.
- All HACMP clients should be in the same NCS cell as the servers.

Daemon Startup in an HACMP Environment

In an HACMP environment the startup of the NCS and iFOR/LS daemons is different from the normal system startup when **ip**-address takeover is enabled (refer to the *High Availability Cluster Multi-Processing/6000 Administration Guide* for more information on starting the HACMP subsystems).

The different daemons and shell scripts involved within HACMP startup process are started in the following sequence:

1. If the HACMP cluster subsystems (**clstrmgr**, **cllockd**, **clsmuxpd**, and **clinfo**) have been selected to be restarted automatically after system reboot, the **init** process uses the respective entry in **/etc/inittab** to run the **/etc/rc.cluster** shell script.

   If they are restarted manually, **/etc/rc.cluster** can be started via SMIT or directly on the command line.

2. The **rc.cluster** shell script calls the **/usr/sbin/cluster/clstart** script, where the different subsystems are started managed by the System Resource Controller (SRC). Thus the command used to start the cluster manager daemon **clstrmgr** is:

   ```
   # startsrc –s clstrmgr
   ```

3. The **clstrmgr** daemon on the local node communicates with the cluster managers on all other HACMP servers within the cluster, asking for permission to join the running cluster. The local node can join the cluster only if ALL running cluster managers agree. If this is the case the local clstrmgr daemon calls the **/usr/sbin/cluster/samples/node_up** shell script.

4. Since it is the local node that is attempting to join the cluster, **node_up** calls the **/usr/sbin/cluster/samples/node_up_local** shell script to start the process of acquiring the locally managed resources and offering services.

5. One of the needed resources is the local **ip** service address. The **node_up_local** shell script calls the **/usr/sbin/cluster/samples/acquire_service_addr** utility in order to switch from the **ip** boot address to the **ip** service address.

6. After switching the **ip** addresses, the **acquire_service_addr** utility runs the command **telinit a** to start the TCP/IP servers and network daemons marked with the **a** flag in the file **/etc/inittab**.

Note: The NCS and iFOR/LS startup files in the **/etc/inittab** file should have the **a** flag posted in order to be started by **telinit a**.

The TCP/IP servers and network daemons should be started in following sequence:

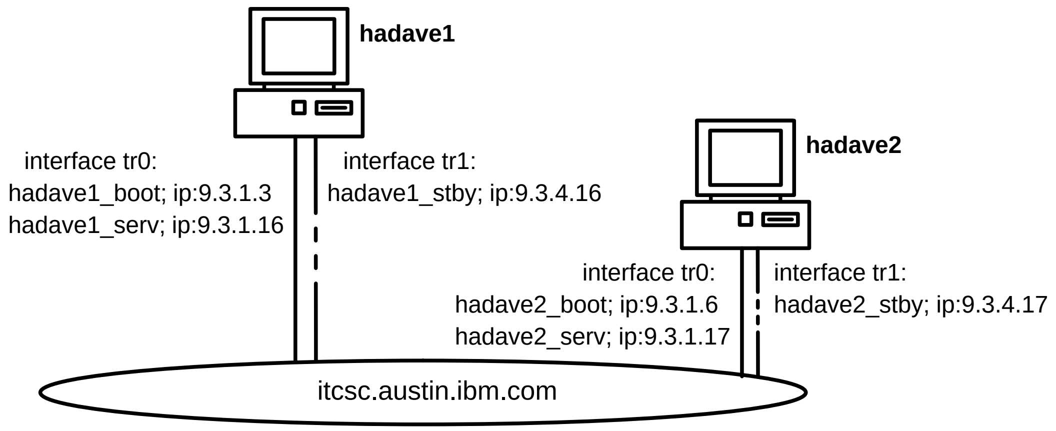
a. **/etc/rc.tcpip** to start up the TCP/IP services on the **ip** service address

b. **/etc/rc.nfs** to start up the NFS and NIS services

c. **/etc/rc.ncs** to start up the NCS 1.5.1 services (**llbd** and **glbd**)

d. **/etc/rc.netls** to start up the iFOR/LS services (**netlsd**)

Note: The steps described above may be different on your system because HACMP services may be started differently.



HACMP Sample Figure

iFOR/LS Setup in an HACMP Environment

The following steps have to be performed to install and configure iFOR/LS on an HACMP system:

1. Install the iFOR/LS software on all HACMP servers using SMIT or the **installp** command. This installs the files and does not configure them.

2. Finish the HACMP installation before you configure the iFOR/LS software.

   Note: All HACMP systems should run on their service network address and not on the boot network address during the iFOR/LS configuration process.

3. The **hacmp_netls.config.sh** shell script included in Appendix A is a sample shell script that may be used to configure iFOR/LS in an HACMP environment. Although the shell script may suffice for most of the HACMP configurations, changes may be necessary to adapt it to your specific HACMP environment (HACMP is very flexible). Read the shell script carefully before using it to make sure it does suit your needs. The shell script has not been submitted to any formal test and is distributed AS IS.

   After the shell script has been run, you may continue with step 7.

4. If the **hacmp_netls.config.sh** shell script is not used, make sure the HACMP standby network interface is changed to the detach status before starting the iFOR/LS configuration. You can do this using one of the following commands:

   ```
   # /usr/sbin/chdev –l interfacename –a state=detach
   ```

   or

```
# /etc/ifconfig interfacename detach
```

Note: Using NCS, iFOR/LS will broadcast over all attached interfaces. This may cause problems if iFOR/LS uses the standby network interfaces too. Clients will not reach their servers and services may be stopped due to IP address takeover on the standby network interfaces.

5. Users have two choices for configuring the iFOR/LS software:

   a. Manually, if the user is a very experienced NCS user and does not need any configuration files. However, check the configuration shell scripts supplied with the iFOR/LS software.

   b. Using the iFOR/LS-supplied configuration shell scripts explained on page 4-6 and following the recommendations suggested.

   The most important file to watch for is **/etc/inittab**. After iFOR/LS configuration the **/etc/inittab** file should look similar to this:

```
init:2:initdefault:
brc::sysinit:/sbin/rc.boot 3 >/dev/console 2>&1 # Phase 3 of system boot
powerfail::powerfail:/etc/rc.powerfail >/dev/console 2>&1  # d51225
rc:2:wait:/etc/rc             > /dev/console 2>&1 # Multi–User checks
srcmstr:2:respawn:/etc/srcmstr            # System Resource Controller
harc:2:wait:/usr/sbin/cluster/harc.net    # HACMP6000 network startup
rctcpip:a:wait:/etc/rc.tcpip  > /dev/console 2>&1 # Start TCP/IP daemons
rcnfs:a:wait:/etc/rc.nfs      > /dev/console 2>&1 # Start NFS Daemons
rcncs:a:wait:sh /etc/rc.ncs
netlsd:a:wait:sh /etc/rc.netls >/dev/console 2>&1 # start netls
clvm6000:2:wait:/usr/sbin/cluster/cllvm status    # Check CLVM stat
clinit:a:wait:touch /usr/sbin/cluster/.telinit # Must be last entry in inittab!
```

6. Reactivate the HACMP standby adapters using the following command:

```
# chdev –l interfacename –s state=up
```

7. Verify and test your HACMP cluster.

Verifying the HACMP Cluster after iFOR/LS Configuration

After iFOR/LS has been configured, the **ncs_test.sh** shell script provided in Appendix B may be used to test the configuration.

The **ncs_test.sh** shell script shows the output of the **lb_admin** command. This command monitors and administers location broker registrations.

The **lb_admin** program inspects and operates on the contents of the databases of the two location brokers, the local location broker (LLB), and the global location broker (GLB). The **set_broker** subcommand can be used to have the **lookup** subcommand display the contents of either the LLB or the GLB database.

The following concepts are used:

- Object

  An object Universal Unique Identifier (UUID) in the format xxxxxxxxxxxx.xx.xx.xx.xx.xx.xx.xx.xx.xx, where x is a hexadecimal digit. An * (asterisk) is the wildcard character for this argument.

- Type

  A type UUID in the format xxxxxxxxxxxx.xx.xx.xx.xx.xx.xx.xx.xx.xx, where x is a hexadecimal digit. An * (asterisk) is the wildcard character for this argument.

- Interface

  An interface UUID in the format xxxxxxxxxxxx.xx.xx.xx.xx.xx.xx.xx.xx.xx, where x is a hexadecimal digit. An * (asterisk) is the wildcard character for this argument.

- Location

  A socket address specifier, designating the location of a local or global location broker, consisting of an address family, a host name, and a port number.

LLB Output

```
ncs_test.sh started from root@hadave1 on aixterm at 21:50:16

LLB   is using UDP port(s) := 135            <– this is fixed.
GLB   is using UDP port(s) := 1802 1803      <– these may differ
NetLS is using UDP port(s) := 1843 1847 1849  <– on your system.

Using long RPC timeouts
------------
Using local broker @ ip:hadave1.itsc.austin.ibm.com[0]
------------
/etc/ncs/uuidname.txt
      object = 6618870a0ee1.02.09.03.04.10.00.00.00
        type = 333b91de0000.0d.00.00.87.84.00.00.00
   interface = 33599c670000.0d.00.00.24.34.00.00.00
”rdrm” @ ip:hadave1.itsc.austin.ibm.com[1802]
------------
      object = 6618870a0ee1.02.09.03.04.10.00.00.00
        type = 333b91de0000.0d.00.00.87.84.00.00.00
   interface = 3e188a057000.0d.00.00.be.8a.00.00.00
”rdrm_debug” @ ip:hadave1.itsc.austin.ibm.com[1802]
------------
      object = 6618870a0ee1.02.09.03.04.10.00.00.00
        type = 333b91de0000.0d.00.00.87.84.00.00.00
   interface = 339a6e4fe000.0d.00.00.87.84.00.00.00
```

```
”rdrm_applic” @ ip:hadave1.itsc.austin.ibm.com[1802]
------------
      object = 6618870a0ee1.02.09.03.04.10.00.00.00
        type = 333b91de0000.0d.00.00.87.84.00.00.00
   interface = 333b2e690000.0d.00.00.87.84.00.00.00
”/sys/ncs/glbd” @ ip:hadave1.itsc.austin.ibm.com[1802]
------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 34275946a000.0d.00.00.05.55.00.00.00
   interface = 34275946a000.0d.00.00.05.55.00.00.00
”NLS @ ip:hadave1.itsc.austin.ibm.com” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 3bd624ea7000.0d.00.00.80.9c.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0] @ ip:hadave1.itsc.austin.ibm.com” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 4ca0fd5cf000.0d.00.02.1a.9a.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0]: Hewlett–Packard NetLS Test” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
```

In above output, the connection to the global location broker is defined with the first four entries. The fully qualified host name and the UDP portnumber 1802 is the pointer to a running **glbd** process. Due to the configuration setup, this process is running on the local host and should be the preferred GLB. Be aware of the fact that your name resolution (DNS, NIS, or **/etc/hosts**) must be working correctly to allow proper NCS communication.

The last three entries in the LLB database reflect that the **netlsd** daemon is running on host `hadave1` and that this daemon is registered with the GLB.

The UDP port used is 1847.

GLB Output

```
Data from GLB replica: ip:hadave1.itsc.austin.ibm.com

------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 34275946a000.0d.00.00.05.55.00.00.00
   interface = 34275946a000.0d.00.00.05.55.00.00.00
”NLS @ ip:hadave1.itsc.austin.ibm.com” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 3bd624ea7000.0d.00.00.80.9c.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0] @ ip:hadave1.itsc.austin.ibm.com” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
      object = 66188b46c772.02.09.03.04.10.00.00.00
        type = 4ca0fd5cf000.0d.00.02.1a.9a.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0]: Hewlett–Packard NetLS Test” @
ip:hadave1.itsc.austin.ibm.com[1847] global
------------
      object = 661ca6ce8a65.02.09.03.04.11.00.00.00
        type = 34275946a000.0d.00.00.05.55.00.00.00
   interface = 34275946a000.0d.00.00.05.55.00.00.00
”NLS @ ip:hadave2.itsc.austin.ibm.com” @
ip:hadave2.itsc.austin.ibm.com[1292] global
------------
      object = 661ca6ce8a65.02.09.03.04.11.00.00.00
        type = 3bd624ea7000.0d.00.00.80.9c.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0] @ ip:hadave2.itsc.austin.ibm.com” @
ip:hadave2.itsc.austin.ibm.com[1292] global
------------
      object = 661ca6ce8a65.02.09.03.04.11.00.00.00
        type = 4ca0fd5cf000.0d.00.02.1a.9a.00.00.00
   interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0]: Hewlett–Packard NetLS Test” @
ip:hadave2.itsc.austin.ibm.com[1292] global
---------
```

Both HACMP systems are running in the same NCS cell, and both of them are running the **glbd** process to provide a fail-safe NCS environment. The GLB database is replicated on both hosts. This means that both hosts contain the same information; therefore, a **lookup** on either host should show the same information.

Each host is running the **netlsd** daemon.

On `hadave1`, **netlsd** is reachable using the UPD port 1847. On `hadave2`, **netlsd** is reachable on the UPD port 1292. The port numbers may be different in your environment because these are run-time-assigned numbers.

iFOR/LS and Load Leveler for AIX

This section describes how to manage load leveler when using it to automatically start applications within an iFOR/LS licensing system.

Load-Leveler Overview

Load leveler is a distributed, network-based, job-scheduling program for AIX workstations.

Each workstation may be individually configured for load leveler, declaring itself a job submitter, a compute server, or both. Each workstation specifies to the central manager, the resources it has for running jobs. These resources include such attributes as memory size, architecture, licensed programs installed, and the classes of jobs it will accept. Each workstation may also specify how its resources will be used.

Personal workstations may be configured so their resources are available to load leveler only when not in use by their owners. Dedicated compute servers may be configured to run only certain types of jobs or to have a preference for certain jobs. Job placement is done by matching a flexible job description to the machine configuration.

Load leveler also attempts to balance the workload over the workstation pool. Jobs can also be submitted from a simple AIXwindows interface that constructs a default job description and allows the user to specify any additions or changes.

In summary, the system provides the capability to locate, allocate, and deliver resources to users, while adhering to load balancing, fair scheduling, and optimal usage of resources.

Problem Description

When load leveler is used in unattended mode to automatically start an application, problems might arise if the application’s license request doesn’t get satisfied due to lack of licenses, and the iFOR/LS enabled application has not implemented the iFOR/LS **l**icense request queueing feature.

Since load leveler attempts to start the job only once, even if later there happens to be enough available licenses, jobs won’t get started after such a failure until the system administrator starts them manually. It would be a good idea to provide load leveler with the ability to be aware of license availability and license status, to avoid application starting failure, and to allow automatic retry in a configurable fashion.

Users might write a shell script that checks for free licenses before starting the application. A configurable number of attempt retries might be posted, and the **sleeping** time between retries might also be configurable. An example of such a shell script is included in the following section.

License-Checking Shell Script

The following shell script example can be used to check for free licenses before starting an iFOR/LS-enabled application that needs one or more licenses. iFOR/LS provides the **ls_stat** command to check for licenses. Use the example as a guide and adapt it to your environment. This example has not been submitted to any formal test and is distributed AS IS.

This shell script prevents load-leveler jobs from failing due to a shortage of available licenses. It provides a simple way to make load-leveler jobs fail-safe. Of course iFOR/LS internally provides the possibility for queuing license requests, but it is the responsibility of the application programmer to make use of these features. If the license-request-queueing feature has been included within the application, the shell script is not needed.

The **lic_test.sh** shell script needs access to the **/usr/lib/netls/bin/ls_stat** and **/usr/lib/netls/bin/_ls_stat** commands on the local system. In case these files are not available, copy them from your iFOR/LS server. The formatting of the **ls_stat** output is done via the **awk** command and the **lic_test.awk** awk definition file, also shown here.

lic_test.sh Shell Script

```
#!/bin/ksh
# @(#)  Version 1.0  lic_test.sh  Example  03/18/94
#=================================================================#
#   Script :  lic_test.sh                                         #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer             #
#   Date   :  94/03/18                                            #
#   Update :  94/03/24                                            #
#   Info   :  This script shows a way to find out if there        #
#             are enough licenses available for a specified.      #
#             product.                                            #
#                                                                 #
#   Input  :  The name of a product which is licensed             #
#             via iFOR/LS. Make sure you specify it in single     #
#             quotes if the name contains blank characters.       #
#                                                                 #
#   Output :  Will return the number of licenses for the          #
#             specified product or 0 if the requested             #
#             number of licenses can not be found                 #
#                                                                 #
#-----------------------------------------------------------------#
AWKF=”/usr/local/bin/lic_test.awk”   #  pathname to awk script
AWK=”/usr/bin/awk”                   #  pathname to awk command
LSSTAT=”/usr/lib/netls/bin/ls_stat”  #  pathname to ls_stat command
PROG=”$(basename $0)”                #
#-----------------------------------------------------------------#
#  This is the usage information.                                 #
#-----------------------------------------------------------------#
usage()
{
 echo ”\nUsage: $PROG ’<product>’
[<need_lic>][<retry>][<timeout>][<mulfact>]”
 echo ”\n\t<product>  : Produkt Name e.g. ’CATIA.MECH FBD–AIX/6000’”
 echo ”\t<need_lic> : Number of licenses needed (Default 1)”
 echo ”\t<retry>    : Retry value               (Default 5)”
 echo ”\t<timeout>  : Start value of timeout    (Default 1)”
 echo ”\t<mulfact>  : Timeout multiplication    (Default 2)\n”
 exit 1
}
#-----------------------------------------------------------------#
#  Test for valid input or display usage() information.           #
#-----------------------------------------------------------------#
if [ –z ”${1}” –o ”${1}” = ’h’ –o ”${1}” = ’?’ ]; then
   usage
fi
#-----------------------------------------------------------------#
#  Prepare the variables.                                         #
#-----------------------------------------------------------------#
PRODUCT=”${1}”
integer LIC=0
integer RETRY=1
integer NEEDLIC=${2:–1}
integer RETRY_MAX=${3:-5}
integer TIMEOUT=${4:-1}
integer MULFACT=${5:-2}
#-----------------------------------------------------------------#
#                                                                 #
#  That’s the master loop of this shell script. It will use the   #
#  ls_stat command to find out about installed licenses. The      #
#  output of the command is piped to awk and the awk script will  #
#  format and return the number of available licenses for the     #
#  the requested product. The awk script will return 0 in all     #
#  cases like failure to find the netlsd, wrong product name etc. #
#  So be warned this is just a very simple example to show you    #
#  how things can be done.                                        #
#                                                                 #
#  You also can specify a RETRY value so that the ls_stat         #
#  command is called several times before the script exits.       #
#  A TIMEOUT value will be used between these different attempts  #
#  this TIMEOUT value can be recalculated after each retry. A     #
#  simple method is just to double the value. Use the MULFACT     #
#  parameter to change this behavior.                              #
#                                                                 #
#-----------------------------------------------------------------#
while [ $RETRY -le $RETRY_MAX ]
do
    LIC=$(${LSSTAT} -t -p ”${PRODUCT}” | ${AWK} -f ${AWKF})
    if [ $LIC -lt $NEEDLIC ]; then
#---------
        echo ”\n\t($RETRY) Found $LIC licenses of $PRODUCT but need
$NEEDLIC.”
        echo ”\tWill sleep for $TIMEOUT sec......and try
again.($RETRY_MAX)”
#---------
        sleep $TIMEOUT
        TIMEOUT=TIMEOUT*MULFACT
        RETRY=RETRY+1
        LIC=0
    else
        RETRY=RETRY_MAX+1
        echo ”\n\tFound $LIC licenses for $PRODUCT.”
    fi
done
#-----------------------------------------------------------------#
#  Return the number of available Lic’s to the caller or 0.       #
#-----------------------------------------------------------------#
exit ${LIC}
```

lic_test.awk Shell Script

The following awk script is called by the **lic_test.sh** shell script to format the **ls_stat** output for a certain product. This is a very simple solution, so please adapt it to your needs.

```
# @(#) awk filter program to format ls_stat output – 03/18/94
#========================================================================#
#    AWK      : lic_test.awk                                             #
#    Authors  : M. Crisanto / L. Denefleh / F. Kraemer                   #
#    Date     : 94/03/18 ––––– (yy/mm/dd)                                #
#    Update   : 94/03/23 ––––– (yy/mm/dd)                                #
#------------------------------------------------------------------------#
# begin() – of awk                                                       #
#------------------------------------------------------------------------#
BEGIN {
         start  = 0;
         target = 0;
         lic    = 0;
      }
#========================================================================#
# main() – loop of awk                                                   #
#------------------------------------------------------------------------#
#  Look for the string ”End of License Usage” – that’s the last          #
#  line of the ls_stat output.                                           #
#------------------------------------------------------------------------#
$1 ~ /End/                  &&
$2 ~ /of/                   &&
$3 ~ /License/              &&
$4 ~ /Usage/                { if (start == 2) start=0; }
#------------------------------------------------------------------------#
#  Look for the string ”In Use Total Active Available” – 2 lines after  #
#  this string the actual numbers follow.                                #
#------------------------------------------------------------------------#
$1 ~ /In/                   &&
$2 ~ /Use/                  &&
$3 ~ /Total/                &&
$4 ~ /Active/               &&
$5 ~ /Available/            { start=1; }
#------------------------------------------------------------------------#
#  Two lines after we found the string ”In Total Active Available”      #
#  we look for the last item on each line which follows until the       #
#  ”End of License Usage” is reached. All results are summed in the     #
#  variable lic.                                                         #
#------------------------------------------------------------------------#
{
   if (start >= 1)
   {
       if (target >= 2)
       {
          i=NF;
          lic = (sprintf(”%d”,$i)) + lic;
       }
       ++target;
       ++start;
    }
}
#------------------------------------------------------------------------#
# end() – send the value of lic to the caller. This value may be 0      #
#         in case of an error occurred the output is wrong etc.         #
#         This is just a very simple example !                          #
#-----------------------------------------------------------------------#
END   {  printf("%d\n",lic); }
#-----------------------------------------------------------------------#
#  ok we're done                                                        #
#-----------------------------------------------------------------------#
```

Use of the lic_test.sh Shell Script

If **lic_test.sh** is called without any argument, the following usage information is displayed.

```
# lic_test.sh

  Usage:lic_test.sh '<product>' [<need_lic>][<retry>][<timeout>][<mulfact>]

          <product>  : Product Name e.g. 'CATIA.MECH FBD-AIX'
          <need_lic> : Number of licenses needed (Default 1)
          <retry>    : Retry value               (Default 5)
          <timeout>  : Start value of timeout    (Default 1)
          <mulfact>  : Timeout multiplication    (Default 2)
```

The default function is to call **lic_test.sh** with a product name and query for all free licenses for this product. Please use single quotes if the product name contains blank characters. The name of the product is the same as specified when licenses were added to iFOR/LS using the **ls_admin** tool.

```
# lic_test.sh 'CATIA.MECH FBD-AIX'
  >
  >     Found 3 licenses for CATIA.MECH FBD-AIX.
```

The following example will query for four licenses for the product **CATIA.MECH FBD-AIX**. The query request will be repeated four times using 3, 12, 48 and 192 seconds as the timeout value between these retries.

```
# lic_test.sh 'CATIA.MECH FBD–AIX' 4 4 3 4
  >
  >     (1) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
  >     Will sleep for 3 sec......and try again.(4)
  >
  >     (2) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
  >     Will sleep for 12 sec......and try again.(4)
  >
  >     (3) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
  >     Will sleep for 48 sec......and try again.(4)
  >
  >     (4) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
  >     Will sleep for 192 sec......and try again.(4)
```

The following example will query for four licenses for **CATIA.MECH FBD-AIX.** The query request will be repeated six times using one second as the timeout value between these retries.

```
# lic_test.sh 'CATIA.MECH FBD–AIX/6000' 4 6 1 1
  >
  >     (1) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
  >     Will sleep for 1 sec......and try again.(6)
```

```
>
>     (2) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
>     Will sleep for 1 sec......and try again.(6)
>
>     (3) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
>     Will sleep for 1 sec......and try again.(6)
>
>     (4) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
>     Will sleep for 1 sec......and try again.(6)
>
>     (5) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
>     Will sleep for 1 sec......and try again.(6)
>
>     (6) Found 3 licenses for CATIA.MECH FBD–AIX but need 4.
>     Will sleep for 1 sec......and try again.(6)
```

Load-Leveler Job

The following load-leveler-job shell script (**catia_job.sh**) shows a small example of how to start a CATIA batch utility only when a valid license is available.

Load-Leveler-Job Shell Script

```
#!/bin/ksh
#-----------------------------------------------------------------#
#  Filename:  catia_job.sh                                        #
#  Info:      This job will start a CATIA utility                 #
#  Note:      The next 3 lines are specific for LoadLeveler.      #
#----------------------------------------------------------------
#
# @  output  =  /tmp/catia_job.out
# @  error   =  /tmp/catia_job.err
# @  queue
#
#-----------------------------------------------------------------#
#  A small example of a job.                                      #
#-----------------------------------------------------------------
#
integer lic=0
integer needlic=1
#-----------------------------------------------------------------#
#  Test for a free license.                                       #
#-----------------------------------------------------------------
#
if [ –x /usr/local/bin/lic_test.sh ]
then
   /usr/local/bin/lic_test.sh ’CATIA.MECH FBD–AIX’ $needlic
   lic=$?
else
   echo ”\n\tCould not find lic_test.sh shell script\n”
   exit –1
fi
#-----------------------------------------------------------------#
#  If the license test was ok – start a simple CATIA job.        #
#-----------------------------------------------------------------
#
if [ $lic –ge $needlic ]
then
   echo ”\n\tStart CATIA Job now”
   catutil –l catprj –i /home/catadm/CATPRJ.in –o /tmp/cat.out
```

```
else
   echo ”\n\t**ERROR** Could not find enough licenses.\n”
   exit –1
fi
```

Starting the Load-Leveler Job

The preceding example of the load-leveler job is started using the following command:

```
$ /home/loadl/bin/llsubmit ./catia_job.sh
```

Load Leveler Output File

This is a sample of the content of the **/tmp/catia_job.out** output file used by load leveler:

```
****************************************
*                                      *
*  CATIA SOLUTIONS VERSION 4 RELEASE 1  *
*          For IBM RISC System          *
*                                      *
****************************************

--------------------------------------------------
Found 3 licenses for CATIA.MECH FBD-AIX.
Start CATIA Job now
--------------------------------------------------

--------------------------------------------------
   CATIA SOLUTIONS
 VERSION 4 RELEASE 1
CATUTIL STARTING ....
  Submitting CATPRJ
  Input  file name : /home/catadm/CATPRJ.in
  Output file name : /tmp/cat.out
--------------------------------------------------

--------------------------------------------------
   CATIA SOLUTIONS
 VERSION 4 RELEASE 1
CATUTIL ENDED ....
   Submitted utility : CATPRJ
 –>Consult Output file name /tmp/cat.out for traces
 EXECUTION SUCCESSFUL
--------------------------------------------------
```

Load Leveler Notification

Load leveler notifies the user through the AIX mail facility when a job has been run. The mail looks similar to this:

```
> From: LoadLeveler
> To: catadm@strider.itsc.austin.ibm.com
> Subject: LoadLeveler Job 7.0
>
> Your LoadLeveler job
>         ./catia_job.sh
> exited with status 0.
>
> Submitted at:        Wed Mar 23 14:04:57 1994
> Completed at:        Wed Mar 23 14:05:05 1994
> Real Time:             0 00:00:08
>
> Job User Time:      0 00:00:01
> Job System Time:    0 00:00:01
> Total Job Time:     0 00:00:02
>
> Shadow User Time:       0 00:00:00
> Shadow System Time:     0 00:00:00
> Total Shadow Time:      0 00:00:00
>
> Starter User Time:       0 00:00:00
> Starter System Time:     0 00:00:00
> Total Starter Time:      0 00:00:00
>
> Virtual Image Size:  2 Kilobytes
```

Chapter 5. iFOR/LS License Management

This section describes how to manage the log data collected by the iFOR/LS servers and how to find out which is the most important information within these log files. System administrators should plan and set up data interpretation mechanisms to keep track of the iFOR/LS servers and the license transactions. Collecting these data can help customers with their license-capacity and investment planning.

iFOR/LS Data Collection

Information data concerning the iFOR/LS servers is automatically collected by the **glbd** and **netlsd** daemons. This data is stored in the following files:

- **/etc/ncs/glb_log**
- **/usr/lib/netls/conf/log_file**

Note: A good idea would be to place this information in a common directory.

1. As root user create a new directory in the **/var** file system:

```
# mkdir /var/netls
```

2. Stop the iFOR/LS daemon and the GLB daemon on the local system:

```
# stopsrc –s netlsd
# stopsrc –s glbd
```

3. Move the existing log files to the new created directory:

```
# mv /etc/ncs/glb_log /var/netls/glb_log
# mv /usr/lib/netls/config/log_file /var/netls/netls_log
```

Note: Although you can create a new **glb_log** file by using the **touch** command, you cannot do the same for the iFOR/LS **log_file**. If this file does not exist, it is created with a specific file structure whenever **netlsd** gets started. If the file exists but the file structure is not as expected by **netlsd**, the daemon will terminate.

4. Use a symbolic link to replace the files in their location:

```
# cd /etc/ncs
# rm glb_log
# ln –s /var/netls/glb_log glb_log


# cd /usr/lib/netls/conf
# rm log_file
# ln –s /var/netls/netls_log log_file
```

5. Restart the GLB daemon and the iFOR/LS daemon on the local system:

```
# startsrc –s glbd
# startsrc –s netlsd
```

The /etc/ncs/glb_log Log File

This file stores information about operations related to the GLB daemon (**glbd**). Because the information stored in this file is in readable ASCII format, no reporting tools are necessary.

The /usr/lib/netls/conf/log_file Log File

This file contains information about **netlsd** activities and license events reported by the iFOR/LS (**netlsd**) daemon. Information about the following activities and license events is stored:

License granted

License released

License checked

License multiple grant

License multiple release

Wait queue

Grant/wait

Wait removed

Vendor added

Vendor deleted

Vendor renamed

Product added

Product deleted

Product renamed

Error

License timed out

Message

License server start

License server exit

The information stored in the **log_file** file can be accessed by using the following commands:

- **/usr/lib/netls/bin/ls_stat** (reports on the status of licenses)
- **/usr/lib/netls/bin/ls_rpt** (reports on the history of license server events)

Use of the ls_stat Command

The **ls_stat** command provides both a graphical and an ASCII interface. This command provides both users and administrators with status information on product licenses. This information includes:

– List of the available **netlsd** servers on the network

– List 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.

Use of the ls_rpt Command

The **ls_rpt** command provides an ASCII interface only. This command generates reports on all kinds of license server activities. Information filters allow different kinds of events to be reported (as well as all events), and use dates, software vendors, products, and users.

Note: Besides monitoring license server activities, **ls_rpt** can track demand for software products. This kind of information is useful for planning future purchases of licenses for those products.

Important Messages for the iFOR/LS System Administrator

This section should help system administrators on recognizing which iFOR/LS errors and messages are relevant for system operation and how to manage the data for automatic report of these errors and messages. Sample shell scripts are included to help system administrators to develop useful reporting tools for use as input when planning new license purchases or network/department license redistribution.

The relevant iFOR/LS errors and messages can be grouped into:

- Error messages caused by rejected license-request events
- Fatal errors
- Queueing information on license-request events

Using iFOR/LS Log Data on Rejected License Requests

It is highly recommended to check and track rejected license requests. This is very important since application vendors often use the hardstop license-request policy for their applications enabled for iFOR/LS. When the hardstop license-request policy is implemented, applications will not start if the license request is not satisfied. This might be very unsatisfactory or even fatal for many business processes.

Collecting information about how often license requests for the different products are rejected might help with planning for license purchases, deciding on how to distribute licenses across machines, and other investment-planning activities.

Using ls_rpt to Track Rejected License Requests

This example shows how the **ls_rpt** command can be used to track rejected license requests. It uses an **awk** script to process the needed output lines and write them into a temporary file.

A shell script is included that uses the **ls_rpt** command with the **–r1** option to collect information about rejected license requests for all products managed by the specified iFOR/LS server. The default iFOR/LS server is the local server.

The shell script also uses the **–x** option to delete log file entries in order to save disk space. You will have to make some changes if you want to keep these entries. In general, this shell script is a very simple example to show how things may be done.

The output is processed by an **awk** script before being stored into the file **/tmp/ifor_ls_report_$TODAY_$SERVER** where TODAY is the current date, and SERVER is the specified iFOR/LS server.

The sample shell script might be started through the **cron** facility as shown here.

lic_deny.sh Shell Script

```
#!/bin/ksh
# @(#) Version 1.1  lic_deny.sh  Example  03/26/94
#=================================================================#
#   Script :  lic_deny.sh                                         #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer             #
#   Date   :  94/03/24                                            #
#   Update :  94/03/26                                            #
#   Info   :  This script checks the iFOR/LS log_file against    #
#             deny events. This might be started weekly as a      #
#             cronjob.                                            #
#                                                                 #
#   Input  :  The name of the iFOR/LS server to check.            #
#             Default is the execution hostname.                  #
#                                                                 #
#   Output :  File /tmp/ifor_ls_report_$DATE_$SERVER with         #
#             information about failed license requests for       #
#             all installed products on the specified iFOR/LS     #
#             server.                                             #
#                                                                 #
#-----------------------------------------------------------------#
AWKF="/home/netlstest/lic_deny.awk"    #  pathname to awk script
AWK="/usr/bin/awk"                     #  pathname to awk command
LSRPT="/usr/lib/netls/bin/ls_rpt"      #  pathname to ls_stat command
PROG="$(basename $0)"                  #
#-----------------------------------------------------------------#
#  This is the usage information.                                 #
#-----------------------------------------------------------------#
usage()
{
 echo "\nUsage: $PROG '<server_name>'"
 echo "\n\t<server_name>: Name of the iFOR/LS server (Default current HOST)"
 exit 1
}
#-----------------------------------------------------------------#
#  Test for valid input or display usage() information.           #
#-----------------------------------------------------------------#
if [ "${1}" = 'h' -o "${1}" = '?' ]; then
   usage
fi
#-----------------------------------------------------------------#
#  Prepare the variables.                                         #
#-----------------------------------------------------------------#
integer RET PER
SERVER="${1-$(hostname)}"
DATUM=$(date +%D)
TODAY=$(date +%m_%d_%y)
INF="/tmp/.ifor_ls_tmp"                       #  input file coming from awk
OUTF="/tmp/ifor_ls_report_${TODAY}_${SERVER}" #  report file for sysadmin
#-----------------------------------------------------------------#
# Prepare the outputfile.                                         #
#-----------------------------------------------------------------#
echo "\t*********************************************************  " > $OUTF
echo ”\t*       iFOR/LS license deny evaluation for                *  ” >> $OUTF
echo ”\t*       Server    : $SERVER                                *  ” >> $OUTF
echo ”\t*       Date      : $DATUM                                 *  ” >> $OUTF
echo ”\t**********************************************************  ” >> $OUTF
echo ”\t following products with failed license requests have been    ” >> $OUTF
echo ”\t found :                                                       ” >> $OUTF
echo ”\t last value shows the rejection rate in percent               ” >> $OUTF
echo ”\t**********************************************************\n” >> $OUTF
#------------------------------------------------------------------#
#                                                                  #
#  This is the shell script’s master loop. It will use the        #
#  ls_rpt command  with the –r1 (report filter 1) flag.           #
#  The awk script will return 0 in case of no log file entries.   #
#  On all other values it’s required to process the INF file if   #
#  the percentage values are greater than zero.                   #
#  So be warned this is just a very simple example to show you    #
#  how things can be done.                                        #
#                                                                  #
#------------------------------------------------------------------#
RET=$(${LSRPT} –n ”${SERVER}” –r1 | ${AWK} –f ${AWKF})
#
if [ ”$RET” –lt ”1” ]
then
  echo ”\t no events recorded                                       \n” >> $OUTF
  exit ${RET}
fi
#
cat $INF | \
while read DUMMY PERCENT DUMMY WHOLE_STRING
do
#
PER=${PERCENT%”%”}
#
if [ $PER –gt 0 ]
then
  echo ”\t $WHOLE_STRING                                              \n” >>
$OUTF
fi
done
#------------------------------------------------------------------#
# write the last line in outputfile                               –#
#------------------------------------------------------------------#
echo ”\t**********************************************************\n” >> $OUTF
#------------------------------------------------------------------#
#                                                                  #
#  To prevent log_file from wasting disk space we will run the    #
#  ls_rpt command  with the -x  (delete log_file entries) flag.   #
#  So be warned this is just a very simple example to show you    #
#  how things can be done.                                        #
#                                                                  #
#------------------------------------------------------------------#
RET2=$(${LSRPT} –n ”${SERVER}” –x ”${DATUM}”) 1>/dev/null 2>/dev/null
#------------------------------------------------------------------#
# EOF                                                              #
#------------------------------------------------------------------#
rm ${INF}
exit ${RET}
```

lic_deny.awk Shell Script

```
# @(#) awk filter program to format ls_rpt –f output – 03/25/94
#========================================================================#
#    AWK      : lic_deny.awk                                             #
#    Authors  : M. Crisanto / L. Denefleh / F. Kraemer                   #
#    Date     : 94/03/24 ––––– (yy/mm/dd)                                #
#    Update   : 94/03/25 ––––– (yy/mm/dd)                                #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
# begin() – of awk                                                       #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
BEGIN {
         OUTF   = ”/tmp/.ifor_ls_tmp”
         target = 0;
         start  = 0;
         ret    = 0;
      }
#=======================================================================#
# main() – loop of awk                                                   #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Look for the string ”finished.” – that’s the last line of the         #
#  ls_rpt –f output.                                                     #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
$1 ~ /finished./            { if (start >= 2) start=0; }
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Look for the string ”Index” – 2 lines after this string the info we   #
#  are looking for                                                       #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
$1 ~ /Vendor/                { start=1; }
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Two lines after we found the string ”Index” we start to look for      #
#  any item on each line which follows until the ”finished.” is reached.#
#  if an hit exist ret value is set to 1                                 #
#  we are interested in the last item of a line (NF) because it is the  #
#  percentage of failed request, and the whole line ($0).                #
#                                                                        #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
{
   if (start >= 1)
   {
       if (target >= 2)
       {
          i=NF;
          if ( $i > 0)
          {
           printf(”: %s : %s : \n”,$NF,$0)>OUTF;
           ret = 1;
          }
       }
       ++target;
       ++start;
    }
}
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
# end() – send the value of ret to the caller. This value may be 0       #
#         in case of no hits.                                            #
#         This is just a very simple example !                           #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
END   {  printf(”%d\n”,ret); }
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
```

```
#  ok we’re done                                                        #
#-----------------------------------------------------------------------#
```

cron Table Entry

The **cron** facility starts the **lic_deny.sh** shell script every Sunday at 22.30 hours. The entry in the cron table is as follows:

```
# crontab –l
         .
         .
0 11 * * * /usr/bin/errclear –d S,O 30
0 12 * * * /usr/bin/errclear –d H 90
01 4 * * * /etc/lpp/diagnostics/bin/test_batt 1>/dev/null 2>/dev/null
01 3 * * * /etc/lpp/diagnostics/bin/run_ela 1>/dev/null 2>/dev/null
30 22 * * 0 /home/netlstest/lic_deny.sh 1>/dev/null 2>/dev/null
```

Output File

```
************************************************************
*       iFOR/LS license deny evaluation for                *
*       Server    : strider                                *
*       Date      : 03/25/94                               *
************************************************************
following products with failed license requests have been
found :
last value shows the rejection rate in percent
************************************************************

Dassault Sys CATIA.MECH F 410   49     40          18%

************************************************************
```

Explanation:

Based on the **lic_deny.sh** shell script output, the system administrator has to determine the cause of the rejected license requests. Perhaps there are not enough licenses available in this cell for that application, and the customer must either order additional licenses or move licenses from other cells to this cell.

Managing Fatal License Events with iFOR/LS

It is highly recommended to track iFOR/LS reported data against fatal errors. This should be done frequently to avoid any error that might cause license-management failures. Due to the high impact of fatal errors within iFOR/LS, it is suggested implementing this database checking daily by using the **/usr/lib/netls/bin/ls_rpt –f** command.

This example shows checking by using shell programming and cron jobs. A shell script is included that uses the **ls_rpt** command with the **–f** option to collect information about fatal error events on a specified iFOR/LS server. The default iFOR/LS server is the local server.

The output of the **ls_rpt –f** command is evaluated by an **awk** script, and if any fatal error is found, electronic mail is sent to the system administrator.

lic_fatal.sh Shell Script

```
#!/bin/ksh
# @(#)  Version 1.0  lic_fatal.sh  Example  03/24/94
#=================================================================#
```

```
#   Script :  lic_fatal.sh                                          #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer               #
#   Date   :  94/03/24                                              #
#   Update :  94/––/––                                              #
#   Info   :  This script checks the iFOR/LS log_file against       #
#             fatal events. This should be started on a daily       #
#             basis as a cron job.                                  #
#                                                                   #
#   Input  :  The name of the iFOR/LS server to check.              #
#             Default is the execution hostname.                    #
#                                                                   #
#   Output :  In case of fatal error a electronic mail is sent to #
#             the system administrators.                            #
#                                                                   #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
AWKF=”/home/netlstest/lic_fatal.awk”   #  pathname to awk script
AWK=”/usr/bin/awk”                     #  pathname to awk command
LSRPT=”/usr/lib/netls/bin/ls_rpt”      #  pathname to ls_stat command
PROG=”$(basename $0)”                  #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  This is the usage information.                                   #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
usage()
{
 echo ”\nUsage: $PROG ’<server_name>’”
 echo ”\n\t<server_name>: Name of the iFOR/LS server (Default current
HOST)”
 exit 1
}
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Test for valid input or display usage() information.             #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
if [ ”${1}” = ’h’ –o ”${1}” = ’?’ ]; then
   usage
fi
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Prepare the variables.                                           #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
integer RET
SERVER=”${1–$(hostname)}”
SYSADMIN1=”root@strider”
SYSADMIN2=”denefleh@mitc53h”
INFO=”a fatal event has been logged on iFOR/LS server $SERVER, pls \
      start problem determination .........”
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#                                                                   #
#  This is the shell script’s master loop. It will use the          #
#  ls_rpt command  with the –f (fatal) flag.                        #
#  The awk script will return 0 in case of no fatal log entry       #
#  So be warned this is just a very simple example to show you      #
#  how things can be done.                                          #
#                                                                   #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
RET=$(${LSRPT} –n ”${SERVER}” –f | ${AWK} –f ${AWKF})
if [ ”$RET” –gt ”0” ]
then
echo $INFO |mail –s ”iFOR/LS Fatal Problem on Server $SERVER” $SYSADMIN1
echo $INFO |mail –s ”iFOR/LS Fatal Problem on Server $SERVER” $SYSADMIN2
fi
```

```
        #-----------------------------------------------------------------#
        # EOF                                                             #
        #-----------------------------------------------------------------#
        exit ${RET}
```

lic_fatal.awk Shell Script

```
# @(#) awk filter program to format ls_rpt –f output – 03/24/94
#========================================================================#
#    AWK      : lic_fatal.awk                                            #
#    Authors  : M. Crisanto / L. Denefleh / F. Kraemer                   #
#    Date     : 94/03/24 ––––– (yy/mm/dd)                                #
#    Update   : 94/__/__ ––––– (yy/mm/dd)                                #
#------------------------------------------------------------------------#
# begin() – of awk                                                       #
#------------------------------------------------------------------------#
BEGIN {
         target = 0;
         start  = 0;
         ret    = 0;
      }
#========================================================================#
# main() – loop of awk                                                   #
#------------------------------------------------------------------------#
#  Look for the string ”finished.” – that’s the last line of the        #
#  ls_rpt –f output                                                      #
#------------------------------------------------------------------------#
$1 ~ /finished./            { if (start >= 2) start=0; }
#------------------------------------------------------------------------#
#  Look for the string ”Index” – 2 lines after this string the info we  #
#  are looking for                                                       #
#------------------------------------------------------------------------#
$1 ~ /Index/                { start=1; }
#------------------------------------------------------------------------#
#  Two lines after we found the string ”Index” we start to look for     #
#  any item on each line which follows until the ”finished.” is reached.#
#  if an hit exist ret value is set to 1                                 #
#------------------------------------------------------------------------#
{
   if (start >= 1)
   {
       if (target >= 2)
       {
          i=NF;
          if ( $i > 0)
          {
           ret = 1;
          }
       }
       ++target;
       ++start;
    }
}
#------------------------------------------------------------------------#
# end() – send the value of ret to the caller. This value may be 0      #
#         in case of no hits.                                            #
#         This is just a very simple example !                           #
#------------------------------------------------------------------------#
END   {  printf(”%d\n”,ret); }
```

```
#------------------------------------------------------------------------#
#  ok we're done                                                         #
#------------------------------------------------------------------------#
```

cron Table Entry

The **cron** facility starts the **lic_fatal.sh** shell script every day from Monday through Friday at 22.30 hours. The entry in the cron table is as follows:

```
# crontab –l
         .
         .
0 11 * * * /usr/bin/errclear –d S,O 30
0 12 * * * /usr/bin/errclear –d H 90
01 4 * * * /etc/lpp/diagnostics/bin/test_batt 1>/dev/null 2>/dev/null
01 3 * * * /etc/lpp/diagnostics/bin/run_ela 1>/dev/null 2>/dev/null
30 22 * * 0 /home/netlstest/lic_deny.sh 1>/dev/null 2>/dev/null
30 22 * * 1–5 /home/netlstest/lic_fatal.sh 1>/dev/null 2>/dev/null
```

Notifying System Administrator of Fatal Error

When a fatal error occurs, the system administrator is informed through electronic mail as follows. Upon receipt of this mail, he should start problem determination procedures.

```
From root Fri Mar 25 11:29:00 1994
Received: by strider.itsc.austin.ibm.com (AIX 3.2/UCB 5.64/4.03)
          id AA18596; Fri, 25 Mar 1994 11:28:59 –0600
Date: Fri, 25 Mar 1994 11:28:59 –0600
From: root
Message–Id: <9403251728.AA18596@strider.itsc.austin.ibm.com>
To: root@strider.itsc.austin.ibm.com
Subject: iFOR/LS Fatal Problem on Server strider
Status: R

a fatal event has been logged on iFOR/LS server strider, pls
start problem determination .........
```

/etc/ncs/glb_log File

Normally the **glb_log** file is used only to determine **glbd** daemon startup problems. Once the **glbd** daemon is up and running, there is no need to check the information stored in this file. For this reason no automated checking of the **glb_log** file is required.

For example, on the GLB server `strider` the contents of the **glb_log** file may be similar to following:

```
# pg /etc/ncs/glb_log

(DRM) 1994/03/17.14:11:29  creating first replica of object glb.
(DRM) 1994/03/17.14:11:30  object glb created.
(DRM) 1994/03/17.15:58:13  opening replica.
(DRM) 1994/03/17.15:58:13  replica of object glb opened.
(DRM) 1994/03/17.16:05:07  opening replica.
(DRM) 1994/03/17.16:05:08  replica of object glb opened.
(DRM) 1994/03/18.16:26:26  opening replica.
```

The data replication manager (DRM), which is running on the same process as the **glbd** daemon, opens the **/etc/ncs/glb.e** database each time the **glbd** daemon is started. The **glb.e** database contains information about servers, vendors and products within the cell. The DRM also uses the file **/etc/ncs/glb.p** (propagation queue) to propagate updates of the

glb.e database to all **glbd** replicas within the cell. This is done by default every 15 minutes and cannot be configured.

If the **glb.e** database is missing, any attempt to start the **glbd** daemon fails, and the following entry is made in the **glb_log** file:

```
(DRM) 1994/03/18.16:26:26  [1c03001e] unable to open replica of
object 000000000000.00.00.00.00.00.00.00.00.
?(GLB) cannot open replica – file io error (network computing
system/DRM)
(GLB) exiting
```

Note: When **glbd** replicas are used, all replicas are hierarchically equal. There is no master/slave concept used.

Tips on Managing iFOR/LS License Servers

Tip 1: Keep Copies of the License Form

It is strongly recommended to keep copies of the license forms received from the license provider in a secure place and to give system administrators access to these copies. This is the fastest way to recover licenses if there is no license-database backup accessible.

The information provided in the license form is needed when the **ls_admin** tool is used to recreate the **lic_db** iFOR/LS database.

Tip 2: Vendor Name and Vendor ID

To make the system administrator's job easier, the **ls_admin** tool provides support for literal-string vendor and product identification. Administrators may refer to vendors and products using literal strings. Internally, the NCS and iFOR/LS daemons use 16-byte alphanumeric identifiers (UUID) when referring to vendors and products registered in the databases.

For example, administrators may use the vendor-name literal string to select a specific vendor, when iFOR/LS needs the vendor id. **ls_admin** does not check for consistency between vendor name and vendor id. For this reason administrators should carefully read the system messages whenever **ls_admin** is used and be aware of possible erroneous handling. The following scenario may occur:

The license form has these entries:

```
Vendor name:          inti
Vendor id:            5242378dbf8d.02.c0.09.c8.93.00.00.00
Vendor password:      uprfdse5rkz4s

Product name:         CATIA.MECH FBD–AIX/6000
Product password:     7uad7duj8d96ci54cje4p9q68pns
Product:              410
```

And:

The administrator uses following command to add the vendor to the iFOR/LS database:

```
# ls_admin –a –v ”sol” 5242378dbf8d.02.c0.09.c8.93.00.00.00
uprfdse5rkz4s
```

After this, any attempt to add the vendor name `inti` using the information provided in the license form will fail with the following error message:

```
# ls_admin –a –v ”inti” 5242378dbf8d.02.c0.09.c8.93.00.00.00
uprfdse5rkz4s

LS_ADMIN Version 2.0.1 (GRI 1.1.2) IBM_AIX
(c) Copyright 1991,1992,1993,1994, Hewlett–Packard Company, All
Rights Reserved
(c) Copyright 1991,1992,1993,1994, Gradient Technologies Inc.,
All Rights Reserved
?(ls_admin) Duplicate vendor id (network license server/server)
```

This is an error because the vendor id has already been added to the database although the vendor name `sol` was used instead of the name `inti`.

Warning: To avoid errors that may lead to failures, especially when recovery is necessary, it is recommended that the vendor and product names supplied with the license form be used.

If in the same scenario, the user assumes not only that the vendor ID already exists, but also that the vendor name already exists, any attempt to add a product for this vendor using the entries in the license form will fail.

```
# ls_admin –a –p ”inti” ”CATIA.MECH FBD–AIX/6000”
7uad7duj8d96ci54cje4p9q68pns 410

LS_ADMIN Version 2.0.1 (GRI 1.1.2) IBM_AIX
(c) Copyright 1991,1992,1993,1994, Hewlett–Packard Company, All
Rights Reserved
(c) Copyright 1991,1992,1993,1994, Gradient Technologies Inc.,
All Rights Reserved
?(ls_admin) No such vendor (network license server/server)
```

When adding the product, the vendor name rather then the vendor ID is supplied. iFOR/LS uses the vendor name to find the vendor ID. Since the specified vendor name does not exist in the database, the error message `No such vendor` is displayed.

Administrators may need a long time to find the error, especially if there is no management plan for adding and deleting vendors and products to the network; even more if there is no graphical user interface available.

Solution:

Use the **ls_admin –n** *hostname* **–s –v** command to list all vendors at the specified server, delete the vendor, and add the vendor again using the information provided in the license form.

Tip 3: netlsd Failure to Communicate with glbd

Generally iFOR/LS servers are also configured as **glbd** servers. If the UUID stored in the file **/etc/ncs/glb_obj.txt** on such a server is changed, the **glbd** daemon continues to use the old UUID even after the **glbd** daemon is stopped and restarted. The communication between the **glbd** daemon and the **netlsd** daemon will fail.

For an example, observe the following scenario on the server `rouse`:

```
Cell UUID:
       65d6f8f6471e.02.09.03.01.45.00.00.00

Content of the /etc/ncs/glb_obj.txt
       657cab79f66f.02.81.23.1c.51.00.00.00
```

The **ls_tv** command displays the following error message:

```
# ls_tv

LS_TV Version 2.0.1 (GRI 1.1.2) IBM_AIX –– NetLS Test and
Verification Tool
(c) Copyright 1991,1992,1993,1994, Hewlett–Packard Company, All
Rights Reserved
(c) Copyright 1991,1992,1993,1994, Gradient Technologies Inc.,
All Rights Reserved
?(ls_tv) init: No servers available for this vendor (network
license server/library)
```

The above failure will occur even if there is a **glbd** replica on another node in the cell. The UUID is a 16-byte alphanumeric string and is hard to remember; therefore it is recommended that a copy of the current valid UUID be kept in a secure place.

The **lb_find** command is still able to communicate with the **glbd** daemon and displays a message similar to the following:

```
# lb_find

sent to broadcast address 9.3.1.255
waiting for replies
received response from glb daemon at
ip:rouse.itsc.austin.ibm.com(9.3.1.69) port 1765......
replicatable   ip:rouse.itsc.austin.ibm.com   alternate_2
65d6f8f6471e.02.09.03.01.45.00.00.00
```

If the change to the file **/etc/ncs/glb_obj.txt** was made without the administrator’s awareness, it is unlikely that he will compare the displayed UUID with the UUID currently stored in the **glb_obj.txt** file.

Solution:

The problem may be solved with the following steps:

1. Stop all running NCS and iFOR/LS daemons.
2. Remove the **/tmp/llbdbase.dat** file.
3. Write the correct UUID into the **/etc/ncs/glb_obj.txt** file. You can use the following command to create the correct **glb_obj.txt** file:

   ```
   # /etc/ncs/lb_find 2> /dev/null | grep HostName | awk ’{print
   $4}’ > /etc/ncs/glb_obj.txt
   ```
4. Restart the **llbd** daemon.
5. Restart the **glbd** daemon.
6. Restart the **netlsd** daemon.

Tip 4: Nodelock File Permissions

Ensure that the following file permissions and ownerships are assigned:

- To the directory **/usr/lib/netls/conf**

  ```
  drwxrwxrwx   2 root     bin
  ```

- To the file **/usr/lib/netls/conf/nodelock**

  ```
  –rwxr–xr–x   1 root     system
  ```

Tip 5: Finding Servers in Other Subnets

The **lb_find** command uses the RPC broadcasting mechanism to find GLB servers in the network. Because RPC broadcasting is limited to the local network, GLB servers in different physical subnets and servers that do not support broadcasting cannot be located. To solve this problem the **glb_site.txt** file can be used to directly contact GLB servers, even across the internet (gateways and routers). This will work only if the GLB server is a member of the **default** cell or if the local machine joins the GLB server's cell.

Solution:

If the GLB server is in an **alternate** cell, the server's **glb_obj.txt** file must be copied to the local machine.

Tip 6: Moving from the Default to an Alternate Cell

Most system administrators choose the default cell for their configuration because it seems to be easier to configure and administer. But, in the case of growing networks and increasing numbers of connected workstations, the need for partitioning into different NCS cells may arise.

Moving machines to a different cell can be done easily if planned carefully. The detailed plan should include:

- Host names and IP addresses of the workstations planned to join the new cell
- GLB server names
- iFOR/LS server names
- Contents of the required **/etc/ncs/glb_site.txt** files if the cell spans several physical subnets

The step-by-step description below includes a sample planning sheet. The following steps must be performed to move machines to a new cell:

1. Generate the new cell's UUID.

   This can be done using the **/etc/ncs/uuid_gen** tool on any host where NCS 1.5.1 is installed. Only one UUID needs to be generated; it will be used by all machines joining the cell. You can use the following command to store the output into a file:

   ```
   # /etc/ncs/uuid_gen > /tmp/newcell.uuid
   ```

2. Fill in the planning sheet

   The following table can be used:

Hostname	IP–Address	glbd	netlsd	glb_site.txt	cell–uuid (glb_obj.txt)
mitc53h	9.39.0.207	y	n	ip:#9.39.0.207 ip:#9.39.0.195	333b91c50000.0d.00.00.87.84.00.00.00
cadrs1	9.39.0.195	n	y	ip:#9.39.0.207	
strider	9.39.1.92	y	y	ip:#9.39.1.92 ip:#129.35.28.82	665daa193e2d.02.09.03.01.5c.00.00.00
balrog	9.39.1.81	n	n	ip:#129.35.28.82	
gandalf	129.35.28.82	y	y	ip:#129.35.28.82 ip:#9.39.1.92	

3. Stop all NCS and iFOR/LS daemons using the **stopsrc** command, and verify the stop using **lssrc.**
4. Delete all static NCS databases files.

   The following files must be deleted on all systems planned to join the new cell:

   – **/etc/ncs/glb.e**

   – **/etc/ncs/glb.p**

   – **/etc/ncs/glb_log**

   – **/tmp/llbdbase.dat**

   The GLB database file **glb.e** cannot be used within the new cell because it contains the old cell's UUID.
5. Create the cell definition file **/etc/ncs/glb_obj.txt**

   The contents of the definition file /**etc/ncs/glb_obj.txt** must be the same on all machines joining the new cell. Create this file as a copy of the output file produced in step 1 (**/tmp/newcell.uuid**). To avoid typing errors use the **cp** and **rcp** commands instead of editing the file. The **glb_obj.txt** file permissions should be –rw–r––r   (644).
6. Verify the contents of the **/etc/ncs/glb_site.txt** files.

   If the newly created cell spans several networks, the **glb_site.txt** file may be needed. Furthermore, some NCS client applications may need the older version **glb_sites**, especially RLM-enabled applications. Use the planning sheet to verify these files point to the selected GLB servers.
7. Start the **llbd** daemon on the system where the first GLB database will be created.

   Choose the machine where the first GLB database replica should be created and use the following command to start the **llbd** daemon:

   ```
   root@strider# startsrc –s llbd
   ```

   ```
   0513–059 The llbd Subsystem has been started. Subsystem PID is
   12906
   ```
8. Create the first GLB database in the new cell.

   The following command will create the first GLB database **glb.e** and the propagation queue **glb.p** in the new cell:

   ```
   root@strider# startsrc –s glbd –a ”–create –first –family ip”
   ```

   After using the command, check for creation of **/etc/ncs/glb.e** and **/etc/ncs/glb.p**, and use the **lssrc** command to verify the **glbd** daemon is active. If the **glbd** daemon is not active, the creation of the GLB database was not successful, even if the **glb.e** and **glb.p** files have been created. To trace the problem, restart the **glbd** daemon but without using the System Resource Controller (SRC). Enter the command:

   ```
   root@balrog# glbd –create –from ip:RemoteHost
   ```

   and read the error messages.
9. Create the GLB database replicas.

   If replicas of the GLB database should be running on other servers, start the **llbd** daemon, and use the following command:

   ```
   root@balrog# startsrc –s glbd –a ”–create –from ip:RemoteHost”
   ```

Where *RemoteHost* is the server hostname and the first GLB database was created; in this example `strider`.

10. Check the NCS communication.

    The NCS communication can be tested using the **lb_find** or **lb_admin** commands. Refer to *AIX Version 4.1 iFOR/LS System Management Guide*, for more information about these commands.

11. Start the iFOR/LS server daemon.

    Ensure the NCS communication is working; then the iFOR/LS server daemon **netlsd** can be started using the following command:

    ```
    # startsrc –s netlsd
    ```

12. Test the iFOR/LS-NCS communication.

    The **ls_tv** command can be used to test if NCS can locate iFOR/LS servers:

    ```
    # ls_tv

    LS_TV Version 2.0.1 (GRI 1.1.2) IBM_AIX –– NetLS Test and
    Verification Tool
    (c) Copyright 1991,1992,1993,1994, Hewlett-Packard Company, All
    Rights Reserved
    Completed license transaction on node 1093710 running NetLS
    Active NetLS Servers:
    strider.itsc.austin.ibm.com (IBM/AIX3.2) running NetLS
    ```

13. Check for iFOR/LS database access.

    To test if **netlsd** can access the information in the iFOR/LS database **lic_db**, the **ls_stat**, or **ls_admin** commands can be used. Refer to *AIX Version 4.1 iFOR/LS System Management Guide*, for more information about the iFOR/LS administrative commands.

14. Restart the **llbd** daemon on the client machines.

iFOR/LS Debugging and Problem Determination

This section provides a checklist for debugging and problem determination. Refer to “Configuring an iFOR/LS Environment” on page 4-1 and to the section on troubleshooting in the *AIX Version 4.1 iFOR/LS System Management Guide*, for more information.

1. Is the **netlsd** daemon running?

    a. Does the file **/etc/inittab** contain the **netlsd** startup procedure?

    b. Is the **/etc/rc.netls** file in good shape?

    c. Are the database files of the **netlsd** still in place? Check the files **/usr/lib/netls/conf/lic_db** and **/usr/lib/netls/conf/cur_db**.

    d. Use the **ls_tv** command to verify which license servers are working.

    e. NCS and iFOR/LS are based on UDP. In a very highly loaded network, UDP connections may receive timeouts before data is delivered. There is nothing wrong with this behavior. Reduce the total network load.

2. Is the NCS system up and running?

a. Does the file **/etc/inittab** contains the **rc.ncs** startup procedure?

b. Is the **/etc/rc.ncs** file in good shape ? The first daemon started there must be **llbd**. The second daemon should be the **glbd**.

c. Is the **llbd** daemon running?

d. Is the **glbd** daemon running?

e. Are the system clocks still in sync? This is mandatory to make NCS work. Use the **setclock** command to sync all systems for a short-term solution. It is recommended to implement external time providers and a distributed time service on the network.

f. Is the **/etc/ncs/glb_obj.txt** file the same on all NCS cell member hosts?

g. Is the **/etc/ncs/glb_site.txt** file still pointing to a valid GLB host?

h. Do the GLB database files still exist? Check particularly for the existence of the files **/etc/ncs/glb.e** and **/etc/ncs/glb.p**.

i. Was **llbd** able to create its temporary file **/tmp/llbdbase.dat**?

j. Use the **ncs_test.sh** script to test set-up and run-time parameters (see Chapter 3 and Appendix B for more information).

3. Is the TCP/IP system up and running ?

a. Has someone changed IP addresses or network interfaces ? The global location broker database may not reflect the changes someone has done to the real network. Check the NCS config files, and restart the NCS and iFOR/LS daemons.

b. Can you reach other systems in the network?

c. Is your routing setup definition valid? The **netstat** command shows the local definition. To see the actual hubs use the **traceroute** command.

d. Is name resolution working? Name resolution is very often the reason for long startup times or many sorts of problems in large networks. Use the Domain Name System (DNS), and spend some time developing good layout.

e. Is the MTU size equal on all hosts?

f. Is the Token-Ring speed equal on all hosts?

4. Is the hardware OK?

a. Has someone changed the CPU planar of your iFOR/LS server? The licenses on the iFOR/LS server are tied to the CPU ID, which changes after a CPU planar swap.

b. Are the cables still where they should be? Check the cables.

c. Have you reached the Ethernet length limitations on your LAN ? Most LANs are growing all the time but there are limits. Be aware of them.

d. Has someone enabled a security feature on a router? Some routers allow enabling of security features. It is possible to block certain TCP/IP ports. The **llbd** program is running on port 135. The **glbd** and **netlsd** programs are using run-time-assigned ports whose port numbers are greater than 1024.

iFOR/LS Backup and Recovery Examples

Because the physical breakdown of license servers may have a potentially severe impact on production, a backup and recovery strategy for iFOR/LS should be in place for customers when they start using the iFOR/LS license management.

License keys are generated based on the hardware processor ID. Thus, today it is not possible to have an automatic rollover in case of a physical license-server crash. Moreover, if there is a need to change the system planar, all current available licenses are lost because the CPU ID changes with the new system planar.

This section describes how to be prepared in the following cases:

- Definitions and database files are corrupted.
- The operating system has to be recovered or reinstalled from scratch.

General Backup and Recovery iFOR/LS Strategy

The backup and recovery of iFOR/LS-managed data can be divided into two main parts:

- Backup and recovery of the NCS definition and database files
- Backup and recovery of the iFOR/LS database files

The minimum backup activity expected of the system administrator is to keep the license form received from the license provider in a secure place.

System administrators may also use shell scripts that call the **ls_admin** administrative tool to update the license server database. It is recommended that these shells be saved along with the definition and database files.

Causes for Corrupted Definition or Database Files

There are many situations that can cause the definition or database files to get corrupted. The most common causes may be split into two groups:

- NCS-related issues
- iFOR/LS-related issues

NCS-related issues
The NCS definition and database files are static and linked to network addresses. For this reason, changing definitions or adapters within the network may lead to connection errors. The following files are used by the **llbd** and **glbd** during startup to establish connection with the network and to register objects.

- The **llbd** daemon uses the **/tmp/llbdbase.dat** file
- The **glbd** daemon uses the **/etc/ncs/glb.e** database

iFOR/LS-related issues
Since iFOR/LS uses the database files dynamically, any disk-related problems like the following may cause the database files to become corrupted:

- Hardware failures (media surface errors)
- File-system problems (for example, file system full)

– Synchronization errors during write of data (that is, loss of electrical power)

When an iFOR/LS database is corrupted, problem determination procedures should be performed after the database is recovered to find out the real cause of the problem.

Backup Strategy

Since the contents of the definition and database files used by NCS and iFOR/LS are changed only by defined administrative commands and tools, there is no need to set up an automated periodic backup strategy. The contents of the dynamically updated files **glb.p** and **cur_db** are rebuilt whenever the **glbd** and **netlsd** daemons, respectively, are started.

Backup and Recovery of the NCS Definition and Database Files

The following files should be saved on:

NCS clients:

- **/etc/ncs/glb_site.txt**
- **/etc/ncs/glb_obj.txt**

NCS GLB servers:

- **/etc/ncs/glb_site.txt**
- **/etc/ncs/glb_obj.txt**
- **/etc/ncs/glb.e**
- **/etc/ncs/glb.p**

The preceding files should be saved after creation and whenever changed. The administrative commands and tools used to change these files are:

drm_admin	Administers servers such as GLB, based on the Data Replication Manager (DRM). When the subcommands **merge**, or **merge_all** are used, the contents of the **glb.e** database are affected.
lb_admin	Administers the registrations of NCS-based servers in Global Location Broker (GLB) or Local Location Broker (LLB) databases. When the subcommands **add**, **delete**, **register**, or **unregister** are used, the contents of the **glb.e** database are affected.
uuid_gen	Generates a Universal Unique Identifier (UUID). This command is used to generate the contents of the **glb_obj.txt** definition file.
glbd	When used with the **create** option, either creates new **glb.e** and **glb.p** databases or copies them remotely from a referenced remote **glbd** depending on the value of the second option **first** or **from** respectively.

The **glb_site.txt** file is updated manually.

Backup and Recovery of the iFOR/LS Database Files

On iFOR/LS servers, the following database files located in the **/usr/lib/netls/conf** directory should be saved:

- **lic_db**
- **nodelock**
- **products**

The preceding files should be saved after creation and whenever changed. The administrative commands and tools used to change these files are:

ls_admin	Administer vendor, product, and license information. This administrative tool can be used to change the contents of the **lic_db** database.
ls_dpass	Used to generate licenses from compound licenses. This administrative tool can be used to change the contents of the products database.

The **nodelock** file is updated manually.

Note: Save the following configuration and startup files if you have changed them:

- **netls_config**
- **netls_first_time**
- **/etc/rc.ncs**
- **/etc/rc.netls**

Backup Implementation Example

The following sample shell script can be used to back up the relevant NCS and iFOR/LS definition and database files. System administrators can run the shell script whenever changes to the files as explained in "General Backup and Recovery iFOR/LS Strategy" on page 5-18 are performed.

The shell script output is the backup format file **/tmp/iforls_bak_TODAY_SERVER**, where TODAY is the current date and SERVER is the iFOR/LS server at which the script has been run. The output file destination can be changed easily to create the backup file on a diskette or any other device.

The shell script saves files on NCS clients, NCS servers, iFOR/LS servers, or a combination of them.

Backup Shell Script Example

```
#!/bin/ksh
# @(#)  Version 1.1  db_back.sh   Example  03/28/94
#=================================================================#
#   Script :  db_back.sh                                          #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer             #
#   Date   :  94/03/28                                            #
#   Update :  94/__/__                                            #
#   Info   :  This script saves the relevant NCS and              #
#             iFOR/LS definition and database files.              #
#                                                                 #
#   Input  :  NONE                                                #
#                                                                 #
#   Output :  file /tmp/iforls_bak_$TODAY_$SERVER as backup       #
#             format file                                         #
#-----------------------------------------------------------------#
set –x
BACKUP=”/usr/sbin/backup”              #  pathname to backup command
PROG=”$(basename $0)”                  #
#-----------------------------------------------------------------#
#  This is the usage information.                                 #
#-----------------------------------------------------------------#
usage()
{
 echo ”\nUsage: $PROG”
 echo ”\n\tDefault backup of NCS and iFOR/LS related definitions”
 echo ”\tand database files\n”
 exit 1
}
#------------------------------------------------------------------#
#  Test for valid input or display usage() information.           #
#------------------------------------------------------------------#
if [ ”${1}” = ’h’ –o ”${1}” = ’?’ ]; then
   usage
fi
#------------------------------------------------------------------#
#  Prepare the variables.                                          #
#------------------------------------------------------------------#
integer RET
SERVER=”$(hostname)”
TODAY=$(date +%m_%d_%y)
INF=”/tmp/.ibackup.$$”                     #  temp. input file for
backup
OUTF=”/tmp/iforls_bak_${TODAY}_${SERVER}” #  database backup file
#------------------------------------------------------------------#
#                                                                  #
#  This is the shell script’s master loop. It saves all           #
#  NCS and iFOR/LS definition and database files                   #
#  using the backup command.                                       #
#                                                                  #
#------------------------------------------------------------------#
rm $INF 1>/dev/null 2>/dev/null
#------------------------------------------------------------------#
 for FILENAME in /etc/ncs/glb_site.txt \
                 /etc/ncs/glb_obj.txt  \
                 /etc/ncs/glb.e        \
                 /etc/ncs/glb.p        \
                 /usr/lib/netls/conf/lic_db           \
                 /usr/lib/netls/conf/nodelock         \
                 /usr/lib/netls/conf/products         \
                 /usr/lib/netls/conf/netls_config     \
                 /usr/lib/netls/conf/netls_first_time \
                 /etc/rc.ncs \
                 /etc/rc.nets

  do
   if [ –f ”$FILENAME” ]
   then
    echo ”$FILENAME” >> $INF
   fi
  done
#------------------------------------------------------------------#
RET=$(${BACKUP} –ivqf ”${OUTF}” < ”${INF}” 1>/dev/null 2>/dev/null)
#------------------------------------------------------------------#
rm $INF 1>/dev/null 2>/dev/null
#------------------------------------------------------------------#
# EOF                                                              #
#------------------------------------------------------------------#
exit ${RET}
```

Recovery Procedure

In cases where the NCS or iFOR/LS definitions or database files are corrupted, the following recovery procedure flow should be followed:

1. Check the backup file for readability.
2. Stop all running NCS and iFOR/LS deamons.
3. Delete the following files:
   - **/tmp/llbdbase.dat**
   - **/etc/ncs/glb_obj.txt**
   - **/etc/ncs/glb_site.txt**
   - **/etc/ncs/glb_log**
   - **/etc/ncs/glb.e**
   - **/etc/ncs/glb.p**
   - **/usr/lib/netls/conf/lic_db**
   - **/usr/lib/netls/conf/lic_db.bak**
   - **/usr/lib/netls/conf/cur_db**
   - **/usr/lib/netls/conf/nodelock**
   - **/usr/lib/netls/conf/products**
   - **/usr/lib/netls/conf/products.bak**
4. Restore the deleted files from the backup media
5. Restart the NCS and iFOR/LS daemons in following order:

```
startsrc –s llbd
startsrc –s glbd
startsrc –s netlsd
```

Recovery Implementation Example

The following sample shell script implements the above explained recovery procedure flow and can be used to recover from NCS and iFOR/LS definition and database file-corruption failures.

The shell script may be used on NCS clients, NCS servers, iFOR/LS servers, or a combination of them. The following is the shell script flow:

1. Stops all active NCS and iFOR/LS daemons.
2. Stores a list of the current definition and database files in a temporary file.
3. Stores a list of the files on the backup media in a second temporary file.
4. Compares both temporary files. If they are not the same or if some files are missing from either the backup or the system configuration,

   then

   a. Displays the name of the files missing on the backup media, or the name of the files in the backup media missing on the current configuration, and asks the user if he wants to continue. Any answer other the **y** will cause the script to **terminate**.

   else

   b. Continues with step 5.
5. Removes the definition and database files and restores from backup media.

6. Restarts the NCS and iFOR/LS daemons.

Note: It is important to check if all current definition and database files to be replaced are also saved in the backup media. In case additional files have been created, this may have introduced a failure. There is no guarantee the system will behave as it used to when the backup file was created. Systems administrators should investigate this case more closely and start recovery procedures manually.

Recovery Shell Script Example

```
#!/bin/ksh
# @(#)  Version 1.1  db_recov.sh   Example  03/30/94
#==================================================================#
#   Script :  db_recov.sh                                          #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer              #
#   Date   :  94/03/28                                             #
#   Update :  94/03/30                                             #
#   Info   :  This script recovers the NCS and iFOR/LS definition #
#             database files from back–up media                    #
#                                                                  #
#   Input  :  filename of the backup format file                   #
#                                                                  #
#   Output :  information about performed steps                    #
#                                                                  #
#------------------------------------------------------------------#
RESTORE=”/usr/sbin/restore”           #  pathname to restore command
LSSRC=”/usr/bin/lssrc”                #  pathname to lssrc   command
REMOVE=”/usr/bin/rm”                  #  pathname to rm      command
AWK=”/usr/bin/awk”                    #  pathname to awk     command
PROG=”$(basename $0)”                 #
#------------------------------------------------------------------#
#  Cleanup procedure                                               #
#------------------------------------------------------------------#
cleanup()
{
 TST=$(${REMOVE} ”${INF1}” ”${INF2}” 1>/dev/null 2>/dev/null)
 exit –09
}
#------------------------------------------------------------------#
#  Usage information                                               #
#------------------------------------------------------------------#
usage()
{
 echo ”\nUsage: $PROG”
 echo ”\n\tDefault recovery of NCS and iFOR/LS related definitions”
 echo ”\tand database files\n”
 cleanup
}
#------------------------------------------------------------------#
#  Test for valid input or display usage() information             #
#------------------------------------------------------------------#
if [ ”${1}” = ’h’ –o ”${1}” = ’?’ ]; then
   usage
fi
#------------------------------------------------------------------#
#  Prepare the variables                                           #
#------------------------------------------------------------------#
integer VAL1 VAL2
```

```
SERVER="$(hostname)"
TODAY=$(date +%m_%d_%y)
RESTOREF="${1:-"/home/netlstest/ok"}"      #  database back-up media
INF1="/tmp/.irecov.$$"                     #  temp. scratch file
INF2="/tmp/.inventory.$$"                  #  temp. scratch file
NCS="/etc/rc.ncs"                          #  ncs startup file
NETLSD="/etc/rc.netls"                     #  netlsd startup file
#-----------------------------------------------------------------#
#                                                                 #
#  This is the shell script's master loop. It performs the        #
#  following steps:                                               #
#                                                                 #
#  1.) Stops all running NCS and iFOR/LS daemons                  #
#  2.) Writes a list of the current definition and database files #
#      into a temporary file                                      #
#  3.) Writes a list of the files saved in the backup media into  #
#      a temporary file                                           #
#  4.) Compares both temporary files. If the contents are not     #
#      the same, a list of the missing files is displayed, and the#
#      user is asked whether to continue                          #
#  5.) If the user wants to continue, the definition and database #
#      files are removed from the system,and the backup files are #
#      restored                                                   #
#  6.) Restarts the NCS and iFOR/LS daemons                       #
#                                                                 #
#                                                                 #
#-----------------------------------------------------------------#
#                                                                 #
#  1.) Stop all running NCS and iFOR/LS deamons                   #
#                                                                 #
#-----------------------------------------------------------------#
echo "\tsearching for running NCS and iFOR/LS daemons on server $SERVER"
for TESTSRC in netlsd \
               glbd   \
               llbd
do
 RET=$(${LSSRC} -s ${TESTSRC})
 TST=$(echo ${RET} | ${AWK} ' { print $8}')
 if [ "${TST}" = "active" ]
  then
   echo "\tstopping the ${TESTSRC} subsystem"
   stopsrc -s ${TESTSRC}
 fi
done
sleep 5  # wait for the netlsd to stop
#-----------------------------------------------------------------#
#  2.) Write a list of the current definition and database files  #
#      into a temporary file                                      #
#-----------------------------------------------------------------#
echo
"\t******************************************************************"
echo "\tfollowing definition and database files have been found"
for FILENAME in /etc/ncs/glb_site.txt \
                 /etc/ncs/glb_obj.txt  \
                 /etc/ncs/glb.e        \
                 /etc/ncs/glb.p        \
                 /usr/lib/netls/conf/lic_db            \
                 /usr/lib/netls/conf/nodelock          \
                 /usr/lib/netls/conf/products          \
                 /usr/lib/netls/conf/netls_config     \
                 /usr/lib/netls/conf/netls_first_time \
                 /etc/rc.ncs \
                 /etc/rc.nets

do
 if [ –f ${FILENAME} ]
   then
   echo ”$FILENAME” >> ${INF2}
 fi
done
echo ”\t...... o.k.
.........................................................”
echo
”\t*************************************************************”
#------------------------------------------------------------------#
#  3.) Write a list of the files saved in the backup media into   #
#      a temporary file                                            #
#------------------------------------------------------------------#
RET=$(${RESTORE} –Tqf ”${RESTOREF}” 1>${INF1})
#------------------------------------------------------------------#
#  4.) Compare both temporary files. If the contents are not      #
#      the same, display a list of the missing files, and ask the #
#      user whether to continue                                    #
#------------------------------------------------------------------#
echo ”\tcomparing current system files with files available on backup media”
VAL1=$(wc –l ${INF1} | ${AWK} ’ { print $1}’)
VAL2=$(wc –l ${INF2} | ${AWK} ’ { print $1}’)
#
if [ ${VAL1} –ne ${VAL2} ]
then
        if [ ${VAL1} –lt ${VAL2} ]
        then
          cat $INF2 | \
          while read INVENSTRING
          do
                RET1=$(grep ${INVENSTRING} ${INF1})
                if [ ”${RET1}” = ”” ]
                then
                  INMISS=”${INVENSTRING}\n\t${INMISS}”
                fi
          done
          if [ ”${INMISS}” != ”” ]
          then
    echo ”\n\tfollowing files are missing on your backup media”
    echo ”\n\t$INMISS”
    echo ”\tcontinuing may lead to unpredictable results”
    echo ”\tpls check what has been changed on the system configuration\n”
          fi
        else
          cat $INF1 | \
          while read RESTORESTRING
          do
                RET2=$(grep ${RESTORESTRING} ${INF2})
                if [ ”${RET2}” = ”” ]
                then
                  RESMISS=”${RESTORESTRING}\n\t${RESMISS}”
                fi
          done
          if [ ”${RESMISS}” != ”” ]
          then
   echo ”\n\tfollowing files are missing on your current system”
   echo ”\n\t$RESMISS”
   echo ”\tcontinuing may lead to unpredictable results”
   echo ”\tpls check what has been changed on the system configuration\n”
          fi
        fi
        echo ”\nDo you still want to continue? y/n: ”
        read ANS
        if [ ”$ANS” != ”y” ]
        then
          cleanup
        fi
fi
echo ”\t...... o.k.
.........................................................”
echo
”\t*****************************************************************”
#------------------------------------------------------------------#
#  5.) Remove database and definition files                        #
#------------------------------------------------------------------#
cat $INF2 | \
 while read FILENAME
 do
  echo ”\tremoving $FILENAME”
  RET=$(${REMOVE} ${FILENAME} 1>/dev/null 2>/dev/null)
 done
${REMOVE} /tmp/llbdbase.dat
echo ”\t...... o.k.
.........................................................”
echo
”\t*****************************************************************”
#------------------------------------------------------------------#
#     Restore from backup media.                                   #
#------------------------------------------------------------------#
echo ”\tstarting the restore process”
RET=$(${RESTORE} –xqvf ”${RESTOREF}”)
echo ”\t...... o.k.
.........................................................”
echo
”\t*****************************************************************”
#------------------------------------------------------------------#
#  6.) Restart all NCS and iFOR/LS daemons                         #
#------------------------------------------------------------------#
echo ”\tstarting the NCS and iFOR/LS daemons”
ksh ${NCS}
ksh ${NETLSD}
#------------------------------------------------------------------#
# EOF                                                              #
#------------------------------------------------------------------#
cleanup
```

Chapter 6. RLM and iFOR/LS Coexistence

iFOR/LS servers can coexist in the same NCS default cell as Resource License Manager (RLM) servers. Client applications enabled for using RLM servers may have their license requests satisfied by iFOR/LS servers.

Furthermore, the RLM server daemon **rlmd** can communicate with the replicatable GLB daemon **glbd** and have its services registered within the **glbd**'s database.

glbd and nrglbd Coexistence

The NCS Version 1.1 GLB daemon **nrglbd** can coexist in the same NCS cell as the NCS Version 1.5.1 GLB daemon **glbd** provided this is the default cell and the **glbd** database is not a replica of the **nrglbd** database, because **nrglbd** does not support database replication.

Although **nrglbd** and **glbd** coexistence is possible within the default cell, it is not absolutely necessary because **glbd** can fully substitute for **nrglbd** and communicate with both RLM and iFOR/LS servers.

Note: It is recommended to use the replicatable GLB daemon **glbd** whenever available.

Coexistence Restrictions

The following restrictions must be considered:

1. NCS Version 1.1 doesn't support:
   - Replicatable global location brokers
   - Cell concepts.
2. iFOR/LS cannot manage RLM databases and vice versa. This means RLM cannot manage the content of the **/usr/lib/netls/conf/lic_db** and **/usr/lib/netls/conf/cur_db**, neither can iFOR/LS manage the **/usr/lpp/rlm/db/lic_db** database files.
3. The RLM daemon (**rlmd**) and iFOR/LS daemon (**netlsd**) cannot run together on the same machine because **rlmd** is based on an old version of iFOR/LS and cannot share its resources with the **netlsd** daemon.
4. RLM is an NCS Version 1.1 based application that can only communicate within the default cell. Trying to start the RLM daemon with **/etc/ncs/glb_obj.txt** pointing to an alternate cell will fail as shown in the following example:

```
# /usr/lpp/rlm/bin/rlmd –v

rlmd host id is 1093710
sizeof(prod_info_t) = 80
sizeof(lic_entry_t) = 176
sizeof(v_hdr_t) = 72
my_ntype = 6
entering: init_vnd_info()
?(rlmd) Unable to register with locating broker communications
failure
(network computing system/RPC runtime)
Terminating...
```

5. To communicate with a GLB server in another network, the RLM server daemon **rlmd** uses the NCS Version 1.1 **/etc/ncs/glb_sites** file, which has been replaced in NCS Version 1.5.1 by the **/etc/ncs/glb_site.txt** file. Also some client applications enabled for RLM have the call to the **glb_sites** file hardcoded (refer to the application documentation to find out if your application needs the **glb_sites** file). Any attempt to use the **glb_site.txt** file instead of the **glb_sites** file will fail. The error message for the **rlmd** daemon is as follows:

   ```
   #/usr/lpp/rlm/bin/rlmd –v

   rlmd host id is 1093710
   sizeof(prod_info_t) = 80
   sizeof(lic_entry_t) = 176
   sizeof(v_hdr_t) = 72
   my_ntype = 6
   entering: init_vnd_info()
   ?(rlmd) Unable to register with locating broker communications
   failure
   (network computing system/RPC runtime)
   Terminating...
   ```

   The error message for the application CATIA Solutions Version 4 and IGES Version 2 is:

   ```
   [1d020006] nlslib_no_svrs_fnd
   ```

Note: The **glb_sites** and the **glb_site.txt** files can coexist on the same machine.

Using Administrative Commands

RLM is based on an earlier version of iFOR/LS (NetLS Version 1.1), as displayed with the lookup **lb_admin** tool subcommand **lookup** (“NLS[1.1]”...), while iFOR/LS is based on NetLS Version 2.0.

As mentioned in “Coexistence Restrictions,” iFOR/LS and RLM can coexist within the same cell, but they cannot administer each other’s license databases. The following administrative tools and commands are able to locate each other’s databases, but they fail when they attempt to open them:

- **ls_admin**
- **ls_stat**
- **rlmstat**
- **rlmadmin**

Refer to the *AIX Version 4.1 iFOR/LS System Management Guide* for more information about the iFOR/LS administrative commands and to the RLM documentation for the RLM commands.

For example, when called with the option **–i** to display all installed licenses on the RLM server `mitc53h`, the **ls_stat** command displays the following error message:

```
# /etc/netls/ls_stat –i

?(ls_stat) Error at server mitc53h.ncs.mainz.ibm.com –
Wrong version of NetLS Server (network license server/server)
```

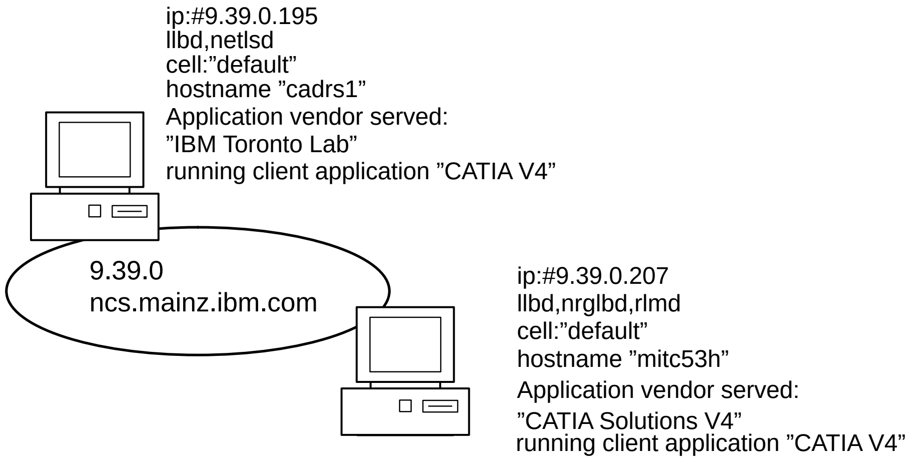
If the **rlmstat** command is used to display all installed licenses on the iFOR/LS server `cadrs1`, the following error message is displayed:

```
# /usr/lpp/rlm/bin/rlmstat –i

?(rlmstat) Error at server cadrs1.ncs.mainz.ibm.com status
1d01001a (network license server)
License Users
End of License Users
```

Coexistence Example

The following sample session describes the coexistence of two machines in the network, an iFOR/LS server and an RLM server. The information displayed by NCS and iFOR/LS is explained in detail for better understanding. The scenario is shown in the following figure and explained in the table below:



	iFOR/LS server	**RLM server**
Host name	cadrs1	mitc53h
IP address	9.39.0.195	9.39.0.207
Application vendor	IBM Toronto Lab, HP NetLS Test	CATIA Solutions V4
Application served		“CATIA V4”
Application running	“CATIA V4”	“CATIA V4”
Daemons	llbd, netlsd	llbd, nrglbd, rlmd
Site files	glb_sites, glb_site.txt	glb_sites
Cell	default	

Notes:

1. Since RLM does not support alternate cells, the default cell is used. The GLB daemon for this cell is the **nrglbd** running on `mitc53h,` but the behavior is the same if the **glbd** daemon were used instead.
2. CATIA V4 and **rlmd** are RLM-based, while **netlsd** is an iFOR/LS-based application.

3. Although it is not required because both hosts are in the same local network, the use of **glb_sites** and **glb_site.txt** is recommended to ensure the GLB services are provided by a specific host (especially when the default cell is used).

Server cadrs1 Example

1. On this server, both RLM-based (CATIA V4) and iFOR/LS-based (**netlsd**) applications are running; therefore both the **glb_sites** and the **glb_site.txt** files are needed (refer to "Configuring the glb_site.txt File" in Chapter 5 of *AIX Version 4.1 iFOR/LS System Management Guide* for more explanation about the **glb_site.txt** file). Both files contain the IP address of the host where the GLB daemon (**nrglbd**) is running, and both are used to contact the GLB server directly, avoiding the use of broadcasting. For example:

```
# cat /etc/ncs/glb_sites
ip:#9.39.0.207

# cat /etc/ncs/gl_site.txt
ip:#9.39.0.207
```

2. The **lb_find** command is used to display all hosts running GLB daemons in the default cell within this local network.

```
# /etc/ncs/lb_find

sent to broadcast address 9.39.0.255
waiting for replies
received response from glb daemon at
ip:mitc53h.ncs.mainz.ibm.com(9.39.0.207)port 1156.
.....
replicatable ip:mitc53h.ncs.mainz.ibm.com  default
333b91c50000.0d.00.00.87.84.00.00.00
```

There is one GLB daemon running in this cell. It is running on host `mitc53h`.

Note: The message says the daemon is *replicatable* although it is not. This is a reported NCS error.

3. The **lb_admin** command is used with the subcommand **lookup** to display the information stored in the GLB database managed by the **rmld** daemon on host `mitc53h`.

```
# lb_admin
lb_admin: us global
lb_admin: lookup

Data from GLB replica: ip:mitc53h.ncs.mainz.ibm.com
------------
   object = 5ecd2d88f406.02.09.27.00.cf.00.00.00
     type = 34275946a000.0d.00.00.05.55.00.00.00
interface = 34275946a000.0d.00.00.05.55.00.00.00
”NLS @ ip:mitc53h.ncs.mainz.ibm.com” @ ip:mitc53h.ncs.mainz.ibm.com[1160] global
------------
   object = 5ecd2d88f406.02.09.27.00.cf.00.00.00
     type = 3bd624ea7000.0d.00.00.80.9c.00.00.00
interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[1.1] @ ip:mitc53h.ncs.mainz.ibm.com” @ ip:mitc53h.ncs.mainz.ibm.com[1160] global
------------
   object = 5ecd2d88f406.02.09.27.00.cf.00.00.00
     type = 5242378dbf8d.02.c0.09.c8.93.00.00.00
interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[1.1]: CATIA SOLUTIONS V4” @ ip:mitc53h.ncs.mainz.ibm.com[1160] global
------------
   object = 64b711e53f49.02.09.27.00.c2.00.00.00
     type = 34275946a000.0d.00.00.05.55.00.00.00
interface = 34275946a000.0d.00.00.05.55.00.00.00
”NLS @ ip:#9.39.0.195” @ ip:cadrs1.ncs.mainz.ibm.com[1097] global
------------
   object = 64b711e53f49.02.09.27.00.c2.00.00.00
     type = 3bd624ea7000.0d.00.00.80.9c.00.00.00
interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0] @ ip:#9.39.0.195” @ ip:cadrs1.ncs.mainz.ibm.com[1097] global
------------
   object = 64b711e53f49.02.09.27.00.c2.00.00.00
     type = 4ca0fd5cf000.0d.00.02.1a.9a.00.00.00
interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0]: Hewlett–Packard NetLS Test” @ ip:cadrs1.ncs.mainz.ibm.com[1097] global
------------
   object = 64b711e53f49.02.09.27.00.c2.00.00.00
     type = 5fbee0ee6feb.02.09.15.0f.48.00.00.00
interface = 3bd624ea7000.0d.00.00.80.9c.00.00.00
”NLS[2.0]: IBM Toronto Lab” @ ip:cadrs1.ncs.mainz.ibm.com[1097] global
------------
```

The information displayed should be interpreted as follows:

– All information is provided by the GLB daemon running on `mitc53h` (“*Data from GLB replica: ip:mitc53h.ncs.mainz.ibm.com*”)

– The license server running on `mitc53h` is an RLM server (“NLS[1.1]”) while the license server running on `cadrs1` is an iFOR/LS server (“NLS[2.0]”).

– There is one application vendor (“CATIA SOLUTIONS V4”) managed by the RLM server on `mitc53h`, and there are two vendors (“Hewlett-Packard NetLS Test” and “IBM Toronto Lab”) managed by the iFOR/LS server on `cadrs1`.

– `1160` is the communication port used by the RLM server **rlmd** on host `mitc53h`, while `1097` is used by the iFOR/LS server **netlsd** on host `cadrs1`.

– “Global” means this object is registered by a GLB daemon running within the cell.

Refer to Chapter 1 for more information about the meaning of *object*, *type*, and *interface.*

4. The **ls_tv** command is used to display all available iFOR/LS servers in the cell:

```
# /etc/netls/ls_tv

LS_TV Version 2.0.1 (GRI 1.1.1e) IBM_AIX –– NetLS 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  1093710 running NetLS
Active NetLS Servers:
cadrs1.ncs.mainz.ibm.com (IBM/AIX 3.2) running NetLS
```

There is one iFOR/LS server running in this cell. It is running on host `cadrs1`.

5. The **ls_stat –a** command is used to display information about all concurrent license users.

```
# /etc/netls/ls_stat –a

LS_STAT Version 2.0.1 (GRI 1.1.1e) IBM_AIX
(c) Copyright 1991,1992,1993, Hewlett–Packard Company, All Rights
Reserved
(c) Copyright 1991,1992,1993, Gradient Technologies Inc., All
Rights   Reserved
?(ls_stat) Error at server mitc53h.ncs.mainz.ibm.com – Wrong
version of NetLS Server (network license server/server)
License Users
End of License Users
```

As stated in “Coexistence Restrictions,” iFOR/LS cannot manage the RLM-based **cur_db** database stored on host `mitc53h`. This is the reason for the error message displayed. The last two lines are from the iFOR/LS-based **cur_db** stored on host `cadrs1`. At the moment no licenses are in use or granted to any application, but the iFOR/LS database is initialized and accessible.

Server mitc53h Example

The **rlmstat –a** command is used to display information about all concurrent license users.

```
# /usr/lpp/rlm/bin/rlmstat –a

?(rlmstat) Error at server cadrs1.ncs.mainz.ibm.com status
1d01001a (network license server)
License Users
End of License Users
```

Similar to server `cadrs1`, step 5, RLM cannot manage the iFOR/LS-based **cur_db** stored on host `cadrs1`, and for this reason the error message is displayed. The last two lines are from the RLM-based **cur_db** stored on host `mitc53h`. At the moment, no licenses are in use or granted to any application, but the RLM database is initialized and accessible.

Migrating RLM Servers to iFOR/LS Servers

Migrating RLM-based license servers to iFOR/LS-based license servers is technically possible, since iFOR/LS is able to integrate, manage and serve license keys generated for RLM servers and is able to satisfy license requests from applications that require RLM license keys.

iFOR/LS servers cannot manage the RLM database **/usr/lpp/rlm/db/lic_db** because the structure is different from the structure of the iFOR/LS database **/usr/lib/netls/conf/lic_db**. For this reason, copying the **lic_db** database from the **/usr/lpp/rlm/db** into the **/usr/lib/netls/conf** directory does not help. The RLM license information must be integrated into the newly created iFOR/LS **lic_db** database using the **ls_admin** command.

Note: To add the RLM license information to the iFOR/LS **lic_db** database, the vendor password and the product password are required. This information cannot be retrieved from the RLM database; therefore, ensure that the license form provided by the software vendor is available for every product, or contact the software vendor to obtain this information before beginning migration. The license form looks similar to the following:

```
Product name . . . . . . : CATIA.MECH FBD–AIX/6000      1
Product number . . . . . : 5626FBD 0CE
Product version. . . . . : 410
Target id. . . . . . . . : 00018734
No. of Concurrent users. : 3
Passwords are valid From : 1994–03–10
To . . . . . . . . . . . : 1999–12–31
Vendor name. . . . . . . : Dassault Systemes
Vendor id. . . . . . . . : 5242378dbf8d.02.c0.09.c8.93.00.00.00
Vendor Password. . . . . : 69kw7w8zjbb2n
Product Password . . . . : p8fqnucgkyrf4mmpxjbn3j97u2
```

Step-by-Step Procedure

The following is a step-by-step procedure for migrating an RLM server to an iFOR/LS server:

1. Back up the following RLM files and databases:
   - **/usr/lpp/rlm/db/lic_db**
   - **/usr/lpp/rlm/db/nodelock**
   - **/etc/ncs/glb_sites.**
2. Copy the RLM **/etc/ncs/glb_sites** file to the iFOR/LS **/etc/ncs/glb_site.txt** file.

Note: Some RLM-enabled applications, although their license requests may be satisfied by iFOR/LS servers, require the existence of the **glb_sites** file to communicate with the **llbd** daemon. Both files may coexist on the same machine.

3. Install, configure, and start the NCS and the iFOR/LS software.

   On AIX Version 3.2 systems, when the iFOR/LS software is installed, the **rlmd** entry in the ODM database is removed; therefore **rlmd** cannot be managed anymore by the System Resource Controller (SRC).

   Configurations may exist where the **nrglbd** and **glbd** daemons are serving the same default cell. This is possible although not necessary because the **glbd** can fully substitute for the **nrglbd** daemon. We recommend always using the **glbd** daemon.

   Refer to “Configuring iFOR/LS Environment” on page 4-1 and to the *AIX Version 4.1 iFOR/LS System Management Guide* for more information on installing, configuring, and starting NCS and iFOR/LS.
4. Use the **ls_tv** command to verify the NCS-iFOR/LS communication. If the communication is not working properly, refer to the section on troubleshooting in the *AIX Version 4.1 iFOR/LS System Management Guide*.

5. Use the **ls_admin** command to add the RLM licenses to the iFOR/LS database. Using the information on the license form example on page 6-7, the command to add the vendor looks like the following:

   ```
   # ls_admin –a –v “Dassault Systemes”
   5242378dbf8d.02.c0.09.c8.93.00.00.00 69kw7w8zjbb2n
   ```

   To add the product and the licenses:

   ```
   # ls_admin –a –p “Dassault Systemes” “CATIA.MECH FBD–AIX/6000”
   p8fqnucgkyrf4mmpxjbn3j97u2 410
   ```

6. Use the **ls_stat –i** command to display all information about all licenses installed in the iFOR/LS database. In our example, the information looks like the following:

```
# /etc/netls/ls_stat –i

LS_STAT Version 2.0.1 (GRI 1.1.2) IBM_AIX
(c) Copyright 1991,1992,1993,1994, Hewlett–Packard Company, All Rights
Reserved
(c) Copyright 1991,1992,1993,1994, Gradient Technologies Inc., All
Rights Reserved
Installed Licenses
Dassault Systemes
CATIA.MECH FBD–AIX/6000 [410]
On server: strider.itsc.austin.ibm.com
Wait queue length: 0
            Licenses        Type  In Use  Start Date   Expiration Date
                   3  concurrent       0     03/10/94     12/31/99
Total active       3                   0
Hewlett–Packard NetLS Test
NetLS Test Product [1.0]
On server: strider.itsc.austin.ibm.com
Wait queue length: 0
            Licenses        Type  In Use  Start Date   Expiration Date
               10000  concurrent       0     01/01/70     01/19/38
Total active   10000                   0
End of Installed Licenses
```

Reverting to RLM Server (AIX Version 3.2)

If the attempt to migrate an RLM server to an iFOR/LS server is not successful, it would be helpful if the system could be restored back to the RLM server as it was before.

When the iFOR/LS software is installed, the **installp** process removes the **rlmd** entry in the ODM database; therefore RLM cannot be controlled by the System Resource Controller (SRC) afterwards. The RLM software and the **rlmd** database are not removed from the system and remain in the original directory **/usr/lpp/rlm**. To move back to the RLM server, it is necessary to recreate the **rlmd** entry in the ODM database.

The following steps need to be performed:

1. Recreate the **rlmd** entry in the ODM database using the **mkssys** command:

   ```
   # export ODMDIR=/etc/objrepos
   # mkssys –s rlmd –p /usr/lpp/rlm/bin/rlmd –u 0 –a ”–s 1000” –S
   –n3 –f3
   ```

   It is not necessary to reconfigure NCS to use the **nrglbd** daemon because **rlmd** can communicate with the **glbd** daemon.

2. Stop any running iFOR/LS and NCS daemons using the **stopsrc** command, and verify the results using the **lssrc** command.

3. Restart any NCS daemons using the **stopsrc** command, and verify the results using the **lssrc** command.
4. Restart the RLM daemon **rlmd**:

   ```
   # startsrc –s rlmd
   ```

Chapter 7. NCS Version 1.5.1 vs NCS Version 1.1

NCS Version 1.5.1 Features

The main features added in NCS Version 1.5.1 are:

- NCS-cell concept
- Support for replicatable Global Location Brokers

Advantages of the NCS-Cell Concept

NCS Version 1.1 supports communications only within the default cell, but, in the case of growing networks and increasing numbers of connected workstations, it may be necessary to partition the network into different NCS cells to make their administration easier.

NCS Version 1.5.1 allows a network to be partitioned into several cells, each serviced by its own GLB daemon. NCS cells have disjoint and independent GLB databases.

There are two classes of cells:

default cell	Uses a default Universal Unique Identifier (UUID) for the GLB object. This UUID is stored in the file **/etc/ncs/uuidname.txt**.
alternate cell	Uses an alternate UUID for the GLB object. This UUID is specified in the **/etc/ncs/glb_obj.txt** file of each host that is a member of the cell. Any number of alternate cells may exist in a network, each of them identified by a different UUID. There is no correspondence between cells and topology. This means that two hosts physically next to each other can belong to different cells while hosts in different networks and hundreds of miles apart can be part of the same cell.

GLB service for the default cell can be provided by either the nonreplicatable GLB daemon **nrglbd** or by the replicatable GLB daemon **glbd**, while in an alternate cell only the **glbd** daemon can provide services.

Advantages of Replicatable GLB

Depending on the availability of a single GLB server in the network is a big risk that many business units cannot afford. When a nonreplicatable GLB server fails, it may be a long time before a new GLB server is set up and all services provided within the cell are registered again.

To avoid this single point of failure in the business process, NCS Version 1.5.1 provides support for replicas of the GLB database that can be managed by GLB servers running on different hosts within the same NCS cell.

The information in replicas of the GLB database is kept consistent by the Data Replication Manager (DRM). A DRM runs in the same process as each GLB and manages the GLB database. The DRM maintains database consistency by propagating any change made in the GLB database to all replicas of the database.

Although the **nrglbd** can communicate with the iFOR/LS daemon **netlsd** and register its services within the default cell, it is strongly recommended to migrate nonreplicatable GLB servers to replicatable GLB servers.

NCS Static Database Files

NCS location brokers use static database files which are initialized during the location broker's startup. On location broker restart, the static database files are used to set up the communication path between the brokers in the network.

For example, a restarted **llbd** uses a static database file to rebuild the path it was using to communicate with the **glbd** before it stopped.

The names of the static database files used within NCS Version 1.1 have been changed in NCS Version 1.5.1.

NCS Version 1.1 Static Database Files

NCS Version 1.1 provides two NCS location brokers:

- The **llbd** daemon uses **/tmp/llbd–***HostName* as the static database file.
- The **nrglbd** daemon uses **/tmp/glbd–***HostName* as the static database file.

where *HostName* is the host where the daemons are running.

NCS Version 1.5.1 Static Database Files

NCS version 1.5.1 provides three NCS location brokers:

- The **llbd** daemon uses **/tmp/llbdbase.dat** as the static database file.
- The **uuid_gen** tool uses **/tmp/last_uuid** as the static database file.
- The **glbd** daemon uses **/etc/ncs/glb.e** as the static database file.
- The **nrglbd** daemon uses **/etc/ncs/glbdbase.dat** as the static database file.

Migrating NCS Servers

On AIX Version 3.2 systems, NCS Version 1.5.1 is distributed with the iFOR/LS software, while on AIX Version 4.1, it is distributed as the option **bos.net.ncs** within the **bos.net** package. Thus migrating servers from NCS Version 1.1 to NCS Version 1.5.1 is only necessary on AIX Version 3.2 systems.

On AIX Version 3.2, when the iFOR/LS software is installed, the **nrglbd** entry is removed from the ODM database (**/etc/objrepos/SRCsubsys**) and replaced with the **glbd** entry. This entry is necessary to allow the **glbd** daemon to be controlled by the System Resource Controller (SRC).

If iFOR/LS is configured using the **netls_config** shell script, the migration to NCS V1.5.1 is done automatically. When the **netls_config** shell script is not used, the following steps must be performed:

1. **If the host remains in the default cell, there are two options:**
   1. If there is already a replicatable GLB daemon **glbd** running on another host, the following command must be used to create a replica of this GLB database:

   ```
   # startsrc –s glbd –a "–create –from ip:HostName"
   ```

   2. If this is going to be the first host running a **glbd**, the following command must be used to create a new GLB database:

   ```
   # startsrc –s glbd –a "–create –first –family ip"
   ```

   3. Continue with step 4.
2. **If a new alternate cell should be created for this host:**

   a. The **/etc/ncs/glb_obj.txt** file must be created using the following command:

```
# /etc/ncs/uuid_gen > /etc/ncs/glb_obj.txt
```

   b. The following command must be used to create the new GLB database:

```
# startsrc –s glbd –a "–create –first –family ip"
```

   c. Continue with step 4.

3. **If joining an existing alternate cell:**

   a. The **/etc/ncs/glb_obj.txt** must be copied remotely from a host within this cell:

```
# rcp HostName:/etc/ncs/glb_obj.txt /etc/ncs/glb_obj.txt
```

   where *HostName* is a host in the existing alternate cell where a **glbd** daemon is running.

   b. If the **glbd** daemon is running on a host in another network, the **/etc/ncs/glb_site.txt** must be created. This file must contain the name or the IP address of the remote host. For example:

```
# cat /etc/ncs/glb_site.txt

ip:inti.austin.ibm.com
```

   c. The following command must be used to create a replica of this GLB database:

```
# startsrc –s glbd –a "–create –from ip:HostName"
```

   d. Continue with step 4.

4. In any of above cases the following must be done:

   a. The **lssrc** command must be used to verify that the **glbd** daemon is up and running:

```
# lssrc –g ncs
```

   b. The **/etc/rc.ncs** startup file must be modified to start the **glbd** instead of the **nrglbd** daemon.

Migrating NCS Clients

The following must be considered when migrating NCS Version 1.1 clients to NCS Version 1.5.1 clients:

Within the default cell:
If the NCS client is using the **/etc/ncs/glb_sites** file to contact the GLB daemon, this file must copied to **/etc/ncs/glb_site.txt.**

In an alternate cell:
If the **glbd** daemon is running on a host in another network, the **/etc/ncs/glb_site.txt** must be created. This file must contain the name or the IP address of the remote host. For example:

```
# cat /etc/ncs/glb_site.txt

ip:inti.austin.ibm.com
```

Some NCS Version 1.1 based applications need the obsolete file **/etc/ncs/glb_sites;** therefore, it is recommended to copy the **glb_site.txt** to the **glb_sites** file.

1. The cell’s **/etc/ncs/glb_obj.txt** file must be copied remotely from the host where the **glbd** daemon is running.

   ```
   # rcp HostName:/etc/ncs/glb_obj.txt /etc/ncs/glb_obj.txt
   ```

   where *HostNname* is a host in the existing alternate cell where a **glbd** daemon is running.

Chapter 8. iFOR/LS Performance and Availability

The use of licensing systems may affect system or application performance and availability, if the client/server model has not been effectively implemented either by the configuration or when developing the application. This chapter provides possible reasons for performance and availability impacts during usage of iFOR/LS.

Since system and application performance and availability strongly depend on the components (such as application, network topology, servers, and transactions) used in each individual environment, no general statement can be made in terms of how the iFOR/LS licensing system may affect these topics. This chapter includes explanations for some behavior system administrators may see on their systems when using iFOR/LS.

It is recommended to perform problem evaluation and determination whenever performance or availability impacts are recognized.

iFOR/LS Performance Implications

The NCS (**llbd**, **glbd**) and iFOR/LS (**netlsd**) daemons involved within the iFOR/LS licensing mechanism do not have a noticeable effect on system performance as a whole.

In a running environment, the **llbd** and **glbd** daemons communicate with each other in the following cases:

- When the **glbd** starts, to register itself to the **llbd** daemon
- When the **netlsd** is started or stopped, to register or unregister its services, respectively.

The **netlsd** communicates with the **glbd** daemon, to register or unregister vendors or products under its management.

The **glbd** daemons within a cell communicate with each other through the Data Replication Manager (DRM) to synchronize their replicated GLB databases whenever they are updated due to services registration.

Although more than one iFOR/LS server may exist within a cell, they are independent and do not need to communicate with each other. There is no impact on the overall network performance due to server-to-server communications.

On the other hand, the workstations or network workload may influence the communication between the daemons, or between applications and daemons. Nevertheless, this will affect only the application startup time, not the application performance.

When there is a heavy system or network workload, communication between client and server may be lost or delayed. iFOR/LS supports several policies which can be implemented in case the communication between iFOR/LS clients (application) and the iFOR/LS server (**netlsd**) fails.

The iFOR/LS policies are as follows:

On license request (application startup):

During startup, the application cannot communicate with the **netlsd** daemon. It behaves in one of the following ways, depending on which policy it has implemented:

Hard stop	The application does not start.
Soft stop	The application may start up without a license.

Note: Applications may have implemented the iFOR/LS queuing function. If licenses are not available, the request may be queued until a license is available.

During run time (heartbeat):

Periodically, a running application must communicate with the **netlsd** daemon to state that it is still active. The period, also called timeout**,** is selected by the application developer and may be configurable. Depending on the implementation, the application behaves as follows:

Heartbeat hard stop
The application terminates.

Heartbeat soft stop
The application continues running.

Notes:

1. The heartbeat period can affect performance through network traffic and CPU usage to acknowledge the heartbeat. If a number of licenses are checked out and the heartbeat is set to a short period, performance may be adversely impacted. Performance impacts can be reduced by increasing the heartbeat timeout. The disadvantage to longer timeout periods is that **netlsd** has to wait longer to recover the licenses.
2. It is recommended to implement a timeout period of 5 minutes. Using this period, and due to the fact that it is asynchronous, the heartbeat will not influence system performance.
3. The default period for the **monitord** which controls the AIX Version 4.1 BOS login processes is 15 minutes.
4. Some applications try to contact the **llbd** first before contacting the **glbd**. If the **llbd** is not running in the local machine, these applications wait until NCS timeout. Since this behavior negatively impacts the application startup time when the **llbd** has not been started, you may consider starting the **llbd** on all hosts running these applications.

Sample Scenario

The following sample scenario shows that license request/response times may be independent of the network. The following two machines are using the NCS default cell and are located thousands of miles apart.

Host name	strider	mitc53h
Daemons	llbd	llbd, glbd, rlmd
CATIA V4 license server		*
Local application	CATIA V4	CATIA V4
Location	Austin - USA	Mainz - Germany

On `strider`: The **ping** command is used to determine the delay in the network.

```
# ping mitc53h

PING mitc53h: (9.39.0.207): 56 data bytes
64 bytes from 9.39.0.207: icmp_seq=0 ttl=248 time=976 ms
64 bytes from 9.39.0.207: icmp_seq=1 ttl=248 time=2033 ms
64 bytes from 9.39.0.207: icmp_seq=2 ttl=248 time=1194 ms
64 bytes from 9.39.0.207: icmp_seq=3 ttl=248 time=1033 ms
64 bytes from 9.39.0.207: icmp_seq=4 ttl=248 time=957 ms
64 bytes from 9.39.0.207: icmp_seq=5 ttl=248 time=1104 ms
64 bytes from 9.39.0.207: icmp_seq=6 ttl=248 time=1149 ms
64 bytes from 9.39.0.207: icmp_seq=7 ttl=248 time=888 ms
^C
----9.39.0.207 PING Statistics----
9 packets transmitted, 8 packets received, 11% packet loss
round-trip min/avg/max = 888/1166/2033 ms
```

The average time delay for a round-trip **ping** is 1.166 s.

On `strider`: The CATIA Version 4 startup time on `strider` is 9.20 s.

```
root@strider# time catutil

real    0m9.20s
user    0m1.95s
sys     0m2.82s
```

On `mitc53h`: The CATIA Version 4 startup time on `mitc53h` is 9.19 s, only 0.01 s less than on `strider`, although `strider` had to obtain the license from `mitc53h`

```
root@mitc53h# time catutil

real    0m9.19s
user    0m1.89s
sys     0m2.72s
```

iFOR/LS Availability Implications

Licensing systems that are designed to deliver licenses through a license server rely on the ability of the application to communicate remotely with the license server and the license database. The client/server model can be implemented with the following architectures:

Single-server architecture
This design dictates that an application be able to obtain a license through communication with a single server designated for this particular application. This scheme is simple and easy to administer but does not provide a solution when the server is not available.

Multiple, nonreplicated server architecture
With this design an application can be serviced by more than one license server. The server is no longer a single point of failure, but the administration is more complex. Administration tools must be provided to reduce complexity.

Multiple, replicated server architecture
This design is similar to the nonreplicated architecture but needs server-to-server communication to maintain and synchronize multiple copies of a license database. This type of communication may have an

impact on the overall network performance and the performance associated with obtaining a license.

iFOR/LS is designed to use multiple, nonreplicatable servers. Any application can be serviced by any one of a set of iFOR/LS servers within the cell. These servers are totally independent from each other and maintain separate license databases.

Warning: iFOR/LS servers are independent from each other. If an iFOR/LS server is not available, the licenses it maintains are not available, even if there are other iFOR/LS servers in the NCS cell.

The iFOR/LS policy implementation of each application may affect the availability of this specific application. For example, if an application has been implemented using the heartbeat-hardstop policy, it will stop if communication with the license server is lost for a time longer than the timeout period. Refer to “iFOR/LS Performance Implications” on page 8-1 for more information on the iFOR/LS implementation policies.

iFOR/LS Availability Strategies

The following strategies should be considered to improve availability when using the iFOR/LS licensing system:

- Use more than one iFOR/LS server and split the licenses for each single product between them to avoid losing all licenses for a product when a server is unavailable.
- Use machines that are available, reliable, and controlled.
- Designate license administrators. Only they should have superuser access to the iFOR/LS servers.
- Implement a backup/recovery management plan. Refer to “iFOR/LS Backup and Recovery Examples” on page 5-18 for more information.
- Use the iFOR/LS license usage reporting tools to which license requests could not been satisfied and why. Maybe you need more licenses for a specific product.
- Use more than one replicated GLB daemon (**glbd**).
- Since availability is a system-wide consideration, ensure the other system components (network, domain servers, etc.) are available.
- Consider eliminating single point-of-failure by using HACMP.

Chapter 9. iFOR/LS Security Features

The security requirements for licensing systems demand support for easy resource sharing and mass distribution of software (on CD, for example). A supplier's ability to ship fully functional trial copies of applications relies heavily on the security of the licensing system to ensure that an interested prospect does not intentionally or unintentionally transgress usage permissions.

The iFOR/LS design goal relating to security is to make the effort to break security more costly than simply being honest and purchasing the licenses needed. In addition, compromises to the licensing system require deliberate and intentional actions that are quite evident.

However, the goal of iFOR/LS is also to delegate as much control over security as possible to the application developers. Thus, the application developer has the ability to implement the level of enforcement and amount of availability that is appropriate for each application.

Vendor Security Key

iFOR/LS uses a naming hierarchy to distinguish between applications. The hierarchy is:

- Vendor
- Product
- Version

Although readable vendor names are supported for easy administration, the vendor-naming space is guaranteed to be unique by internally using a vendor ID, thus removing the possibility of inter-vendor interference. The license creation process requires knowledge of a per-vendor security key. As long as this security key remains private, the vendor retains total control over license creation.

The license server uses the vendor security key both to encode and decode license information in a password and to encode and decode communications between license servers and the products they control.

Compound Keys

Since the security key to generate licenses should remain the application developer's secret, the ability to grant license-generation capabilities to distributors and resellers requires a mechanism that works without the disclosure of the security key.

To solve this problem, iFOR/LS uses a license type known as the compound license, which allows authorized agents to create a specified set of licenses for end users, or other authorized agents, without needing the application developer's security key.

The application developer retains complete control not only over the security key, but also over the license-generation rights passed on to the distributor. The application developer, when creating the compound-license key for the distributor, can set an expiration date, beyond which no more end-user licenses can be created.

To generate derivative keys from a compound key, the **ls_dpass** utility included in the **bos.ifor_ls.server** option, on AIX Version 4.1, must be used.

Security of License Data

To ensure the security of license data, passwords are nodelocked to the iFOR/LS server, encoded, and time-stamped. These security precautions prevent a number of possible security violations, as follows:

Copying passwords to multiple servers

Because passwords are nodelocked, users cannot increase the number of licenses by copying passwords to multiple servers.

Reinstalling passwords at server nodes

Passwords are time-stamped, and if the license server receives a new password with the same timestamp as an old password for the same product, it does not update the license database.

Manually patching the license database

Patching license data in the license database will not work. Although the server decodes the vendor-key-encoded passwords when the passwords are installed, it also subsequently re-encodes the license data using a special license server key.

Nonsecure License Data

Some application vendors may want their software products to be available to certain customers, or on on certain classes of computer, without security restrictions. To this end, nonsecure passwords can be created. For example, one can create passwords that can be installed on and copied to any node of any type.

Client/Server Authentication

Without security, an impostor server could be established by eavesdropping on valid client-server communications and then simulating the license-granting protocol. A successful impostor client could disrupt legitimate license activities by artificially returning a license to the server that is actually in use, in so doing making the license available for other users.

iFOR/LS ensures that valid clients and servers are used by enforcing a bilateral authentication scheme. Each client and server authenticates the other in all communications to verify that each is a legitimate iFOR/LS process.

Security Impact for iFOR/LS End Users

The NCS and iFOR/LS products do not impact the AIX Version 4.1 Base Operating System and additional security features the customer may have chosen.

License Database Security

Since all iFOR/LS tools utilize the remote procedure call mechanism of NCS, the tools have transparent access to all networked license servers within an NCS cell. This allows all nodes to be centrally administered from one administration console. This capability reduces administration costs but also allows for situations that, if not handled carefully, might impact license-availability security.

Within an NCS cell, any iFOR/LS administrator on any host has administrator access to any iFOR/LS license database. This means, the administrator on host A can add, change, and delete licenses stored on host B. This may be dangerous when working in the internet, and especially when the default cell has been chosen.

Warning: License databases can be manipulated by any iFOR/LS administrator within the NCS cell. Since hosts in a NCS alternate cell need to know the cell’s UUID, it is recommended to use alternate cells for better license-database protection. An NCS cell should be seen as a licensing system unit with a single administrator or administration team. See the **netlsd –s** flag for a method of restricting database access.

License Usage Security

By default, any user can request any concurrent-use license serviced by any iFOR/LS server within an NCS cell. Optionally, user files (**/usr/lib/netls/conf/user_file**) can be used to prohibit some users from accessing certain products or to give certain users different access priorities.

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.

Appendix A. Configuring iFOR/LS within an HACMP Environment

The **hacmp_netls.config.sh** shell script may be used instead of the **netls_config** and **netls_first_time shell** scripts to set up iFOR/LS systems in an HACMP environment. The shell script may work for most HACMP configurations, but because of the configuration flexibility of HACMP, changes may be necessary for adapting it to your specific HACMP environment. Please read the shell script carefully before using it to make sure it meets your needs.

hacmp_netls.config.sh Shell Script

Warning: The shell script has not been submitted to any formal test and is distributed AS IS.

```
#!/bin/ksh
# @(#) Version 1.0  hacmp_netls.config.sh   03/28/94
#=================================================================#
#   Script :  hacmp_netls.config.sh                               #
#   Authors:  M. Crisanto / L. Denefleh  / F. Kraemer             #
#   Date   :  94/03/28                                            #
#   Update :  94/--/--                                            #
#                                                                 #
#   Info   :  This shell script might be used instead of the      #
#             netls_config and netls_first_time shell scripts     #
#             provided with the iFOR/LS software to iFOR/LS       #
#             enable an HACMP/6000 cluster environment consisting #
#             of two or more servers.                             #
#                                                                 #
#   When   :  Run this shell script on each HACMP/6000 server     #
#             when the cluster manager daemon, clstrmgr, is       #
#             running and is in the stable state.                 #
#                                                                 #
#   Special:  This script will reconfigure the /etc/inittab       #
#             file and detach the standby network interfaces.     #
#             This is mandatory to avoid the GLB daemon           #
#             broadcasting over the standby subnet.               #
#                                                                 #
#             If the standby interfaces are not detached during   #
#             the first Global Location Broker daemon (glbd)      #
#             startup, the standby network interfaces might be    #
#             used instead of the service interfaces whenever     #
#             the glbd daemon tries to reach the network.         #
#                                                                 #
#   Step by   1) Login as the root user.                          #
#   Step:     2) Copy the shell script to the first HACMP system. #
#             3) Edit the shell script and verify the various     #
#                defined variables.                               #
#             4  Run the script.                                  #
#             5) Copy the changed script to the second            #
#                HACMP system.                                    #
#             6) Verify the defined variables in the script again.#
#             7) Run the script on the second HACMP system.       #
#                                                                   #
#===================================================================#

#>>>>>>>>>>>>>>>>>>> START–OF–CONFIGURE–SECTION <<<<<<<<<<<<<<<<<<<#

#-------------------------------------------------------------------#
#  The following variables MUST be set:                             #
#-------------------------------------------------------------------#
#–– Hostname of the first HACMP system. This value is compared with
#–– the THIS_HOST variable to find out on what system we are running.
HA_HOST1=””
#
#–– Hostname of the second HACMP system.
HA_HOST2=””
#
#–– DNS/BIND DOMAIN Name of local host (e.g. ”.itsc.austin.ibm.com”)
#–– Start name with a leading ’.’ (dot).
DOMAIN=””
#
#–– List of all HACMP STANDBY adapters (e.g. ”tr1 tr3 en1 en3”)
STDBY=””
#
#-------------------------------------------------------------------#
#  The following variables CAN be set                               #
#-------------------------------------------------------------------#
#–– Use the output of the /usr/lib/ncs/uuid_gen command to define
#–– your own NCS cell or ”” if you would like to set up a default cell
#–– this script will always configure the HA_HOST1 as the first member
#–– of a certain cell (either default or alternate)
#–– (e.g. ”662c167abb83.02.09.03.01.5c.00.00.00”)
CELL_UUID=””
#
#–– Defines the local hostname. This value is compared with the variable
#–– HA_HOST1 to find out on what system we are running.
THIS_HOST=$(/usr/bin/hostname –s)
#
#-------------------------------------------------------------------#
#  Please verify these variables.                                   #
#-------------------------------------------------------------------#
#
GLB_OBJ_FILE=”/etc/ncs/glb_obj.txt”
#
GLB_SITE_FILE=”/etc/ncs/glb_site.txt”
#
PROG=$(/usr/bin/basename $0)
#
NCS_TEST_SCRIPT=”/usr/local/bin/ncs_test.sh”
#
NCS_TEST_LOG=”/tmp/ncs_test.log.$$”
#
#-------------------------------------------------------------------#
#>>>>>>>>>>>>>>>>>>>> END–OF–CONFIGURE–SECTION <<<<<<<<<<<<<<<<<<<<#
#-------------------------------------------------------------------#
#  Check if variables are not set up                                #
#-------------------------------------------------------------------#
if [ ”${HA_HOST1}” = ’’ –o ”${HA_HOST2}” = ’’ –o ”${STDBY}” = ’’ ]
then
   echo ”$PROG ** Please edit this script and define the variables ! **\n”
   exit –1
```

```
fi
#-------------------------------------------------------------------#
#  Stop all NCS & NetLS related programs                            #
#-------------------------------------------------------------------#
for RESOURCE in  netlsd  \
                 glbd    \
                 nrglbd  \
                 llbd
do
     echo ”$PROG will stop subsystem ${RESOURCE}”
     /usr/bin/stopsrc –s ${RESOURCE} >/dev/null 2>&1
     /usr/bin/sleep 1
done
#-------------------------------------------------------------------#
#  Fix the netls daemon entry in /etc/inittab if clinit is there  #
#-------------------------------------------------------------------#
/usr/sbin/lsitab ”clinit” >/dev/null 2>&1
if [ $? –eq 0 ]; then
     echo ”$PROG will set up /etc/inittab entries for netlsd”
     CMD=’sh /etc/rc.netls >/dev/console 2>&1 # start netls’
     /usr/sbin/rmitab ”netlsd” >/dev/null 2>&1
     /usr/sbin/mkitab –i rcncs ”netlsd:a:wait:${CMD}”
fi
#-------------------------------------------------------------------#
#  Start the LLB and GLB deamons from /etc/rc.ncs                   #
#-------------------------------------------------------------------#
/usr/bin/grep ”^#startsrc” /etc/rc.ncs > /dev/null 2>&1
if [ $? –eq 0 ]
then
    echo ”$PROG: Update the /etc/rc.ncs file to start daemons at boot
time”
    /usr/bin/cp /etc/rc.ncs /tmp/rc.ncs.$$
    /usr/bin/cat /tmp/rc.ncs.$$ |
       /usr/bin/sed ”/^#startsrc/s//startsrc/” > /etc/rc.ncs
    /usr/bin/rm –f /tmp/rc.ncs.$$
else
    echo ”$PROG: The /etc/rc.ncs file was found ok”
fi
#-------------------------------------------------------------------#
#  Remove these files to make sure we have a plain system           #
#-------------------------------------------------------------------#
for RMFILE in /etc/ncs/glb.e         \
              /etc/ncs/glb.p         \
              /etc/ncs/glb_log       \
              /etc/ncs/glb_site.txt  \
              /etc/ncs/glb_obj.txt   \
              /tmp/llbdbase.dat
do
    if [ –f $RMFILE ]; then
       echo ”$PROG will remove $RMFILE to clean up the local system”
       /usr/bin/rm –f $RMFILE
    fi
done
#-------------------------------------------------------------------#
# Set up the glb_obj.txt file with the right UUID for this cell   #
#-------------------------------------------------------------------#
if [ $CELL_UUID = ’’ ]
then
   echo ”$PROG $THIS_HOST is a member of the ’default’ NCS cell”
```

```
else
   echo ”$PROG $THIS_HOST is a member of the NCS cell using UUID
$CELL_UUID”
   echo $CELL_UUID > $GLB_OBJ_FILE
fi
#------------------------------------------------------------------#
# Set up the glb_site.txt file for the HACMP systems               #
#------------------------------------------------------------------#
if [ ${THIS_HOST} = ${HA_HOST1} ]
then
     echo ”ip:${HA_HOST1}${DOMAIN}”  >  ${GLB_SITE_FILE}
     echo ”ip:${HA_HOST2}${DOMAIN}”  >> ${GLB_SITE_FILE}
fi
#-----------------------------------
if [ ${THIS_HOST} = ${HA_HOST2} ]
then
     echo ”ip:${HA_HOST2}${DOMAIN}”  >  ${GLB_SITE_FILE}
     echo ”ip:${HA_HOST1}${DOMAIN}”  >> ${GLB_SITE_FILE}
fi
#------------------------------------------------------------------#
#  Detach the STANDBY network adapters to make sure the GLB data– #
#  base contains only entries for the SERVICE network adapters.   #
#------------------------------------------------------------------#
for DEVICE in $STDBY
do
     echo ”$PROG the STANDBY network adapter $DEVICE will be detached”
     /usr/sbin/chdev –l $DEVICE –a state=detach
done
#------------------------------------------------------------------#
#  Start up the local location broker daemon (LLBD)                #
#------------------------------------------------------------------#
echo ”$PROG start up the Local and Global Location Broker Daemon”
/usr/bin/startsrc –s llbd
/usr/bin/sleep 3
#------------------------------------------------------------------#
#  Start up the global location broker daemon (GLBD)               #
#  On HA_HOST1 start the first (master) glbd daemon of the cell    #
#  On HA_HOST2 start the glbd daemon as a replica of HA_HOST1      #
#------------------------------------------------------------------#
if [ ${THIS_HOST} = ${HA_HOST1} ]
then
     /usr/bin/startsrc –s glbd –a ”–create –first –family ip”
else
     /usr/bin/startsrc –s glbd –a ”–create –from ip:${HA_HOST1}${DOMAIN}”
fi
/usr/bin/sleep 6
#------------------------------------------------------------------#
#  Start up the local netlsd daemon                                #
#------------------------------------------------------------------#
echo ”$PROG start up the Netls Daemon”
/usr/bin/startsrc –s netlsd
/usr/bin/sleep 3
#------------------------------------------------------------------#
#  Bring up the STANDBY network adapters again.                    #
#------------------------------------------------------------------#
for DEVICE in $STDBY
do
     echo ”$PROG the STANDBY network adapter $DEVICE will be activated”
     /usr/sbin/chdev –l $DEVICE –a state=up
     /usr/bin/sleep 2
done
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Test the NCS database using the ncs_test.sh shell script       #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
if [ –x $NCS_TEST_SCRIPT ]
then
   echo ”$PROG starting the NCS test shell script”
   $NCS_TEST_SCRIPT | /usr/bin/tee $NCS_TEST_LOG
   echo ”$PROG NCS test shell script output is saved in $NCS_TEST_LOG”
fi
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Ok guess we are done now                                        #
#––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
exit 0
```

Appendix B. Testing the NCS Configuration for iFOR/LS

The **ncs_test.sh** shell script can be used to ensure all the definitions in the NCS environment are set up properly. This shell script uses the **lsof** command, if installed, to find out the port numbers on which the **llbd**, **glbd**, and **netlsd** daemons are communicating. The **lsof** command is public-domain software and can be found on **vic.cc.purdue.edu** (128.210.15.16). The identification of the run-time-assigned **glbd** and **netlsd** ports is not essential: therefore, the shell script will not break if the **lsof** software is not installed.

After the UDP ports have been identified, the **ncs_test.sh** shell script traces the well-known **llbd** UDP port (135) and collects information on the traffic caused by the **lb_admin** command. The formatted report is displayed after it is processed by the **ipreport** command.

ncs_test.sh Shell Script

Warning: The shell script should be used for reference only. It has not been submitted to any formal test and is distributed AS IS.

```
#!/bin/ksh
# @(#) ncstest.sh NetLS/NCS 1.5.1 verify/debug script 94/03/15
#=================================================================#
#   Authors :  M. Crisanto / L. Denefleh / F. Kraemer             #
#-----------------------------------------------------------------#
LSOF=”/usr/local/bin/lsof”          #  path to lsof command
LBADMIN=”/usr/lib/ncs/bin/lb_admin” #  path to lb_admin command
AWK=”/usr/bin/awk”                  #  path to awk  command
CUT=”/usr/bin/cut”                  #  path to cut  command
SED=”/usr/bin/sed”                  #  path to sed  command
MYPID=”$$”                          #
LPORT=”135”                         #  llbd runs here
IPTRACE=”1”                         #  start iptrace (1=yes/0=no)
IPTRFILE=”/tmp/iptrace.$MYPID”      #  filename of iptrace output
#-----------------------------------------------------------------#
#  Prepare some staff before doing real work.                     #
#-----------------------------------------------------------------#
if [ ”$(/usr/bin/whoami)” != ”root” ]; then
  echo ”\n\tYou must be root to run this script.”
  exit –1
fi
TMP_FREE=$(/usr/bin/df ”/tmp” | $AWK ’$3 ~ /[0–9]/{print $3}’)
if [ ”${TMP_FREE}” –lt 1000 –a ”${IPTRACE}” = ”1” ]; then
   echo ”\n\tThere is not enough room in your /tmp directory.”
   echo ”\tYou need 1000 KB free, and you have only $TMP_FREE KB free.\n”
   exit –1
fi
if [ ! –x ${LBADMIN} ]; then
   echo ”\n\tCan not find the ${LBADMIN} command on the system.”
   echo ”\tPlease verify the NetLS installation.\n”
   exit –1
fi
#-----------------------------------------------------------------#
#  Ok all checks are done we can take off.                        #
#-----------------------------------------------------------------#
PROG=$(basename $0)
```

```
HOST=$(hostname –s)
TODAY=$(date +%H:%M:%S)
echo ”\n\t$PROG started from $LOGNAME@$HOST on $TERM at $TODAY.\n”
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Use lsof to find the portnumber of llbd,glbd and netlsd daemon #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
if [ –x $LSOF ]; then
   LRT=$($LSOF –i”UDP” | $AWK ’{if ($1 == ”llbd”)   print $9}’ | $CUT –c3–7)
   GRT=$($LSOF –i”UDP” | $AWK ’{if ($1 == ”glbd”)   print $9}’ | $CUT –c3–7)
   NRT=$($LSOF –i”UDP” | $AWK ’{if ($1 == ”netlsd”) print $9}’ | $CUT –c3–7)
#––––––––
   LPORT=$(echo $LRT | $SED ’s/\\n/ /g’)   #  Format the staff
   GPORT=$(echo $GRT | $SED ’s/\\n/ /g’)   #
   NPORT=$(echo $NRT | $SED ’s/\\n/ /g’)   #
#––––––––
   if [ ”$LPORT” = ”” ]; then             #  Print info
      echo ”\n\t** No Local Location Broker is running on local system **”
   else
      echo ”\n\tLLB is using UDP port(s)   := $LPORT”
   fi
#––––––––
   if [ ”$GPORT” = ”” ]; then             #  Print info
      echo ”\n\t** No Global Location Broker is running on local system **”
   else
      echo ”\n\tGLB is using UDP port(s)   := $GPORT”
   fi
#––––––––
   if [ ”$LPORT” = ”” ]; then             #  Print info
      echo ”\n\t** No NetLS daemon is running on local system **”
   else
      echo ”\n\tNetLS is using UDP port(s) := $NPORT\n”
   fi
else
   echo ”\n\tlsof – List_of_Open_Files is not installed on your system.”
   echo ”\tthe tool is a public domain program and can be found on”
   echo ”\tvic.cc.purdue.edu (128.210.15.16), Vic Abell is the author.\n”
fi
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Fire up an IP trace on the llbd UDP (135)                      #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
if [ $IPTRACE = ”1” ]; then
   /usr/bin/rm –f ${IPTRFILE} 2>/dev/null
   /usr/bin/iptrace –P ”UDP” –p ”$LPORT” ${IPTRFILE}
   sleep 3
fi
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
#  Use lb_admin command to hear on the llbd UDP port (135)        #
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
cat <<EOF | ${LBADMIN}
set_timeout long
set_timeout
use_broker local
use_broker
lookup
use_broker global
lookup
quit
EOF
#–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––#
```

```
#  Stop the IP trace and format its output                         #
#------------------------------------------------------------------#
if [ $IPTRACE = ”1” ]; then
   Target=$(ps –e | grep ”iptrace”)
   echo ”\n\tKilling $Target with signal 1”
   /usr/bin/kill 1 $(echo $Target | cut –f1 –d” ”)
   echo ”\n\tFormating iptrace output via ipreport.....please wait\n”
   sleep 3
   /usr/bin/ipreport –r ${IPTRFILE}
   /usr/bin/rm –f ${IPTRFILE}
fi
#------------------------------------------------------------------#
#  We are done.....hope you had fun.                               #
#------------------------------------------------------------------#
exit 0
```

Index

Reader's Comment Form

iFOR/LS Tips and Techniques

Order number SC23-2666-00

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.

☐ 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.

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

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