FLEXlm™

Programmers Guide

VERSION 7.0

GLOBE*trotter* Software, Inc.

November, 1999

Chapter 1

Introduction

1.1 About this Manual

This manual, the FLEX*lm*® Programmers Guide, is an introduction to FLEX*lm* and includes a complete description of the simplest Application Programming Interfaces (APIs) that can be used to incorporate license management into your application. This manual describes the license administration tools that are bundled with FLEX*lm*, and provides guidelines for integration of FLEX*lm* into your application.With this manual, you should be able to have an example license-managed application up and running within a few hours.

The FLEX*lm* Reference Manual provides a comprehensive description of all other aspects of FLEX*lm* from the software developer's perspective, including a complete description of the FLEX*ible* API, the most complete API available for license management. Companies upgrading from a version older than version 6 should refer to Appendix F in the Reference Manual, "Upgrading FLEXlm to Version 6—Compatibility Issues".

The FLEX*lm* End-User Manual contains information for products that utilize FLEX*lm* as their licensing system. This manual describes the setup and administration of a FLEX*lm* licensing system.

1.2 How to use this Manual

If you are getting started with FLEX*lm*, we suggest that you read chapters 1-3 of this manual. By the end of chapter 3, you will have installed FLEX*lm* and run a sample application with license management in a number of different situations.

Once you are familiar with how FLEX*lm* operates, the remainder of this manual provides reference material for integrating FLEX*lm* into your application and includes instruction in the use of the administration tools provided with FLEX*lm*, and general guidelines on configuring your application and its licensing software.

1.3 Introduction to FLEX*lm*

FLEX*lm* is a software licensing package that allows licensing a software application on a concurrent-usage as well as on a per-computer basis. FLEX*lm* allows the implementation of a wide variety of *license policies* by the developer of an application.

With FLEX*lm*, you, the application developer, can restrict the use of your software packages to:

- A single specified computer
- A specified number of users on a network of one or more computer systems

FLEX*lm* is available on UNIX, Windows, Windows NT, VMS, and Netware systems. FLEX*lm* features include:

- Operation in a heterogeneous network of supported computer systems
- Transparent reconnection of applications when their license server process becomes unavailable, including conditions of license server node failure
- Simple configuration by using a single license file per network
- Configuration controls for System Administrators
- Administration tools for System Administrators
- Independent features from one or multiple vendors with independent vendor security codes
- A wide variety of *license policies* and *license styles,* including:
    - ✧ Floating licenses
    - ✧ Node locked licenses
    - ✧ Personal use licenses
    - ✧ Demo licenses
    - ✧ Counted and uncounted licenses
    - ✧ Optional license expiration dates
    - ✧ Several vendor-definable fields for each application feature
- License management on redundant server hosts for improved license availability due to hardware failure

1.4 FLEX*lm* Terms and Definitions

The following terms are used to describe FLEX*lm* concepts and software components:

feature	Any functionality requiring licensing. The meaning of a feature will depend entirely on how it is used by an application developer. For example, a feature could represent any of the following: • An application software system consisting of hundreds of programs • A single program (regardless of version) • A specific version of a program • A part of a program • A piece of data (restricted via the access routines)

license	The legal right to use a feature. FLEX*lm* can control licenses for features by counting the number of licenses currently in use for a feature when new requests are made by the application software (client). FLEX*lm* can also restrict software usage to particular nodes or user names.
client	An application program requesting or receiving a license.
daemon	A process that “serves” clients. Sometimes referred to as a *server.*
vendor daemon	The daemon that dispenses licenses for the requested features. This daemon is built by an application’s vendor (from libraries supplied by GLOBE*trotter* Software) and contains the vendor’s unique encryption seeds.
lmgrd	The daemon process, or license daemon, that sends client processes to the correct vendor daemon on the correct machine. The same license daemon process can be used by all applications from all vendors, as this daemon neither authenticates nor dispenses licenses. lmgrd processes no user requests on its own, but forwards these requests to other daemons (the vendor daemons).
server node	A computer system running the license server software. The server node will contain all site-specific information regarding all feature usage. Multiple server nodes used for redundancy can logically be considered the *server node*.
license file	A site-specific file that contains descriptions of server node(s) that can run the license daemons, various vendor daemons, and licenses (features) for all supported products.
license file list	A list of license files separated with a colon ‘:’ on Unix, a semi-colon ‘;’ on Windows and a space on VMS. A *license file list* can be accepted in most places where a license file is appropriate. When a directory is specified, all files matching **.lic* in that directory are automatically used, as if specified as a list.
license key	A 12- to 20-character hexadecimal number which “authenticates” the readable license file text, ensuring that the license text has not been modified.
license server	The lmgrd and vendor daemon processes. License server refers to the processes, not the computer.

1.5 How FLEX*lm* Works

FLEX*lm* is a client-server application toolkit. The client (your application) requests a license from the license server and is either granted or denied the license.

The five main components in FLEX*lm* are:

- Client library (embedded in the license-managed application)
- lmgrd, the license daemon
- Vendor daemon(s)
- Vendor and end-user license administration tools
- License file(s)

The end-user installs lmgrd and the vendor daemon on the license server node. Once the license file(s) and the daemons are in place, the only requirement is to start lmgrd. The daemon is typically started when the machine boots (in the machine startup file, or on Windows as a system service), but can also be started later by any user.

With the license file installed in an expected location and the FLEX*lm* daemons running, FLEX*lm* use is transparent to the end-user.

SEE ALSO

- Chapter 10, "The License File" on page 60
- Chapter 13, "End-User License Administration" on page 76

1.6 FLEX*lm* APIs

The application program interfaces to FLEX*lm* via a set of routines that request (checkout) and release (checkin) licenses of selected feature(s).

There are 4 APIs available to the developer:

- Trivial API
- Simple API
- FLEX*ible* API
- Java API (for programs written in the Java language)

Globetrotter recommends using the Trivial API; if the application requires functionality not provided in this API, use the Simple API functions; for complete flexibility use the FLEX*ible* API. The FLEX*ible* API is described in the FLEX*lm* Reference Manual.

In the Trivial and Simple APIs, a licensing "policy" is selected as an argument to the license request call. In these APIs, a "heartbeat" function is usually called explicitly by the application and policy upon server failure must be programmed into the application.

Most of the important functionality and flexibility in FLEX*lm* is contained in the license file; all license file attributes are available to all APIs.

Chapter 2

Installing the Distribution Kit

FLEX*lm* kits are obtained from the Globetrotter web site:

```
http://www.globetrotter.com/lmpostreg.htm
```

Vendor Keys are required from Globetrotter to successfully download and install FLEX*lm*.

Once installed, the manuals are available on-line.

If you are installing on Windows, NT, or Netware, skip to Section 2.2, “Installation for Windows, NT, and Netware Systems,” on page 13.

2.1 Installation from FTP site on Unix

1. Get the 5 Vendor Keys from Globetrotter Software. (Note that FLEXlm v7 Vendor Keys are different from FLEXlm previous versions).
2. Connect to Globetrotter’s web site:

   ```
   http://www.globetrotter.com/lmpostreg.htm
   ```
3. Download the following files into your local directory (use BINARY mode):

   ```
   install_flexlm.ftp
   machine_ind.tar
   PLATFORM_1.tar
   PLATFORM_2.tar
   ...
   PLATFORM_n.tar
   ```

   where “PLATFORM” is the type of machine you are running on, e.g.:

   ```
   sun/solaris - sun4_u5
   hp9000/700  - hp700_u9
   ```

   You must download the tar file for the machine on which you will run the install_flexlm.ftp, so that it can install the files properly and run the complete installation.
4. Run the install_flexlm.ftp script in your local directory (note that this directory is referred to as the “GSI home directory” in the demo guide):

   ```
   % sh install_flexlm.ftp
   ```

   This will create the flexlm directories. It will also start the FLEXlm INSTALL script to install FLEXlm. This INSTALL script, in turn, requires FLEXlm Vendor Keys, which you must obtain from Globetrotter.

The install_flexlm.ftp script does the complete install from the ftp site tarring the files as necessary. Should any errors occur during the install, or it ever needs to be re-run, you can always re-use install_flexlm.ftp, and it will start from where it last left off.

install_flexlm.ftp calls flexlm/v6.1/INSTALL, which automatically edits *lm_code.h* and changes the DAEMON name in the makefile to your vendor-daemon name (which is listed along with the vendorkeys from Globetrotter, and is always "demo" with an evaluation kit. You can also manually edit *lsvendor.c*, *lm_code.h* and the *makefile* should the need arise.

2.1.1 Example Unix install_flexlm.ftp Session (highlights)

This example illustrates the use of `install_flexlm.ftp` to complete binary kit installation for a sun4, sunOS4 kit. Data you enter is typed in **bold**. The example is edited, but includes all prompts that you must enter. Skipped output is indicated with "[...]"

```
% ls
install_flexlm.ftp machine_ind.tar sun4_u4.tar
% sh install_flexlm.ftp
Installation for FTP FLEXlm
[...]
Extracting from machine_ind.tar
[...]
Is this is a Evaluation installation? (Y/n) y
[...]
Your demo kit is for the ../sun4_u4 platform
[...]
Do you want to configure your daemon now [y/n]? y
[...]
FLEXlm vendor key #1 for demo [hexadecimal]: 12345678
FLEXlm vendor key #2 for demo [hexadecimal]: 12345678
FLEXlm vendor key #3 for demo [hexadecimal]: 12345678
FLEXlm vendor key #4 for demo [hexadecimal]: 12345678
FLEXlm vendor key #5 for demo [hexadecimal]: 12345678
Do you want to continue and build the daemon now? [y/n, default: n]: y
[...]
Building kit in directory ../sun4_u4
cc -c -g -I../machind -DSUNOS4 ../machind/makekey.c
[...]
```

2.2 Installation for Windows, NT, and Netware Systems

1. Connect to Globetrotter's download site:

   `http://www.globetrotter.com/lmpostreg.htm`

2. Download the following files into your local directory (use BINARY mode):

   `flexlm.exe(32-bit)`

flexlm.exe also contains all the platform independent files.

To install, run the `flexlm.exe` program; it will expand into all files needed to install FLEX*lm*. If you are evaluating flexlm, answer YES when asked if this is a demo installation.

The installation locations closely parallel the Unix structure. A single `machind` directory will be created with all of the machine independent files and documentation. A machine dependent directory will be created for each platform during installation. `i86_n3` stands for Intel 32-bit (NT, Windows 95/98 32-bit)

For your convenience, we have pre-built all applications and utilities to assist in evaluating the product.

During the installation, a `BUILD.BAT` file will be created in each machine dependent directory, which is designed to run correctly with MS VC++. Running VCVARS32.BAT for VCC 2.0 will normally set up the necessary environment for BUILD.BAT. If you have environment variables set up correctly for MS Visual C++, then the `BUILD.BAT` file will be able to compile all necessary programs.

Although the FLEX*lm* license server relies on Microsoft Visual C++, any application can be a FLEX*lm* client if it is written in a language that can interact with the FLEX*lm* client DLL LMGR327x.DLL (which is a standard Windows WIN32 DLL).

All Documentation is provided on-line in the HTMLMAN directory and can be accessed by running any HTML browser.

2.2.1 What's installed

The following directories are available

i86_n3	Windows executables and files necessary for evaluation
htmlman	Browser documentation
machind	Machine independent files, include lm_code.h, manuals in PDF format, and sample programs. These files are identical on Unix
examples	More example programs

The following files will be loaded into the C:\Program Files\FLEXlm\v7.0\i86_n3:

demo.exe	Evaluation vendor daemon
flsetup.exe	FLEXlock setup program
genlic32.exe	Evaluation program for making license files
installs.exe	NT service installer for lmgrd.exe
license.dat	Sample license, including licenses for f1 and f2
lmclient.exe	Sample FLEX*lm* client program, using Trivial API
lmcrypt.exe	FLEX*lm* license generation program
lmgrd.exe	FLEX*lm* license manger

lmtools.exe	GUI tool for testing the FLEX*lm* license manager
lmwin.exe	Sample FLEXlm Windows GUI application
lmutil.exe	Command line (DOS prompt) tool for testing the FLEX*lm* license manager
makekey.exe	Sample program for making license files

Chapter 3

Evaluating FLEX*lm* and Your First FLEX*lm* Application

This chapter will walk you through the process of integrating FLEX*lm* into a small application. Before you start this chapter, please install the FLEX*lm* software as described in Chapter 2, “Installing the Distribution Kit” on page 12.

The FLEX*lm* kit has pre-configured source files to build a vendor daemon (demo), sample client program (lmclient), and license generation utility (lmcrypt). If your demo kit has expired, you will have to enter the new FLEX*lm* vendor keys provided by Globetrotter Software before you re-build these components.

If you’re familiar with FLEX*lm* on Unix or are comfortable with commands in a DOS window, the commands available on Unix are available on Windows and behave the same, so you can try , “Evaluating FLEXlm on Unix” on page 16. Otherwise, Windows users should following the evaluation instructions in , “Windows demo kits—Pre-configured Demo” on page 19.

Note that FLEX*lm* demo kits expire (the expiration date is a function of the FLEX*lm* vendor keys that were issued).

3.1 Evaluating FLEX*lm* on Unix

Change directories (cd) to flexlm/v7.0/*platform*, where *platform* is the technical name for your system, e.g., sun4_u5 is solaris, hp700_u9 is HP, and so on. Make sure ..\machind\lm_code.h has the correct VENDORKEYs. Run

```
make
```

To ensure that all the binaries are up to date in the *platform* kit directory.

COMMAND SUMMARY

You’ll be running the following commands to evaluate FLEX*lm*:

```
lmcrypt license.dat           #      generate license file)
lmgrd -c license.dat -l lmgrd.dl# Start license server
lmclient                      # run sample application
lmstat -a -c license.dat      # Check server status
lmdown -c license.dat         # Stop license server
```

USING LMCRYPT ON LICENSE.DAT

A sample license file is pre-shipped in the kit directory (e.g., sun4_u5/license.dat):

```
SERVER myhost ANY
VENDOR demo /t/tmp/flexlm/v7.0/sun4_u5/demo
FEATURE f1 demo 1.0 permanent 4 E4123D3F54B8
```

However, the license-key in this file may not be up-to-date. Run

```
% lmcrypt license.dat
```

to ensure the license-key is up-to-date.

START THE LICENSE SERVER

Run

```
% lmgrd -c license.dat -l lmgrd.dl
```

to start the license server. This is required because the license is *counted*. For *uncounted* licenses the license server is not needed. The lmgrd debug log file will be located in lmgrd.dl. At this point it will look something like:

```
11:11:49 (lmgrd) FLEXlm (v7.0c) started on myhost (Sun) (9/3/1999)
11:11:49 (lmgrd) FLEXlm Copyright 1988-1999, Globetrotter Software, Inc.
11:11:49 (lmgrd) US Patents 5,390,297 and 5,671,412.
11:11:49 (lmgrd) World Wide Web:  http://www.globetrotter.com
11:11:49 (lmgrd) License file(s): license.dat
11:11:49 (lmgrd) lmgrd tcp-port 27000
11:11:49 (lmgrd) Starting vendor daemons ...
11:11:49 (lmgrd) Started demo (internet tcp_port 34641 pid 9159)
11:11:49 (demo) FLEXlm version 7.0c
11:11:49 (demo) Server started on myhost for:    f1
```

RUN SAMPLE APPLICATION, LMCLIENT:

```
lmclient
```

```
Enter feature to checkout [default: "f1"]: <=Enter
f1 checked out...press return to exit...<=Enter
```

lmgrd.dl will now have at the bottom

```
11:22:35 (demo) OUT: "f1" daniel@myhost
11:22:40 (demo) IN: "f1" daniel@myhost
```

RUN LMSTAT FOR LICENSE SERVER STATUS

Run

```
lmstat -a -c license.dat
```

and you should see output like:

```
License server status: 27000@myhost
       License file(s) on myhost: /flexlm/v7.0/sun4_u5/license.dat:
       myhost: license server UP (MASTER) v7.0
Vendor daemon status (on myhost):

       demo: UP v7.0
Feature usage info:
Users of f1:  (Total of 4 licenses available)
```

If you run “lmstat -a -c license.dat” *while* lmclient has a license checked out, you’ll see something like:

```
Users of f1:  (Total of 4 licenses available)
  "f1" v1.0, vendor: demo
  floating license
    tom myhost 19.16.18.26 (v1.0) (myhost/27000 102), start Fri 9/3 7:29
```

FLEX*LM* ERRORS

Try checking out an unlicensed feature:

```
Enter feature to checkout [default: "f1"]: <myfeat=
Checkout failed: License server does not support this feature
Feature:       myfeat
License path:  @localhost:license.dat:./*.lic
FLEXlm error:  -18,147
For further information, refer to the FLEXlm End User Manual,
available at "www.globetrotter.com".
```

On windows, the error message appears by default in a popup window.

STOP LICENSE SERVER

Run

```
lmdown -c license.dat
    Port@Host         Vendors
1) 27000@myhost       demo
Are you sure (y/n)?  y<=E
Shut down FLEXlm server on node myhost
1 FLEXlm License Server shut down
```

The lmgrd.dl log file will have text at the end like

```
11:35:46 (lmgrd) SHUTDOWN request from daniel at node myhost
11:35:46 (lmgrd) lmgrd will now shut down all the vendor daemons
11:35:46 (lmgrd) Shutting down demo
11:35:46 (demo) daemon shutdown requested - shutting down
```

CHANGING THE LICENSE

Edit the license file, copy the FEATURE line for f1, so that it's now duplicated. On the 2nd FEATURE line, change f1 to my feat. Then run

```
lmcrypt license.dat
```

The license should now look like:

```
SERVER myhost ANY
VENDOR demo /flexlm/v7.0/sun4_u5/demo
FEATURE f1 demo 1.0 permanent 4 1F0E784111C6
FEATURE myfeat demo 1.0 permanent 4 E92DCA5E9ADD
```

If you restart the server (lmgrd command above) and run lmclient, asking for *myfeat*, it should succeed now.

Note that if you don't run lmcrypt on the edited license, you'll get:

```
"Invalid (inconsistent) license key"
```

which is the error end-users get if they attempt to alter a license file, or if they type it in incorrectly.

Don't forget to shutdown the demo server when you're done evaluating.

3.2 Windows demo kits—Pre-configured Demo

A pre-configured set of executables is automatically installed by `flexlm.exe`. Follow these steps to evaluate FLEX*lm*.

3.2.1 Start the license server with lmtools.exe

The evaluation kit automatically installs a sample license file and configures the registry to know about this file.

1. Launch lmtools.exe. You can do this from the Start Menu with

   Start->Programs->FLEXlm v7.0->FLEXlm Utilities

   or by simply executing lmtools.exe.
2. Click on "'Configuration using Services" and make sure "Demo License Manager" is highlighted. (This should have been installed during your installation. If not, you can create an entry here with "Configure Services" in this utility.)
3. Click the "Start/Stop/Reread" tab
4. Click "Start Server".

3.2.2 Check Server Status

To make sure the server is running, click:

Server Status->Perform Status Enquiry

At the bottom of the output window, it should say:

```
       demo: UP v7.0
Feature usage info:
Users of f1:  (Total of 4 licenses available)
Users of f2:  (Uncounted, node-locked)
```

3.2.3 Run sample application—lmwin.exe

1. Launch lmwin.exe. You can do this from the Start Menu with
   Start->Programs->FLEXlm v7.0->FLEXlm Test Program
   or simply executing lmwin.exe. By default, the feature that appears in the box is "f1". This is a "floating" license.
2. Click on Checkout.

   If successful, you rerun Server Status in lmtools, you will see at bottom of the output like this:

   ```
   Users of f1:  (Total of 4 licenses available)
     "f1" v1.0, vendor: demo
     floating license
         daniel myhost myhost(v1.0) (myhost/27000 101), start ...
   ```

   If you examine lmgrd's debug log file, Program files\FLEXlm\v7.0\i86_n3\lmgrd.dl, you'll see at the end of it something like:

   ```
   19:03:54 (demo) OUT: "f1" daniel@myhost
   ```
3. Click on Checkin, to free the license. The status and log will be updated automatically.
4. Edit the "f1" in lmwin and change it to "f2". This is an "uncounted, nodelocked" license and does not require the license server. Checkout will always work, even if the server is not running.

3.2.4 Stop license server with lmtools

In lmtools, select Start/Stop/Reread, then Stop Server. After completion you can check server status, and it should report something like:

```
lmgrd is not running: Cannot connect to license server
  The server (lmgrd) has not been started yet, or
  the wrong port@host or license file is being used, or the
  port or hostname in the license file has been changed.
Server name:   myhost
License path:  @myhost
FLEXlm error:  -15,10.  System Error: 10061 "WinSock: Connection refused"
For further information, refer to the FLEXlm End User Manual,
available at "www.globetrotter.com".
```

3.2.5 lm_code.h

Edit lm_code.h in the C:\Program Files\FLEXlm\v7.0\machind directory and insert your demo vendor codes. (If you do not have demo vendor codes, contact Globetrotter Software to receive your demo vendor codes.) You can perform this step later if desired, however, if you get an expiration date error when running the sample application, be sure to edit lm_code.h with your demo keys and run build.bat to rebuild the applications.

3.2.6 Generating and testing your own licenses

These examples require using genlic32.exe, which in turn requires a correct lm_code.h file. If lm_code.h is not correct, you can still generate licenses with makekey.exe or lmcrypt.exe, though the descriptions below or only for genlic32.exe.

TEST AN “ANYWHERE” LICENSE

genlic32.exe is a visual license generation program provided with the evaluation kit. Eventually, you’ll want to learn to use lmcrypt.exe, but genlic32 is easier to get started with. The “Run Anywhere” license is a license that will run on any computer. Choices of expiration date allow the license to be used for a limited or unlimited amount of time.

First, make a “Run Anywhere” license using genlic32.exe. Select

Start->Programs->FLEXlm v7.0->Evaluation License Generator

or simply start genlic32.exe.

Choose the *Run Anywhere* radio button to set the license type. Enter the feature name, “f1,” in the Feature Name edit field. Click the *Permanent* check box to make a non-expiring license.

Click the *Make License* button. The text of the license file will appear in the window. (To create multiple features, edit the form and click *Make License* again. Each feature will be appended to the window — but don’t do this now.) Type “gen.lic” in the “License File” edit field and click *Save* to create a license file called gen.lic. For more information, see “genlic32.exe (Windows only)” on page 56.

Next, run the program lmwin.exe, if it isn’t already running, located in the FLEX*lm* program group. Click the *Checkout* button. Note that the display window states that the checkout has succeeded. To release the license, press the *Checkin* button.

TEST A NODE LOCKED LICENSE

A Node Locked license is a license that only runs on a specific machine. Launch genlic32 if it is not already running, or click the *CLEAR* button at the bottom if it is still running from the prior test. Choose the *Node Locked* radio button to set the License Type. Enter the feature name “f1” in the Feature Name edit field. Click the *Permanent* check box to make a non-expiring license.

Click the drop-down button in the *Node Lock Host ID* edit box. Choose the DISK_SERIAL_NUM entry. This is the disk volume serial number for Drive C.

Click the *Make License* button. The license line will appear in the box. Make sure that gen.lic is in the License File edit field. Click *Save* to create a license file called gen.lic\license.dat. When the dialog warns you that you are overwriting the file, click *YES* to create a new file.

Next, run the program *Test Application* located in the FLEX*lm* program group (if it's not already running). Choose the 32 bit version for running on Windows 95 or NT.

Use Checkout and Checkin in the lmwin window to try the license. This license will only work on this computer because you have specified the host ID of DISK_SERIAL_NUM with a value specific to this computer. If you were to run the test application with this license file on another computer, it would not run because that computer would not have this same host ID.

TEST A FLOATING LICENSE.

A floating license is a type of license sometimes called, "Concurrent Licensing" or "Server Based Licensing." A floating license can be used by different people on a network.

Launch GENLIC32 if it is not already running, or click the *CLEAR* button at the bottom if it is still running from the prior test. Choose the *Floating* radio button to set the License Type. Enter the feature name "f1" in the *Feature Name* edit field. Click the *Permanent* check box to make a non-expiring license. In the *Number of Licenses* edit field, enter "1". In the *Server Name* edit field enter the name of your computer (this will be filled in if TCP/IP is properly installed on your computer). In the *Server Host ID* edit field, pull down the menu and select *Any.*

Click the *Make License* button. The license line will appear in the box. Make sure that gen.lic is in the License File edit field and then click the *Save* button to create a license file called gen.lic. When the dialog warns you that you are overwriting the file, click *YES* to create a new file.

Next, run the license server using lmtools. Select Configure Services, and enter

Service Name	mytest
Path to the lmgrd.exe file	C:\Program Files\flexlm\v7.0\i86_n3\lmgrd.exe
Path to the license file	C:\Program Files\flexlm\v7.0\i86_n3\gen.lic
Path to the debug log file	C:\Program Files\flexlm\v7.0\i86_n3\lmgrd.dl

Click "Save Service. Then, in Start/Stop/Reread, click "Start Server" run lmwin (if it's not already running). Checkout and checkin the license. To test that the server is actually counting the licenses, bring up another instance of the test program. Check out one feature from the first of the test applications, and then try to check out one feature from the other test application. The server will reject the second license check out.

Finally, stop the server by clicking the *Stop Server* button in the lmtools window.

3.3 Unix Demo kits—Incorporating FLEX*lm* into your Application

After you have become familiar with lmgrd, the vendor daemon (demo), lmcrypt and the sample client program, you might want to incorporate the FLEX*lm* client calls into your own software. If you wish to do this, follow these steps:

1. Modify your application code to call the license manager client routines. Add calls like the following to your application code:

   ```
   #include "lmpolicy.h"
    /*...*/
    CHECKOUT(LM_RESTRICTIVE, "myfeature", "1.0", “license.dat”);
    /*...*/
    CHECKIN();
   ```

   (see “lmclient.c” or Chapter 5, “Trivial API”, for a full description of the parameters).
2. Run “make” in the FLEX*lm* directory (e.g., flexlm/v6.1/sun4_u4), in order to re-build the daemon “demo” and lmcrypt. This is necessary to make *lm_new.o* (*lm_new.obj*) and for lmcrypt to use your new encryption seeds that you entered during the installation.

   Whenever you re-configure your daemon or update your FLEX*lm* encryption seeds, you must make sure to re-build lm_new.o, your application, vendor daemon, and lmcrypt so that they are built with the correct encryption seeds (in lm_code.h).
3. Build your application as usual, adding the following to your link (*ld* or *cc* on Unix) command:

   Unix: *lm_new.o* (created in step 2) and *liblmgr.a*

   Windows: *lm_new.obj* (created in step 2) and *lmgr.lib*
4. Run *lmcrypt* to create a new license file. You'll want to start with an existing file, using the files in examples/licenses.

If you get a link error complaining about missing “l_n36_buf”, the problem is that you need to add lm_new.o (lm_new.obj on Windows) to your link list.

3.4 Windows—Incorporating FLEX*lm* into your Application

This section presumes you're using Microsoft Visual C++. If you're using any other compiler or a different language, see , "Shared Library/DLL on Windows/NT" on page 27.

lmwin.exe is used as a sample application. The source is in `machind\lmwin.c`.

3.4.1 Step 1: Evaluate using the pre-built kit

On windows, integration with your C compilers can take some planning. For this reason, we recommend that you first familiarize yourself with the pre-built kit, as outlined above: Section 3.2,"Windows demo kits—Pre-configured Demo," on page 19.

3.4.2 Step 2: Edit lm_code.h

Make sure that `machind\lm_code.h` has correct vendorkeys. If this is a demo version, set only the vendorkeys to your evaluation vendorkeys. If it's a non-evaluation set of vendorkeys, in addition to changing the vendor keys, you must also change the `ENCRYPTION_SEEDS` to 2 32-bit numbers that you make up, and change the `VENDOR_NAME` to your vendor name. Keep the lm_code.h file and the ENCRYPTION_SEEDS secret. This file, lm_code.h is only used to build license generators (lmcrypt.exe, makekey.exe) and the lm_new.obj file. You do not need the lm_code.h file to build your licensed applications once the lm_new.obj file is built. So not all programmers in your company need access to lm_code.h, only the resulting lm_new.obj file.

If you have real vendor keys, we recommend that you edit i86_n3\makefile and make the following change

From:

DAEMON = demo.exe

to

DAEMON = *vendor*.exe

where *vendor* is the same as VENDOR_NAME in lm_code.h. Otherwise, after demo.exe is built by build.bat, you'll need to rename it to *vendor*.exe, where *vendor* is your vendor daemon name.

3.4.3 Step 3: C Development Environment

First, make sure that you have your Microsoft VC++ Development environment correctly configured, which is what is presumed for this manual. If you don't have this compiler, you'll have to use the FLEXlm DLL; See If you're running this from a DOS command window (which is usually the way this is done, but not necessarily), you may need to run vcvars32.bat to setup your development environment correctly. In a DOS window, you can easily see if your environment is set correctly: type “set” and make sure that MSVCDIR is set, and that PATH includes the BIN directory in MSVCDIR.

If you’re running from a “My Computer” window, your C Development environment (include MSVCDIR) must be part of your normal environment set in Windows 95 in autoexec.bat, or via the Control Panel on NT.

3.4.4 Step 4: Run build.bat

In i86_n3, run build.bat. This builds lm_new.obj, which is used to build your application, plus your vendor daemon as well as the sample applications lmclient.exe and lmwin.exe. Build.bat itself only calls nmake. If you’re familiar with nmake, you can run this directly. Nmake uses the i86_n3\makefile to build all files which are out-of-date. If you’ve just edited lm_code.h, it will rebuild everything except your application.

If you’ve done this correctly, you should be able to generate a license file, and use it with the sample applications, lmclient.exe or lmwin.exe.

3.4.5 Step 5: Add FLEXlm calls to your application

Add calls like the following to your application code:

```
#include "lmpolicy.h"
 /*...*/
 if (CHECKOUT(LM_RESTRICTIVE, "myfeature", "1.0", "license.dat"))
{
        PERROR("Checkout failed");
        exit(1);
}
 /*...*/
 CHECKIN();
```

(see `machind\lmclient.c` or Chapter 5, “Trivial API”, for a full description of the parameters).

For the rest of this example, we use machind\lmwin.exe as a sample application.

3.4.6 Step 6: Compile your object file

INCLUDE DIRECTORY

Add C:\Program files\FLEXlm\v7.0\machind to your INCLUDE path. If you use CL on the command line or in a batch file, add “/I C:\Program Files\FLEXlm\v7.0\machind” to your CL command line.

If you use the IDE (GUI development environment), add the INCLUDE directory in Tools/Options/Directories. Pick “Include files” from the choicelist, and add the full path to the machind directory.

/MD OR MT (MULTI-THREADED SHARED OR STATIC LIBRARIES)

FLEXlm requires multi-threaded libraries. You can use either the static (libcmt.lib) or shared (msvcrt.lib) multi-threaded C runtime libraries. The cl compiler /MT switch indicates static multi-threaded library (preferred) and /MD indicates shared, and one or the other must be used when compiling. If you indicate /MT, then you must link with libcmt.lib and lmgr.lib. If you indicate /MD, then you must link with msvcrt.lib and lmgr_md.lib.

If you use the command line, make sure /MT (or /MD) is included in the CL command line.

If you use the IDE, in Project/Settings pick the C/C++ tab. Then make sure that /MT (or /MD) is specified in the switches listed in the bottom window. In more current versions of VC++, there's a choicelist for multi-threaded.

Or, if you're not using /MT, link with the lmgr_md.lib library.

COMPILING LMWIN.OBJ ON THE COMMAND LINE

```
c:>cl /nologo /c /O1 /I../machind /MT  ../machind/lmwin.c
```

COMPILING LMWIN.OBJ IN THE IDE

1. Make an empty project:

   File -> New

   Project name: lmwin

   Pick "Win32 Application"

   Click OK
2. Add source file

   Project -> Add to Project -> Files...

   C:\Program Files\FLEXlm\v7.0\machind\lmwin.c
3. Add Include directory

   Tools -> Options -> Directories -> Include Directories

   C:\Program Files\FLEXlm\v7.0\machind
4. Specify Multi-threaded

   (MSVC++ v6+): Specify Multi-threaded

   Otherwise: Change /ML or /MD to /MT in the options window
5. Compile

   File -> Recent Files -> lmwin.c

   Build -> Compile lmwin.c

3.4.7 Step 7: Link

You need to add the following files to your link line:

lm_new.obj	has security information from lm_code.h, made by build.bat. (Or lm_new_md.obj if using /MD.)
lmgr.lib	FLEXlm library. (Or lmgr_md.lib if using /MD).
/nodefaultlib	A link switch that's needed to avoid a conflict.

In addition, the following MS libraries must be on the link line:

```
oldnames.lib  libcmt.lib kernel32.lib user32.lib netapi32.lib
advapi2.lib  gdi32.lib comdlg32.lib  comctl32.lib
```

COMMAND LINE EXAMPLE

On the command line, the lmwin link line looks like:

```
C:> LINK /nologo /NODEFAULTLIB  /out:lmwin.exe lmwin.obj lm_new.obj
lmwin.res lmgr.lib oldnames.lib  libcmt.lib kernel32.lib user32.lib
netapi32.lib advapi32.lib  gdi32.lib comdlg32.lib  comctl32.lib
```

IDE Settings

Change the following settings:

1. Library path

   Tools -> Options -> Directories

   Pick the Library Files choicelist, and add

   C:\Program Files\FLEXlm\v7.0\i86_n3

2. Libraries and options

   Project -> Settings -> Link

   Make sure the following are included in the list of object/library modules:

```
/NODEFAULTLIB lm_new.obj lmgr.lib oldnames.lib  libcmt.lib kernel32.lib
user32.lib netapi32.lib advapi32.lib  gdi32.lib comdlg32.lib
comctl32.lib
```

   If you're using shared C runtime library (msvcrt.lib), you'll need to make the following substitutions:

```
lm_new.obj     lm_new_md.obj
lmgr.lib       lmgr_md.lib
libcmt.lib     msvcrt.lib
```

3. Link

   Build -> Build lmwin.exe

3.5 Shared Library/DLL on Windows/NT

For supporting alternate languages, like Visual Basic, or a different C compiler, like Borland. Any compiler that can use Windows DLLs can use the FLEXlm DLL.

Caution: Use of the shared is a security risk. For this reason we recommend that you use the static library, as outlined above. One way to accomplish this, when you're not using MSVC++ is to use MSVC++ to make your own shared library, which your application would then interface to. This is somewhat more secure than using our standard shared library, as it is less standard, particularly if the DLL has other functionality is required by the application.

Nearly all the steps are same as the previous section. Following are the few differences:

3.5.1 FLEXLM_DLL

Caution: Using the FLEXlm DLL is less secure than using the static library, and should be avoided if possible. If your application does not use MSVC++, the way to use the static library is to use MSVC++ to make your own DLL, linking in the FLEXlm static library.

If you're using the Trivial or Simple API, when compiling your source code, you must make sure the FLEXLM_DLL is defined. You can do this by

command line	cl /DFLEXLM_DLL
in the source code	Add "`#define FLEXLM_DLL`" before any FLEX*lm* headers are included.
in the IDE	Project->Settings->C/C++->Preprocessor Definitions Add FLEXLM_DLL

3.5.2 lmgr327a.lib/dll

Link with lmgr327a.lib instead of lmgr.lib, as outlined in the section above. Ship lmgr327a.dll with your application. This must be in the same directory as your application, or in the users PATH.

3.6 Windows demo kits—Additional Information

The makefiles have been designed to work with MSVC++ Version 2.0 or greater. Other compilers may work, but will take additional time and effort to get a running product.

Each Unix platform normally supports one kind of hardware-based hostid. However, on the PC platforms, FLEX*lm* supports the following types of hostids:

- Intel CPU ID (Pentium III+ and FLEXlm v7.0d+ required). This hostid is probably preferable where avaialable, but since this turned off by default, the user must enable it using BIOS Setup.
- Disk Volume Serial Number (`DISK_SERIAL_NUM=`)
- FLEX*id* hardware key, often called a *dongle* (`FLEXID=`)

- Ethernet Address, default (12 hex characters, e.g., “1234567890ab”)

The decision to use a particular hostid is made in the license file. You do not need to do anything in your application to decide which hostid-type to use. For more hostid information refer to the FLEX*lm* Reference Manual.

3.7 Hostids for FLEX*lm*

A *hostid* is a means used to uniquely identify a specific machine. When you create a license with FLEX*lm*, you bind this license to a hostid. This binding allows only the authorized user(s) to run your software. The binding is created when you generate the *license file* for your customer. Section 10.4.1,“Simple Uncounted License,” on page 63, and Section 10.4.3,“Simple Floating (Counted) License,” on page 64 are two good examples of license files you might create to authorize your customer to run your software.

Chapter 4

Incorporating FLEX*lm* Into Your Application

To incorporate FLEX*lm* into your application, you will add function calls to your application program, build your application, and build a custom vendor daemon as discussed in the following sections.

4.1 FLEX*lm* Naming Conventions

All FLEX*lm* client routines and variables adhere to certain naming conventions. These conventions are:

- Trivial API functions are all uppercase MACROS defined in `lmpolicy.h`.
- Simple API function names start with "lp_". The 'p' stands for "policy", since this is policy-based licensing.
- FLEXible API client routine names start with "lc_".
- All FLEX*lm* server function and global data names start with "ls_". These routines are contained in the library files *liblmgr_s.a* and *liblmgr_as.a*, (lmgras.lib and lmgrs.lib for Windows Platforms) as well as in *liblmgrd.a*. Note that the *liblmgrd.a, lmgr_s.a* and *liblmgr_as.a* (lmgras.lib, lmgrs.lib) routines will not be linked into your application.

All FLEX*lm* client routines are contained in the library file *liblmgr.a* (Unix), *lmgr.lib (static) lmgr327a.dll*/*lmgr327a.lib, lmgr.lib (STATIC)* (Windows/32). In addition, clients need to link *lm_new.o* (*lm_new.obj,* Windows), part of FLEX*lm* security.

4.2 Installation and Directory Naming for Java

No special installation of the FLEX*lm* class files is required. The classes are in the java_01/flexlm directory of the FLEX*lm* v6.1 main directory. Note that the FLEX*lm* class files must reside in a directory called "flexlm", since they are in a Java package named "flexlm".

If you are running Java applications, you must set your CLASSPATH environment variable to include a component reflecting the location of the FLEX*lm* classes. For example, if you installed the FLEX*lm* classes into "/a/b/c/lmgr/v5.12/java_01/flexlm" you would include the following component in your CLASSPATH:

```
"/a/b/c/lmgr/v5.12/java_01"
```

If you are setting up the FLEX*lm* class files for access by an http server, they must reside in a directory called "flexlm", again because the FLEX*lm* class files are in the "flexlm" package.

4.3 Building your Vendor Daemon

To build your vendor daemon, see Chapter 8, "License Daemons" on page 50.

4.4 FLEX*lm* Example Applications

The FLEX*lm* distribution kit contains an example client application program in the *machind* directory called `lmclient.c`. This is a small stand-alone licensed program and is a good place to start to learn how to integrate FLEX*lm* with your application.

For Windows and Windows NT systems, the `machind\lmwin.c` example application which uses Microsoft Visual C++ to build a slightly more complicated example program to demonstrate a GUI Windows application.

The lmcrypt, makekey, and GENLIC programs can be used to generate licenses for your customers, or they can be used as examples of license generation programs. Source to the makekey and lmcrypt programs is in the machind directory.

The lmcrypt and makekey programs generate the same license keys on all FLEX*lm* supported platforms for all FLEX*lm* versions, allowing you to create license keys for any supported platform on any other supported platform.

FLEX*lm* kits also contain an `examples` directory at the top-level of the kit hierarchy. The `examples` directory contains *example* programs, which have been put on the kit to illustrate how to perform various operations with FLEX*lm*. These programs are **not supported** and GLOBE*trotter* Software may not include them in future FLEX*lm* releases.

4.5 Client Heartbeats and License Server Failures

Your client application will need to communicate regularly with the server via HEARTBEAT calls to ensure the server is still running. How the heartbeats occur and what action takes place when the server is not running are the most important part of incorporating FLEX*lm* in an application. This is addressed in the following sections:

- Section 5.6, "HEARTBEAT," on page 36
- Section 6.6, "lp_heartbeat," on page 41

4.6 License Policies

The Trivial and Simple APIs both require that you specify a license policy. A policy can be modified by ORing a list of optional modifiers. License policies and policy modifiers are described in the following sections.

4.6.1 LM_RESTRICTIVE

With this policy, any failure in the license, checkout, or server will be reported to the calling application as an error. The application decides what action to take with this error — it is not necessary that the application fail to run. For example, the application may report the error and continue running, it may exit, or it may run in a limited mode.

4.6.2 LM_QUEUE

This policy is the same as LM_RESTRICTIVE, except that the checkout call will wait for a license if the licenses are all currently in use. To the end-user, the application will appear to "hang" until the license is available.

4.6.3 LM_FAILSAFE

With this policy, the application will attempt a checkout, but failures of any kind will not be reported to the calling application. This policy provides "optional" licensing to the user. If the user wants to use licensing, he can, in which case the checkout will succeed. If the user doesn't want to use licensing, or if licensing is for some reason broken, applications will always continue to run.

In the case where all licenses are currently in use, the application will still run. The end-user could use FLEX*admin* to report on historical usage, which will show when licensed use is exceeded. Application users will never be denied usage. Errors that normally make a checkout fail are available as warnings.

4.6.4 LM_LENIENT

In this policy, if all licenses are in use, the checkout will return a failure status showing that all licenses are in use. For any other error, no error is returned. This is another form of "optional" end-user licensing, where the user is not penalized if licensing is not set up, or an operational error occurs. Errors that would normally make a checkout fail are available as warnings.

4.7 Policy Modifiers

These modifiers are binary ORed ("|") with the main policies, listed above. For example:

```
        LM_RESTRICTIVE | LM_MANUAL_HEARTBEAT
```

indicates that the main policy is LM_RESTRICTIVE, and the application will call HEARTBEAT() manually.

4.7.1 LM_MANUAL_HEARTBEAT

If this policy modifier is not specified, heartbeats, via *HEARTBEAT()* or *lp_heartbeat()* are automatically sent every 2 minutes from the application to the server. On Unix, SIGALRM is used to send the heartbeats.

If specified, LM_MANUAL_HEARTBEAT indicates that no automatic heartbeats should be sent to the server and the application will call *HEARTBEAT()* or *lp_heartbeat()* directly.

Most Unix applications will require LM_MANUAL_HEARBEAT, but some simple applications may prefer to have the heartbeats sent automatically. If your application does not send heartbeats to the license server, the application will not know if the license server has been shut down and restarted. If the server is restarted, the old copies of the applications continue running and a (new) full compliment of licenses becomes available, making license over-usage possible.

4.7.2 LM_RETRY_RESTRICTIVE

If this policy modifier is set, the application will exit with a short error message after 5 failed heartbeat messages. This is not normally recommended, but is useful for some simple applications.

4.7.3 LM_CHECK_BADDATE

If set, attempts are made o detect whether the user has set the system date back. This should be used in conjunction with setting ls_a_check_baddate to 1 in the machind/lsvendor.c file.

Chapter 5

Trivial API

5.1 Overview of Trivial API Calls

The Trivial API consists of macros that call the Simple API. What makes this API trivial is the simplified arguments to the macros. The only required header file is `lmpolicy.h`, and no other macros or function calls are needed.

CHECKOUT	Acquires a license
HEARTBEAT	Sends a heartbeat to the server
PERROR, PWARN	Presents current error/warning message to user
ERRSTRING	Returns a string describing the most recent error
WARNING	Returns a string describing the most recent warning
CHECKIN	Releases a license, and frees all FLEX*lm* memory

To use the Trivial API, simply include “`lmpolicy.h`” at the top of your source file. Only one feature can be checked out at a time from a single process.

Where possible, this is the preferred FLEX*lm* API.

Note: You cannot mix Trivial API calls with either Simple or FLEXible API calls.

5.2 Trivial API Example Program

The following is a complete example of the FLEX*lm* calls required in an application which uses the Trivial API:

```
#include "lmpolicy.h"
        /*...*/
        if (CHECKOUT(LM_RESTRICTIVE, "myfeature", "1.0", "license.dat"))
        {
                PERROR("Checkout failed");
                exit(-1);
        }
/*
 *      Checkout succeeded. Actual application code here
 */
        /*...*/
        CHECKIN();     /* Done with "feature", check it back in. */
```

5.3 CHECKOUT

SYNTAX

`status= CHECKOUT(policy, feature, version, license_path)`

DESCRIPTION

Acquires a license for a feature.

PARAMETERS

(int) *policy*	See Section 4.6,“License Policies,” on page 31. Example: LM_RESTRICTIVE.
(char *) *feature*	The feature name to check out.
(char *) *version*	The version of the feature to check out. This is a string in floating point format (e.g., “12345.123”). If the license in the license file has the same version number or a higher version number, the checkout will succeed. GLOBEtrotter recommends that this version number be a license version level and *not* the application’s version number. This version number should only be changed when you want old licenses to no longer work with a new version of the software.
(char *) *license_path*	The “default” location for the license file. If 0, this argument is unused.

The application will look in the following places for the license file:

- the location specified by the $*VENDOR*_LICENSE_FILE and/or $LM_LICENSE_FILE environment variable or registry settings.
- the *license_path* specified here.

Upon success, the path to the license file used is set in $VENDOR_LICENSE_FILE in the Registry on Windows (\HKEY_LOCAL_MACHINE\SOFTWARE\FLEXlm License Manager) and $HOME/.flexlmrc on Unix

RETURN

(int) *status*	0 if successful, otherwise the FLEX*lm* error number.

SEE ALSO

- Section 4.6,“License Policies,” on page 31
- Section 4.7,“Policy Modifiers,” on page 32

5.4 CHECKIN

SYNTAX

```
(void) CHECKIN()
```

DESCRIPTION

Releases the license for the feature and frees memory associated with the checkout.

5.5 ERRSTRING

SYNTAX

```
string = ERRSTRING()
```

DESCRIPTION

Returns a string describing the last FLEX*lm* error.

RETURN

(char *) *string*	An explanatory string.

5.6 HEARTBEAT

SYNTAX

```
status = HEARTBEAT()
```

DESCRIPTION

Exchanges heartbeat messages with the server. If the server goes down and later comes back up, *HEARTBEAT()* will automatically reconnect and check the license out. On failure, returns the number of failed attempts to reconnect to the server. On failure, applications should at a minimum notify the user of the failure. For a restrictive policy, applications may exit after a certain number of failures. In addition, applications may want to exit if reconnections succeed more than 3 or 4 times in a relatively short period (e.g. 10 minutes), which may indicate a user restarting the license server in an attempt to acquire extra licenses. Do not call *CHECKOUT()* on failure from *HEARBEAT()* — this is not necessary and will cause problems if attempted.

HEARTBEAT() should not be called unless LM_MANUAL_HEARTBEAT is set in the *CHECKOUT()* call. If LM_MANUAL_HEARTBEAT is not set, then *HEARTBEAT()* is called automatically.

RETURN

(int) *status*	0 if successful. Otherwise, it returns the number of failed attempts to reconnect to the server.

SEE ALSO

- Section 4.7.1,“LM_MANUAL_HEARTBEAT,” on page 32

5.7 PERROR

SYNTAX

```
PERROR(string)
```

DESCRIPTION

Presents *string* and a description of the most recent error to the user. On Windows this appears in a dialog; on other systems, it prints to stderr.

PARAMETERS

(char *) *string*	A string describing the error context.

5.8 PWARN

SYNTAX

```
PWARN(string)
```

DESCRIPTION

Presents *string*, plus a description of the most recent warning to the user. On Windows this appears in a dialog; on other systems, it prints to stderr. This is useful with policy set to LM_LENIENT or LM_FAILSAFE. Nothing is printed if there is no warning.

PARAMETERS

(char *) *string*	A string describing the error context.

5.9 WARNING

SYNTAX

```
string = WARNING()
```

DESCRIPTION

Returns a string describing the last FLEX*lm* warning.

RETURN

(char *) *string*	An explanatory string. This is useful with policy set to LM_LENIENT or LM_FAILSAFE

Chapter 6

Simple API

The Simple API can do nearly everything the FLEXible API can do. Use this API if your application requires checking out more than one feature name at a time or if you need to acquire more than one license for a feature.

This API requires that you include the "`lmpolicy.h`" header file.

6.1 Simple API Library Routines

lp_checkout	Acquires a license
lp_heartbeat	Sends a heartbeat to the server
lp_perror, lp_pwarn	Presents current error/warning message to user
lp_errstring	Returns a string describing the most recent error
lp_checkin	Releases a license, and frees all FLEX*lm* memory
lp_warning	Returns a string describing the most recent warning

Note: You cannot mix Simple API calls with either Trivial or FLEXible API calls.

6.2 Simple API Example Program

The following is a complete example of the FLEX*lm* calls required in an application that uses the Simple API. The primary differences between this and the Trivial API example are:

- The setup is a bit more complicated.
- The args to *lp_checkout()* are more complicated than *CHECKOUT()*.
- You can checkout more than one feature simultaneously, or more than one license of a feature (although neither of these are illustrated in the example).

```
#include "lmpolicy.h"
LP_HANDLE *lp_handle;
        /*...*/
        if (lp_checkout(LPCODE, LM_RESTRICTIVE|LM_MANUAL_HEARTBEAT,
                        "myfeature","1.0", 1, "license.dat", &lp_handle))
        {
                fprintf(stderr, "Checkout failed: %s",
                                        lp_errstring(lp_handle));
                exit(-1);
        }
/*
 *      Checkout succeeded. Actual application code here
 */
        /*...*/
/*
 *      Done with "feature", check it back in.
 */
        lp_checkin(lp_handle);
```

6.3 lp_checkout

SYNTAX

```
#include "lmpolicy.h"
LP_HANDLE *lp_handle;
status= lp_checkout(LPCODE, policy, feature, version, nlic,
                                license_path, &lp_handle)
```

DESCRIPTION

Acquires a license for a feature.

PARAMETERS

(`LPCODE_HANDLE` *) `LPCODE`	From the "`lmpolicy.h`" include file. Use the literal "`LPCODE`".
(int) *policy*	See Section 4.6,"License Policies," on page 31. Example: LM_RESTRICTIVE.
(char *) *feature*	The desired feature name to check out.
(char *) *version*	The version of the feature to check out. This is a string in floating point format (e.g., "12345.123"). If the license in the license file has the same version number or a higher version number, the checkout will succeed. GLOBEtrotter recommends that this version number be a license version level and *not* the application's version number. This version number should only be changed when you want old licenses to no longer work with a new version of the software.
(int) *nlic*	The number of licenses to check out. Usually this is 1.

(char *) *license_path*	The default location for the license file. The application will look in the following places for the license file: the location specified by the `$`*`VENDOR`*`_LICENSE_FILE` and/or `$LM_LICENSE_FILE` environment and/or registry variables, this default location, then the FLEX*lm* default (/usr/local/flexlm/licenses/license.dat or C:\flexlm\license.dat for PC's). It is highly recommended that this be set to a place in your product's installation hierarchy. The application may need to do some work to determine the exact path at runtime. If 0, this argument is unused.
(Pointer to LP_HANDLE *) lp_handle	This is the return handle, and is used for subsequent calls that apply to this checkout, e.g. *lp_checkin()*, *lp_errstring()*, etc. If *lp_checkout* is called more than once, separate *lp_handle* variables must be used, and the corresponding handle must be used with the other lp_xxx calls.

Upon success, the path to the license file used is set in $VENDOR_LICENSE_FILE in the Registry on Windows (\HKEY_LOCAL_MACHINE\SOFTWARE\FLEXlm License Manager) and $HOME/.flexlmrc on Unix (v7+).

RETURN

(int) *status*	0 if successful, otherwise FLEX*lm* error number.

6.4 lp_checkin

SYNTAX

```
(void) lp_checkin(lp_handle)
```

DESCRIPTION

Releases a license, and frees memory associated with the corresponding checkout. *lp_checkin()* should be called even if the checkout fails, in order to free associated memory and resources.

PARAMETER

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.

6.5 lp_errstring

SYNTAX

```
string = lp_errstring(lp_handle)
```

DESCRIPTION

Returns a string describing the previous FLEX*lm* error.

PARAMETER

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.

RETURN

(char *) *string*	Error description.

6.6 lp_heartbeat

SYNTAX

```
status = lp_heartbeat(lp_handle, num_reconnects, num_minutes)
```

DESCRIPTION

Exchanges heartbeat messages with the server. If the server goes down and later comes back up, *lp_heartbeat()* will automatically reconnect and check the license out. On failure, returns the number of failed attempts to reconnect to the server. On failure, applications should at a minimum notify the user of the failure. For a restrictive policy, applications may exit after a certain number of failures. In addition, applications may want to exit if reconnections succeed more than 3 or 4 times in a relatively short period (e.g. 10 minutes), which may indicate a user restarting the license server in an attempt to acquire extra licenses.

lp_heartbeat() should not be called unless LM_MANUAL_HEARTBEAT is set in the *lp_checkout()* call. If LM_MANUAL_HEARTBEAT is not set, then *lp_heartbeat()* is called automatically by the FLEX*lm* client library.

RETURN

(int) *status*	0 if successful. Otherwise, it returns the number of failed attempts to reconnect to the server.

PARAMETER

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.
(int *) num_reconnects	The number of reconnections in the last *num_minutes* minutes. This value is returned. If set to 0, no value is returned.
(int) num_minutes	Number of minutes for *num_reconnects*. If 0, num_reconnects is not returned.

RETURN

(int) *status*	0 if successful. Otherwise, it returns the number of failed attempts to reconnect to the server.

SEE ALSO

- Section 4.7.1,“LM_MANUAL_HEARTBEAT,” on page 32

6.7 lp_perror

SYNTAX

```
lp_perror(lp_handle, string)
```

DESCRIPTION

Presents *string* and a description of the most recent error to the user. On Windows this appears in a dialog; on other systems, it prints to stderr.

PARAMETERS

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.
(char *) *string*	A string describing the error context.

6.8 lp_pwarn

SYNTAX

```
lp_pwarn(lp_handle, string)
```

DESCRIPTION

Presents *string* and a description of the most recent warning to the user. On Windows this appears in a dialog; on other systems, it prints to stderr. This is useful with policy set to LM_LENIENT or LM_FAILSAFE. Nothing is printed if there is no warning.

PARAMETERS

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.
(char *) *string*	A string describing the error context.

6.9 lp_warning

SYNTAX

```
string = lp_warning(lp_handle)
```

DESCRIPTION

Returns a string describing the last FLEX*lm* warning.

PARAMETERS

(LP_HANDLE *) lp_handle	The handle from the *lp_checkout()* call.

RETURN

(char *) *string*	An explanatory string. This is useful with policy set to LM_LENIENT or LM_FAILSAFE

Chapter 7

Java API

The Java implementation of the FLEX*lm* client library allows applets and applications written in Java to use FLEX*lm* licensing. The Java API is similar to the Simple API.

Note: Any of your Java classes which invoke FLEX*lm* methods must "import flexlm.*;".

7.1 Methods

The license class has the following methods. All methods are instance methods of the "license" class.

7.1.1 license class constructor

SYNTAX

```
public license(String name, int d1, int d2, int k1, int k2, int k3, int
k4)
```

DESCRIPTION

Creates an instance of the license class, using the specified vendor name, encryption seeds, and vendor keys.

PARAMETERS

String name	the vendor daemon name
int d1	the XOR of vendor key 5 and encryption seed 1
int d2	the XOR of vendor key 5 and encryption seed 2
int k1	vendor key 1
int k2	vendor key 2
int k3	vendor key 3
int k4	vendor key 4

Note: The encryption seeds are numbers you make up and keep secret. The vendor keys are numbers given to you by Globetrotter. See Section Chapter 2,"Installing the Distribution Kit," on page 12 for more information.

Note: It is wise not to store your encryption seeds in variables, but rather to mention them only in an expression where they are XOR'd with vendor key 5. Given that expressions are evaluated at compile time, this makes it harder for somebody to discover your encryption seeds by decompiling your classes.

7.1.2 checkout

Syntax

```
public int checkout(int policy, String feature,String version,
                                                 int nlic, String licpath);
```

Description

Acquire a license for a feature.

Parameters

int policy	See, "License Policies" on page 31 for a list of valid policies. Example LM.RESTRICTIVE.
String feature	The feature name to be checked out.
String version	The version of the feature to check out. This is a string in floating point format (e.g., "12345.123"). If the license in the license file has the same version number, or a higher version number, the checkout will succeed. GLOBEtrotter recommends that this version number be a license version level and not the application's version number. This version number should only be changed when you want old licenses to no longer work with a new version of the software.
int nlic	The number of licenses to check out. Usually this is 1.
String licpath	The location of the license file. This may be a filename, a [port]@host specification, or a colon-separated list of filenames and/or [port]@host specifications. Note that there is no Java equivalent for environment variables, so LM_LICENSE_FILE is not used in the Java version.

Return

int	0 if successful, else FLEX*lm* error number.

7.1.3 checkin

Syntax

```
public void checkin()
```

DESCRIPTION

Checks in (releases) the licenses acquired with checkout().

PARAMETERS

none

RETURN

none

7.1.4 heartbeat

SYNTAX

```
public int heartbeat(int[] ret_reconnects, int num_minutes)
```

DESCRIPTION

Exchanges heartbeat messages with the server. If the server goes down and later comes back up, heartbeat() will automatically reconnect and check the license out. On failure, returns the number of failed attempts to reconnect to the server. On failure, applications should at least notify the user of the failure. For a restrictive policy, applications may exit after a certain number of failures. In addition, applications may want to exit if reconnections succeed more than 3 or 4 times in a relatively short period, say 10 minutes, which may indicate a user restarting the license server in an attempt to acquire extra licenses.

Note: There is no automatic heartbeat in FLEX*lm*/Java; the heartbeat() method *must* be called periodically by your Java code.

PARAMETERS

int[] ret_reconnects	is an array of length 1, or null. If non-null, the number of reconnects in the last “num_minutes” minutes is returned in the 0’th element of this array.
int num_minutes	the number of minutes over which to count reconnections returns: The number of unsuccessful reconnection attempts since the last successful heartbeat. Zero indicates that the last heartbeat was successful. Note that if the license policy is LM.RETRY_RESTRICTIVE, the application exits after 5 unsuccessful heartbeat attempts.

RETURN

0 if successful; otherwise the number of failed reconnect attempts since the last heartbeat response was received.

7.1.5 get_errstring

SYNTAX

```
String get_errstring()
```

DESCRIPTION

Returns the string describing the most recent FLEX*lm* error. The returned string contains a text string describing the error, a “major” numeric error code corresponding to the error, and a “minor” numeric error code which indicates to Globetrotter technical support exactly where in the source code the error occurred.

7.1.6 warning

SYNTAX

```
String warning()
```

DESCRIPTION

Returns a string describing the most recent FLEX*lm* error, or null if no error has occurred. Similar to get_errstring(), but useful when the policy is LENIENT or FAILSAFE, and the result of get_errstring() is null.

7.2 Public Constants

All constants are declared “public static final int” in class LM.

7.2.1 Policies

These values are used in the *policy* argument in the checkout method. The meaning of these policies can be found in “License Policies” on page 31.

The four policies available in the JAVA API are: LM.RESTRICTIVE, LM.QUEUE, LM.LENIENT, and LM.FAILSAFE. The one policy modifier, which can be ORed with the policy, is LM.RETRY_RESTRICTIVE. Note that the JAVA policies substitute a “.” in place of the leading “_” for the Trivial/Simple APIs.

Note: The LM_MANUAL_HEARTBEAT modifier from non-Java FLEX*lm* is not implemented in FLEX*lm*/Java. All heartbeats are manual in FLEX*lm*/Java.

7.2.2 Error Codes

See Appendix D of the FLEX*lm Reference Manua*l for a list of FLEX*lm* error codes.

7.3 Java and Security

There are special security considerations for companies using FLEXlm for Java, since Java applications can be relatively easily decompiled. For this reason we recommend that licenses and license servers for Java applications be different than non-Java applications. Otherwise, compromising security through a Java application would imply compromised security for non-Java applications.

Chapter 8

License Daemons

8.1 lmgrd

The purpose of *lmgrd* is to:

- start and maintain all the vendor daemons listed in the VENDOR lines of the license file(s),
- refer application checkout (or other) requests to the correct vendor daemon, and
- establish and maintain communications between redundant servers.

lmgrd is a standard component of FLEX*lm* that neither requires nor allows for vendor customization. The license daemon allows the license file location and a few other parameters to be set by the end-user. These options are set by command line arguments when starting lmgrd. The most common command line for lmgrd is:

```
lmgrd [-app] [-c license_list] [-l logfile][-x lmdown|lmremove]
```

If *license_list* is more than once license file, it needs to be a list, separated by colon on Unix, or semi-colon on Windows. If a directory is specified, **.lic* in that directory is used.

A complete description of *lmgrd* options is contained in the FLEX*lm* Reference Manual.

Note: *lmgrd* is not used on Netware or VMS Systems.

8.2 Configuring Your Vendor Daemon

You have configured and built your vendor daemon if you followed the procedures in Chapter 2, “Installing the Distribution Kit” and Chapter 3, “Evaluating FLEXlm and Your First FLEXlm Application”. If you need to re-build your vendor daemon, you must supply the following information:

- Your encryption seed(s)
- Your FLEX*lm* vendor keys
- The name of the daemon — VENDOR_NAME in `lm_code.h`

Although normally not required, your vendor daemon may require customization by editing *lsvendor.c*

When you have completed the necessary edits to *lsvendor.c*, the makefile will create *lm_new.o* (lm_new.obj) and then build your vendor daemon and lmclient, the demo client application, and the license generators, lmcrypt and makekey. One of the functions of *lm_new.o* is to remove the vendor name and encryption seeds from the executables; they are constructed at run-time.

8.2.1 Building Your Vendor Daemon—UNIX Systems

To build your vendor daemon (assuming the FLEX*lm* kit is in `/usr/gsi/flexlm/v6.1):`

```
% cd /usr/gsi/flexlm/v6.1/arch_os
```

(Edit `lm_code.h`, and `lsvendor.c` if needed).

```
% make
```

8.2.2 Building Your Vendor Daemon — Windows NT Systems

To build your vendor daemon, first edit ls_vendor.c and lm_code.h, then enter the following commands (assuming the FLEX*lm* kit is in `C:\Program Files\FLEXlm\v7.0\`):

```
C:> cd \Program Files\FLEXlm\v7.0\i86_n3
C:> build
```

8.2.3 Building Your Vendor Daemon — Netware Systems

To build your vendor daemon, first edit ls_vendor.c and lm_code.h, then enter the following commands (assuming the FLEX*lm* kit is in `D:\FLEX_SDK`):

```
D:> cd \FLEX_SDK\i86_Z3
D:> build
```

Note: For Netware systems, you must have the Watcom compiler and Novell Netware SDK.

8.3 Redundant License Servers

FLEX*lm* supports a set of three license servers to be used in a redundant manner. If any two of the three license servers are up and running, the system is functional and hands out its total complement of licenses (Two out of three license servers is referred to as a “quorum”).

Note: The VMS and Netware versions of FLEX*lm* do not support redundant servers.

8.3.1 Selecting Server Nodes

If all end-user data resides on a single file server, there is no need for redundant servers, and GLOBE*trotter* Software recommends the use of a single server node for the FLEX*lm* daemons. If the end-user's data is split among two or more server nodes and work is still possible when one of these nodes goes down or off the network, then multiple server nodes can be employed. In all cases, an effort should be made to select stable systems as server nodes; in other words, do not pick systems that are frequently rebooted or shut down.

8.3.2 Generating a license file for redundant servers.

To generate a license file that uses redundant servers, specify three servers when you create your license. Unlike independent servers, each SERVER line will require a port number, which can be any number from 1024 to 32000 which is unused at the end-user site. Note that the use of two license servers is not recommended, since a license file with two servers would require that you always had both running.

When redundant servers are started, they elect a *master,* which performs all licensing operations. The other one or two servers are there to provide a secure licensing mechanism in the event of hardware failure or if the master server node needs to be rebooted. Should the master fail, if two servers are still running, one of the remaining two will be elected master, and licensing operations will continue.

The order of SERVER lines in the license file (for redundant servers) specifies the end-user's desired selection order for the master server node. If the order of the SERVER lines do not agree in all license files, FLEX*lm* uses alphabetical order to determine the master and the following messages are generated in the debug log file:

```
6/26 11:00 (lmgrd) License File SERVER line order mismatch.
6/26 11:00 (lmgrd) Using alphabetical order
```

If the server order does not match, the daemons will come up initially, but reconnection in the event of server node failure may not work, depending on which node fails and who was the master before the failure. If the automatic failover in the event of node failure is important, ensure that the order of the server lines is consistent on all server nodes.

When only two of the three license server machines are up, it is possible for the client to experience a timeout before connecting to the license server. Specifically, if the first license server in the license file is down, the client will timeout before attempting to connect to the second server in the license file. This timeout is set to 10 seconds by default, so there will be a 10-second delay before the license is granted. If the first server is to be down for a while, the order of the SERVER lines in the license file which the client reads could be changed to avoid this timeout.

Chapter 9

Software Vendor Utility Programs

9.1 makekey

The FLEX*lm* distribution kit includes the makekey utility program used for the creation of license files. makekey is the easiest way to get started, since it asks a few questions and creates a correct license file. Once you have become familiar with the license file format, which is described in detail in the FLEX*lm* Reference Manual, you may want to use the lmcrypt utility to generate your license keys from a template file.

makekey is a stand-alone license file generator. makekey can be used "as-is" to generate license files for your customers or it can be used as an example for you to create your own customized license file generation program. If customizing, note that the essential function call is *lc_cryptstr()*.

makekey asks a number of questions and then generates the license file for the specific end-user.

makekey allows you to enter all data for a customer's license file from scratch or use an existing customer license file to update the feature lines.

The license file is left in the current directory with the name `license.dat`.

Licenses can be generated to be compatible with older versions (i.e., not using any of the features of the newer versions) by using

```
% makekey -verfmt n
```

where n is 2,3, 4, 5, or 5_1.

Licenses can have shorter or longer lines with the -maxlen argument:

```
% makekey -maxlen n
```

This is normally used to generate files with short lines so they'll be less likely to have newlines inserted by mailers. For this a length of 50 is common.

9.2 lmcrypt

Once you know the format of the license file FEATURE or INCRMENT lines that you need to create, lmcrypt is an easier way to create your license keys than makekey. lmcrypt replaces the license key, which can be simply '0', in a license file with the correct key, which is then ready to ship to a customer. To use lmcrypt, you need to either understand the FLEX*lm* license file syntax or use an example license file. Examples are available in the *examples/licenses* directory.

Usage:

```
lmcrypt [files][-i infile] [-o outfile] [-verfmt n] [-maxlen n]
        [-e errfile]
```

If no input file is specified, or if specified as "-" or stdin, standard input is used. If no output file is specified, or if specified as "-" or stdout, standard output is used. *files* are read and written back in place. If no file arguments are specified, lmcrypt reads stdin and writes stdout. All license keys are recomputed. *lmcrypt* will only work on lines that have a daemon name matching the vendor's daemon name.

LIcenses can be generated to be compatible with older versions by using "-verfmt *n*", where *n* is 2, 3, 4, 5, 5_1 or 6. If this is not possible, an error is produced, and the affected license line is left unaltered.

The maximum line length is controlled with "-maxlen *n*". A value of 50 is commonly used to make shorter lines less subject to mailers inserting newlines.

The simplest way to use lmcrypt is:

- copy an existing good file to another name, say `newlicense`
- edit `newlicense`, and make any desired changes, such as changing the feature name, or number of licenses, or adding new features, and save the file
- type the following

  ```
  % lmcrypt newlicense
  ```

  newlicense is then filled with correct license keys and is usable by a customer

Comments are passed through unaltered.

See `lmcrypt.c` in the machind directory.

ERROR RETURNS

Errors are printed to *stderr*, or as specified with -e. Most errors will prevent generation of license keys and the text will be output unchanged from the input. An example of error reporting: If -e is not used, the error messages also appear at the top of the output file.

Input:

```
FEATURE f1 demo 1.a50 01-jan-99 0 0 HOSTID=08002b32b161
```

Error reported:

```
stdin:line 1:Bad version number - must be floating point number, with no
letters
```

SEE ALSO

- “The License File” on page 60

9.3 makepkg

The makepkg utility is similar to makekey and is used to make PACKAGE lines for license files. The reason this utility is separate from makekey is that you will often want to ship PACKAGE lines with your product and issue enabling FEATURE or INCREMENT lines later when the product is sold to individual customers.

The source for makepkg, `makepkg.c`, is in the machind directory. You are encouraged to modify this source as needed.

Note: PACKAGE lines can also be created with the lmcrypt utility, but not with makekey.

9.4 genlic32.exe (Windows only)

GENLIC.EXE is a visual license generation program provided with the evaluation kit. After experimenting with the license files that come with the demo, you may wish to generate you own license files using genlic32.exe. While GENLIC32 can generate most common license types, it cannot generate every license type which FLEX*lm* supports. The range of license types which GENLIC23 can generate has been limited in order to make it easier for you to get started. If you need a license which GENLIC32 cannot generate, you will need to use *lmcrypt* or the *lc_cryptstr() function*.

You must edit lm_code.h and insert your demo vendor codes to use GENLIC.EXE. If you don’t, GENLIC.EXE will run but will warn you that the licenses you generate will not be valid.

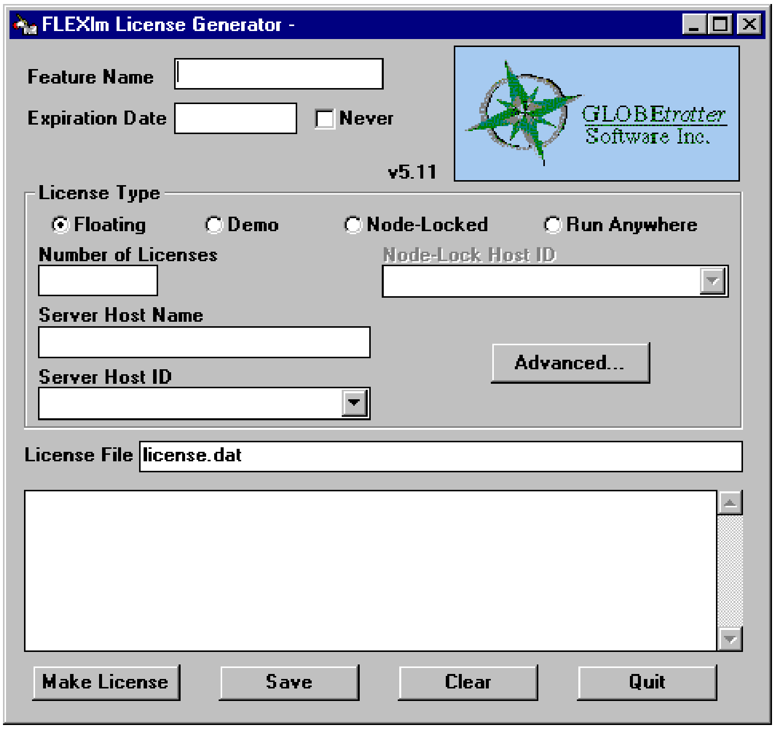
To use GENLICE.EXE, fill out the form for the type of license you want, then click the “Make License” button. The text of the license file will appear in the window. To create multiple features, edit the form and click “Make License” again. Each feature will be appended to the window. Click “Save” to save your work.

The main screen consists of the following:

Feature Name — name of the feature to be licensed. Characters must be alpha-numeric and/or ‘_’.

Expiration Date — date the license will expire. Valid date format is dd-mmm-yy[yy] (example 01-nov-2007 or 30-dec-2007). If the *Permanent* box is checked, the license will never expire and it has the date of 1-jan-0 (or “permanent”) in the license file.

License Type — any of the four following license types:



License Type: [✓ Floating]

License File: license.dat

Floating — Anyone on the network can use the licensed software, up to the allowed number of licenses. The floating license type requires the following fields:

> Number of Licenses — total number of licenses that can be checked out at any given time.
>
> Server Host Name — name of the license server.
>
> Server Host ID — Ethernet address or Disk Volume Serial Number of the license server. The information for your machine is available in the drop-down menu.

Demo — The licensed software will run on any system (it uses DEMO as the host ID). The licensed software is in DEMO mode.

Node-Locked — The licensed software can only run on one particular computer. The node-locked license requires the following field:

> Node-Lock Host ID - Ethernet address, Dongle ID, or Disk Volume Serial Number of a particular system on the network (workstation). This information for your system is available in the drop-down menu.

Run Anywhere — The licensed software will run on any system (it uses ANY as the host ID). The licensed software is in normal mode. (The difference between the ANY and DEMO host id is that the licensed software can check for demo feature lines, and then alter the behavior as desired.)

License File — name and path to the license file. To activate the BROWSE feature, leave the field blank and click on the SAVE button.

Make License — generate a feature line without saving to a file.

Save — write the licenses window to a license file.

Clear — reset the content of the licenses window.

Quit — exits the program.

Advanced — settings concerning the vendor information and license server data.

Version — Feature's version that is supported by the current license file.

Start Date—Date the current license file will take effect. This date will be authenticated.

Use Decimal Format — This allows you to generate the decimal format of the license file (easier to read numbers over the telephone)

License Sharing — This allows you to specify how multiple license requests share licenses.

None — Disable license sharing feature (default).

User — Allows multiple copies with the same user to share the same license.

Host — Allows multiple copies on the same computer to share the same license.

User and Host — Allows multiple copies of the same user on one computer to share the same license.

Site — Allows any system on the network to share the same feature (Unlimited use at one site).

Vendor Data (optional) — Information that can be recorded on the license line.

Optional—

TCP Port — Optional port number (> 1024 < 64000) to use on the SERVER line.

SPX Address — If using a FLEX*lm* Novell server, Port number for IPX/SPX protocol to use on the SERVER line.

Vendor Name: — "demo" by default. This is used in the VENDOR line. The name will change with modification to `lm_code.h` for VENDOR_NAME "....."

Vendor Daemon — Path to the vendor daemon. This is used in the VENDOR line.

Note: After the advanced or optional features are SAVE(d), they will be used for all future license files until they are changed.

9.5 Integrating the License Certificate Manager

The License Certificate Manager supports automatic downloading and installation of license files over the internet. This is enabled by default. It can be disabled using the FLEXible API, as described in the Reference Manual.

Chapter 10

The License File

10.1 Format of the License File

A license file consists of the following sections:

1. Optional license server section, with information about the node where the SERVER (or redundant SERVERs) is running, along with a list of all vendor-specific VENDOR(s) that run on the license server node(s). This section is required if any features are *counted*.
2. Optional USE_SERVER section, indicating that client applications should not process the rest of the license file, but should checkout the license directly from the server. Globetrotter recommends the use of USE_SERVER, particularly where performance is important.
3. Features section, consisting of any combination of FEATURE, INCREMENT, UPGRADE, or PACKAGE lines. This section is required for lmgrd. When client applications read a license file, this section is required, unless USE_SERVER is used.
4. Comments. The convention is to begin comments with a '#' character. However, in practice all lines not beginning with a FLEX*lm* reserved keyword are considered comments.
5. Long lines can be broken up. It is customary to use a '\' line continuation character, but in v7+ this is not required, particularly since newlines are often added by emailers.

Vendors and license administrators will read the license file to understand how the licensing will behave, e.g.: what features are licensed, the number of licenses, and whether these licenses are node-locked. The options are very broad for what can be specified in a license file.

End-users often need to edit the license file. Nearly all of the file is authenticated; if these portions are edited by the license administrator, an `LM_BADCODE` error will result. However, license administrators often edit the license file for the following reasons:

- To change the SERVER nodename
- To change the (optional) SERVER TCP/IP port number
- To change the (optional) path to the vendor daemon(s)

Any amount of white space of any type can separate the components of license file lines; data can be entered via any text editor. Vendors can therefore distribute license data via FAX or telephone.

The only data items in the license file that are editable by the end-user are:

- hostnames on SERVER lines
- (optional) port numbers on SERVER or VENDOR lines
- (optional) pathnames on VENDOR lines
- (optional) options file pathnames on VENDOR lines
- (optional) Decnet object numbers on VENDOR lines (VMS only)

Note: The SERVER hostid(s) and everything on the FEATURE line (except the daemon name) are input to the authentication algorithm to generate the license key for that FEATURE.

The License file format is documented in detail in the FLEX*lm* Reference Manual.

10.1.1 Example License File

The following example illustrates the license file for a single vendor with two features, and a set of three server nodes, any two of which must be running for the system to function:

```
SERVER pat 17003456 27009
SERVER lee 17004355 27009
SERVER terry 17007ea8 27009
VENDOR demo
FEATURE f1 demo 1.0 1-jan-1996 10 1AEEFC8F9003
FEATURE f2 demo 1.0 1-jan-1996 10 0A7E8C4F561F
```

10.2 Locating the License File

Client applications use the following rules for locating a license file.

1. If either $LM_LICENSE_FILE or $*VENDOR*_LICENSE_FILE (where *VENDOR* is the ISVs vendor-daemon name) environment variable is set, they are used. $*VENDOR*_LICENSE_FILE is used first. $*VENDOR*_LICENSE_FILE is used only by products from $*VENDOR* and by lmutil and lmtools.exe
2. LM_LICENSE_FILE and/or $VENDOR_LICENSE_FILE can also be set in the Windows registry or, on Unix, in $HOME/.flexlmrc. The Windows registry is in \HKEY_LOCAL_MACHINE\SOFTWARE\FLEXlm License Manager. These locations are also automatically set upon successful checkout.
3. The license location(s) can be set the *CHECKOUT()* or *lp_checkout()* calls. This is searched after $LM_LICENSE_FILE and/or $*VENDOR*_LICENSE_FILE.

4. FLEX*lm* utilities accept *-c license_file_path* argument, which specifies the license path(s). If this is set, the environment variables are ignored.
5. If none of the other locations are specified, then a “backup”/default location is used:

   Unix:

   ```
   /usr/local/flexlm/licenses/license.dat
   ```

   Windows:

   ```
   C:\FLEXLM\LICENSE.DAT
   ```

Note: In practice, use of the default location is discouraged, since it is not searched if the applications specifies a different default, or the user has a license-path environment variable set. That is, the default location, which may have successfully worked for an application, will fail once $LM_LICENSE_FILE is set (unless, of course, $LM_LICENSE_FILE includes the default location).

10.2.1 License specification

Wherever a license path can be specified, it can consist of

- a single file
- a list of files, separated by a colon on Unix, a semi-colon on Windows.
- a directory, where *dir*/*.lic are used in alphabetical order, as if specified like a license file list.
- *@host*, where *host* is the hostname of the license server, when the SERVER has no port number, or the number is between 27000 and 27009. (*New* in v6—unsupported in older versions.)
- *port@host*, where port is the port number and hostname come from the SERVER line
- *port@host,port@host,port@host*, when a 3-server redundant server is being used.
- The actual license file text, with “START_LICENSE\n” as a prefix, and “\nEND_LICENSE” as suffix, where the embedded newlines are required. While awkward to specify in most shells, this is most useful inside a program.

Examples:

```
% setenv LM_LICENSE_FILE license.dat
```

Only one file is specified.

```
% setenv LM_LICENSE_FILE license.dat:/usr/local/flexlm/licenses
```

license.dat and /usr/local/flexlm/licenses/*.lic are used.

```
% setenv LM_LICENSE_FILE @localhost:.:/proddir/licenses
```

If the server is running on the same system on a “default” port, @localhost will find it. After that it looks for *./*.lic* and /*proddir*/licenses/*.lic

Note: Before v5, both the client and server needed to read the *same* license file, since the client passes the license key from the FEATURE line to the vendor daemon. With FLEX*lm* v5+, port@host or USE_SERVER solves this problem, since the client never reads the license file features in these cases.

10.3 Hostids for FLEX*lm* Supported Machines

FLEX*lm* uses different machine identifications for different machine architectures. For example, all Sun Microsystems machines have a unique integer hostid, whereas all DEC machines do not. The program lmhostid will print the exact hostid that FLEX*lm* expects to use on any given machine. The FLEX*lm* Reference Manual lists alternate methods of determining the hostid for common machine architectures.

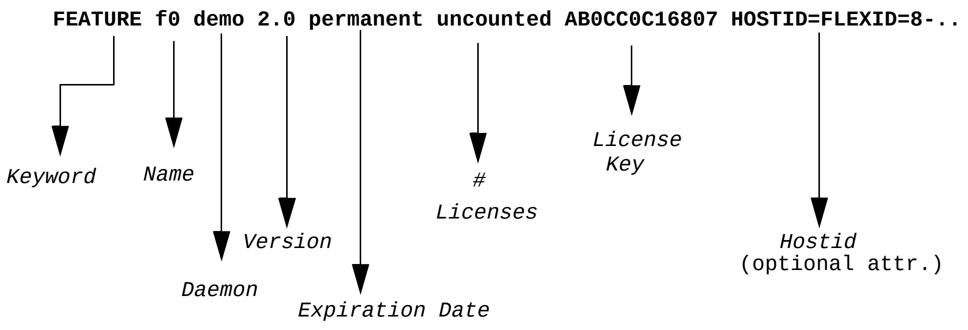
10.4 Types of License Files

Depending on the information in this file, the contents will be interpreted differently by FLEX*lm*. The license file supports network licensing, node locking, network licensing on a limited set of hosts, and demo/evaluation software.

Following are license file examples, starting with the simplest. In the examples, the changes from the previous example are in **bold** text.

10.4.1 Simple Uncounted License

```
FEATURE f0 demo 2.0 permanent uncounted AB0CC0C16807 HOSTID=FLEXID=8-...
```

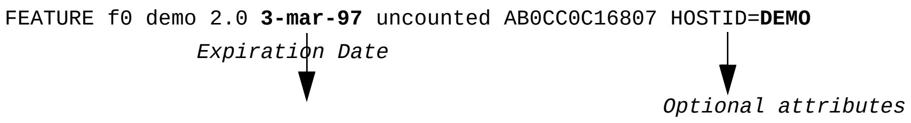


- Uncounted licenses have unlimited use on the hostid specified. Uncounted licenses require no server.
- When the expiration date is “permanent” (or if a date is specified with a year of 0), the license never expires.

- This license supports versions 0.0 through 2.0 (inclusive).

10.4.2 Expiring Demo License

```
FEATURE f0 demo 2.0 3-mar-97 uncounted AB0CC0C16807 HOSTID=DEMO
```


FEATURE f0 demo 2.0 3-mar-97 uncounted AB0CC0C16807 HOSTID=DEMO
Expiration Date
Optional attributes

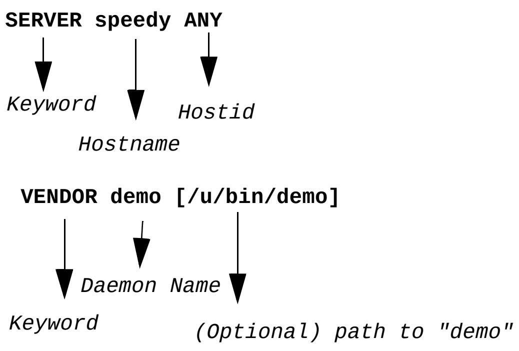
- This license expires on 3 March, 1997.
- If an expiration date has a 2-digit year, the actual expiration date is these two digits + 1900. You can use all four digits for this expiration date, so the example above could be written with an expiration date of “3-mar-1997”. For a license which expires after 1999, use all four digits, e.g. “1-jan-2001”.
- The DEMO hostid indicates that this license allows f0 to run on any system. In addition, the client application can also detect that it is in “demo” mode, and could behave differently.

10.4.3 Simple Floating (Counted) License

```
SERVER speedy 08002b32b161
VENDOR demo
FEATURE f1 demo 2.0 permanent 9 DBCC10416777
```

- Server and Daemon lines required
- Unexpiring
- Floating — runs on any node. No hostid on FEATURE line.
- Limited to 9 concurrent licenses.
- Server restricted to hostid *08002b32b161*. To remove this restriction, use hostid of ANY (e.g. SERVER speedy **ANY** 2837).

The breakdown of the SERVER and VENDOR lines is illustrated here:



- Hostname can be changed by the end-user. If hostname is "*this_host*", clients running on the same node as the server will work fine. Clients on other nodes will fail unless the hostname is changed, or the clients use @host (or port@host if a port number is specified on the SERVER line) to find the server.
- Path to daemon can be changed by the end-user. If unspecified, lmgrd uses the PATH environment variable, or the current directory to find the vendor-daemon binary.
- Nothing else can be changed on these 2 lines. Everything else is authenticated by the license key.

10.4.4 INCREMENT

```
SERVER speedy 08002b32b161
VENDOR demo
INCREMENT f1 demo 2.0 permanent 1 2B8F621C172C
INCREMENT f1 demo 2.0 permanent 2 2B9F124C142C
```

- INCREMENT — the server adds up licenses for all lines for the same feature name. The concurrent usage limit is 3 (= 1 + 2).
- The first INCREMENT line could be a FEATURE line and the behavior would be the same.
- INCREMENT lines must differ in some way — otherwise only one will be used.

10.4.5 INCREMENT, node-locked

```
SERVER speedy 08002b32b161
VENDOR demo
INCREMENT f1 demo 2.0 permanent 1 7B9F02AC0645 HOSTID=80029a3d
INCREMENT f1 demo 2.0 permanent 2 6BAFD2BC1C3D HOSTID=778da450
```

- One license is available on hostid 80029a3d.
- Two licenses are available on 778da450.
- The server tracks these licenses independently, in separate *pools*.
- This behavior ONLY works with INCREMENT, not FEATURE, since with FEATURE, the server only recognizes the first FEATURE line for a given feature name.

10.4.6 Mixed Floating (Counted) and Uncounted

```
SERVER speedy 08002b32b161
VENDOR demo
FEATURE f1 demo 2.0 permanent 1 7B9F02AC0645 HOSTID=80029a3d
INCREMENT f1 demo 2.0 permanent 2 6BAFD2BC1C3D HOSTID=778da450
FEATURE f0 demo 2.0 permanent uncounted AB0CC0C16807 HOSTID=554066fa
```

- Checkouts of f0, since it is *uncounted*, may not communicate with the server — they only verify that the client is on node 554066fa, and that the version is <= 2.0. If USE_SERVER is specified, or either *VENDOR*_LICENSE_FILE or

LM_LICENSE_FILE is set to @host (or port@host if a port number is specified on the SERVER line), then checkouts do require a server and their usage is logged.

- The f0 line does not require the SERVER or VENDOR lines, and in fact could reside in another license file altogether.

10.4.7 Optional FEATURE attributes

```
SERVER speedy 08002b32b161
VENDOR demo
INCREMENT f1 demo 2.0 permanent 4 DBCC10416777
INCREMENT f1 demo 2.0 permanent 3 BBDC1081492A
UPGRADE f1 demo 2.0 3.0 permanent 5 3B8C60B10227
FEATURE pkg2 demo 1.0 permanent 1 BB9C4071436D \
       VENDOR_STRING=vd HOSTID=12345678 OVERDRAFT=1 \
       DUP_GROUP=UHD ISSUER=issuer NOTICE=notice\
       START=1-jan-2000 vendor_info=vi dist_info=di\
       user_info=ui asset_info=ai ck=161
```

- Most optional attributes are in *keyword=value* format.

```
FEATURE pkg2 demo 1.0 permanent 1 BB9C4071436D \
 VENDOR_STRING=vd HOSTID=FLEXID=8-12345678 ...
```

Keyword
Value

- The following keywords (which are printed in lowercase by the license generators) can be modified by the user, and are *not* part of the license key: asset_info, dist_info, user_info, vendor_info, and ck.

ATTRIBUTES IN DETAIL

`VENDOR_STRING=vd`

The vendor-defined string is used for customization by the vendor, often to license subfeatures.

`HOSTID=FLEXID=8-12345678`

Use locked to hostid FLEXID=8-12345678 (a node with the hardware key with id 8-12345678 attached).

`OVERDRAFT=1`

Usage is limited to number-of-users (1) plus OVERDRAFT (1) = 2. The application can detect this state, and it is logged in the REPORTLOG.

`DUP_GROUP=UHD`

All usage by the same user on the same host and display are counted as a single use.

`START=1-jan-2000`

Optional start date.

`ISSUER=issuer NOTICE=notice vendor_info=vi dist_info=di\`

```
     user_info=ui asset_info=ai
```

Unused by FLEX*lm*. Can be used for customization by vendor or end-user.

```
ck=161
```

A checksum, used by the lmcksum utility to validate the line.

10.4.8 PACKAGE

```
PACKAGE pkg demo 1.0 504091605DCF\
       COMPONENTS="comp1 comp2 comp3 comp4 comp5 comp6 comp7 comp8"
FEATURE pkg demo 1.0 permanent uncounted DB5CC00101A7 HOSTID=778da450
```

The 2 lines above are a more efficient way of delivering:

```
FEATURE comp1 demo 1.0 permanent uncounted D03F02432106 HOSTID=778da450
FEATURE comp2 demo 1.0 permanent uncounted 99375F40FD85 HOSTID=778da450
FEATURE comp3 demo 1.0 permanent uncounted 68FAC130DB90 HOSTID=778da450
FEATURE comp4 demo 1.0 permanent uncounted D3D617E2075A HOSTID=778da450
FEATURE comp5 demo 1.0 permanent uncounted 5A91D6EFB68C HOSTID=778da450
FEATURE comp6 demo 1.0 permanent uncounted 8F75798EB975 HOSTID=778da450
FEATURE comp7 demo 1.0 permanent uncounted 790545E90575 HOSTID=778da450
FEATURE comp8 demo 1.0 permanent uncounted 9EE9E788087F HOSTID=778da450
```

- The Feature line *enables* the Package line.
- The COMPONENTS all inherit the information from the enabling FEATURE line. In this example, they all inherit the expiration date, number of licenses, and hostid.
- The enabling FEATURE line must match the name, version, and vendor name of the PACKAGE.
- The PACKAGE line is usually shipped with the product, since it contains no customer-specific fields.
- PACKAGE lines can be shipped in a separate file that never needs user editing, so long as the file is include in the license-file-list.

10.4.9 SUITE

```
PACKAGE office demo 1.0 00504091605D OPTIONS=SUITE \
               COMPONENTS="write paint draw"
FEATURE office demo 1.0 permanent 3 DB5CC00101A7
```

- There is no equivalent in FEATURE lines for this behavior.
- The client application checks out both a component feature, and, automatically, a copy of a feature called “office”. Without OPTIONS=SUITE, this additional checkout would not occur.
- A feature called “office” is created, in addition to all the components.
- This license file indicates that after *any* three licenses of any of the components are used, no further licenses are available for checkout. Without the OPTIONS=SUITE qualifier, there would be three licenses of *each* of the three components.

10.5 License in a buffer

The license file does not need to be located on disk—it can be specified in the program itself. The license-path in *CHECKOUT*, or *lp_checkout()* can specify the actual license, as in this example:

```
CHECKOUT(LM_RESTRICTIVE, "f1", "1.0",
        "START_LICENSE\n\
        FEATURE f1 demo 1.0 permanent \
        uncounted 50A35101C0F3 HOSTID=ANY \
        VENDOR_STRING=\"Acme Inc\"\n\
        END_LICENSE");
```

The license must begin with “START_LICENSE\n” and end with “\nEND_LICENSE”, where the embedded newlines are required.

This can also be a license-file-list; as in the following example:

```
CHECKOUT(LM_RESTRICTIVE, "f1", "1.0",
        "path/to/license.dat:START_LICENSE\n\
        FEATURE f1 demo 1.0 permanent \
        uncounted 50A35101C0F3 HOSTID=ANY \
        VENDOR_STRING=\"Acme Inc\"\nEND_LICENSE"
```

In this example, “path/to/license.dat” is first in the list, followed by the license in the string.

License in a buffer is particularly useful when selling libraries if a separate license file is not desirable, or as a final “fail-safe” license in the event that the license server is not running.

10.6 Decimal Format Licenses

Licenses can be represented in format, to make license delivery easier for customers without access to e-mail. Decimal has the advantage that it’s simpler to type in, and often the licenses are much shorter. There are notable exceptions, however, which are explained below.

To generate a decimal format license, use the *-decimal* arg for lmcrypt or makekey.

To convert an existing license to decimal, use lmcrypt -decimal, or

```
        % lminstall -i infile -o outfile -odecimal
```

If needed, decimal lines can be mixed with readable format lines.

End-users will normally use the lminstall command to install decimal format licenses. Note that lminstall converts the decimal lines to readable format. lminstall does not, however, know where your application expects to find the license file. You will need to make the license-file location clear to the user. Please see the FLEX*lm* Reference Manual for a more complete description of the decimal format. Refer to the FLEX*lm* End-User Manual for more information on lminstall.

SEE ALSO

- Section 13.3.5, “lminstall,” on page 83

Chapter 11

FLEXlock and License Certificate Manager (Windows only)

11.1 FLEXlock

FLEXlock is a new feature in FLEXlm designed to make it extremely easy to add “try before you buy” licensing to a product that uses FLEXlm. After enabling FLEXlock in a FLEXlm licensed product, FLEXlock automatically allows a trial period after the initial installation along with explanatory warnings and dialogs explaining the evaluation system to the user. At the completion of the trial period, the user has the option to either stop using or purchase the product. Upon purchasing, the user is given a license file that will enable your product to run in a normal FLEXlm manner. During the evaluation period it is possible to detect if the user has made unlicensed copies.

To use the FLEXlock functionality you need to:

1. Enable the functionality in your program.
2. Determine the operation and parameters of the “try before you buy” functionality.
3. Run flsetup.exe to define these FLEXlock parameters and create the FLEXlock distribution file “`fldata.ini`”.

11.1.1 Enabling the FLEXlock Features

To enable the functionality of the FLEXlock you will need to specify to FLEXlm that you will be permitting this functionality.

If you are using the TRIVIAL or SIMPLE APIs of FLEXLM, you will simply specify an additional License Policy of LM_FLEXLOCK to your checkout call, i.e.

```
CHECKOUT(LM_RESTRICTIVE | LM_FLEXLOCK, "myfeature", "1.0",
        "license.dat");
```

11.1.2 Security

We've made every effort to make the FLEXlock feature secure. However, due to the type of security technology used for FLEXlock, it is less secure than the rest of FLEXlm. This is why it is disabled by default. You should only enable FLEXLOCK if the convenience of FLEXlock licensing is more important than the reduced security it exposes your product to.

For additional security, you will need to use the FLEXible API, and LM_A_FLEXLOCK_INSTALL_ID and LM_A_FLEXLOCK, as outlined there.

11.1.3 The FLEXlock Configuration Editor

To define the operation of FLEXlock features, you must run the flsetup.exe program and generate the "fldata.ini" file. Flsetup.exe allows developers to quickly define:

- The type of product trial including:
  - fixed number of days
  - fixed numbers of executions
  - expiration on a fixed date
- How to behave when the trial expires:
  - don't run
  - run with a warning
- The greeting given when the product is first executed
- The greeting given during the trial phase
- The message given when the trial phase expires including instructions on how to purchase the product
- Product's Attributes:
  - product name
  - company name
  - copyright notice

After configuring these parameters and saving the file, run the "File->Create Distribution Files". This generates the `fldata.ini` file that will need to be shipped with the product. Also the `flcflxA.dll` file will need to be shipped with the product as well. These files should be placed in the same directory as your program executable.

FLEXlock has been designed to work with LCM, so software vendors may have a fully automated try-before-you-buy program, coupled to a web-based unlocking and payment system. With the use of GLOBEtrack Licensing, it is possible to obtain a license file over the internet, and have it automatically installed without user intervention of copying and pasting.

11.2 License Certificate Manager (LCM)

LCM requires use of GLOBEtrack Licensing. If the instructions in GT Licensing are followed correctly, the application will automatically support it.

When a license is unavailable for an application, a dialog appears and one option is to download the license from the internet. If they select this option, the URL and key are prompted for.

To demo the LCM, use the lmclient sample program with feature “lcm”.

The default URL for the LCM can be changed, but only with the FLEXible API. See LM_A_LCM_URL in the Reference Manual. By default the LCM URL is www.globetrotter.com/*vendorname*, where *vendorname* is your vendor daemon name.

Chapter 12

Integration Guidelines

The following sections describe some things to consider when you integrate FLEX*lm*.

12.1 Where to Install Your Licensing Software

When your installation procedure installs the FLEX*lm* software at your end-user site, there are some things you should keep in mind:

- All license server executables (lmgrd and vendor daemons) should be LOCAL on the system(s) that will run them. A corollary of this is that you should not run license servers on diskless nodes.
- There should be a *local copy* of the license file on each server node. It is fine to NFS-mount the license file for client access, but each server node should have a local copy. A better approach is to set either the *VENDOR*_LICENSE_FILE or `LM_LICENSE_FILE` environment variable to *@host* (or *port@host* if a port number is specified on the SERVER line) for all clients. In fact, for large license files, *@host* is more efficient, since it doesn't need to read the license file. For more information, see Section 10.2, "Locating the License File," on page 61.
- It is poor policy to configure redundant servers and then keep only one copy of either the daemons or license file — in this case you still have a single point of failure. Place a copy of lmgrd, your vendor daemon, and the license file(s) on the disk of each license server node.

12.2 Redundant vs. Single-server licensing

You will have to help your end-user decide how many server nodes to run, as the license keys are partially derived from the list of server node hostids.

One server node is recommended, unless the network and/or server machines on the network regularly go down. In this case, use three server nodes, so that any one server can go down and still allow the software to run. Note that the server nodes do not have to be of the same machine architecture; all FLEX*lm* platforms (except VMS) can operate in a heterogeneous manner.

12.3 Keeping Your Software Secure

No software is completely secure. FLEX*lm* is no exception. While GLOBE*trotter* Software has made every effort to ensure the integrity of FLEX*lm*, all points of attack can never be anticipated. The following lists known points of vulnerability in FLEX*lm* in increasing order of difficulty to break. Globetrotter Software also maintains a list of techniques for making your implementation more secure — please contact technical support (support@globes.com) for a description of these techniques.

EASY

- Running the debugger on the application code if it is released with unstripped executables (on Unix) or as a debug version (on Windows).

DIFFICULT, DEPENDING ON APPLICATION POLICY

- Killing the daemons, since a majority of daemons must be up in order for anything to run, and a dead daemon is detected within the timer interval in a client. If, however, you do not use one of the built-in timers and you do not call *HEARTBEAT()*, then your software protection could be bypassed by someone who kills the daemons each time that the application reaches the maximum license limit, as the applications would never detect that the daemon went down.

  To reduce the potential for theft by killing and starting daemons:

  - ✧ Call *HEARTBEAT()* at least every 120 seconds (but not more often than every 30 seconds).
  - ✧ Once reconnection is being attempted, notify the user and take whatever action is appropriate.

VERY DIFFICULT

- Guessing the license keys that belong in the license file. FLEX*lm*'s standard authentication algorithm takes the user-visible data fields (number of licenses, expiration date, version number, vendor-defined string, feature name, host IDs of all servers, plus any optional authenticated fields) and combines them with the vendor's private encryption seeds to produce a license key. The algorithm used is a proprietary one-way block chaining encypherment of all the input data.
- Writing a new daemon that emulates your vendor daemon. FLEX*lm* encrypts the traffic between client and vendor daemon to make this point of attack much more difficult.
- Running the debugger on a stripped (Unix) or a non-debug (Windows) executable. This requires someone to find the FLEX*lm* calls without any symbol table knowledge.

Chapter 13

End-User License Administration

13.1 End-User Options File

End-users can customize software usage via the daemon options file provided by FLEX*lm*. This options file allows the end-user to reserve licenses for specified users or groups of users, to allow or disallow software usage to certain people, to set software timeouts. The daemon options file can be specified in the license file on the VENDOR line as the last parameter as follows:

```
VENDOR daemon-name [optional-path] [options-file]
```

In addition (*new* in version 6), the options file does not need to be specified if:

1. the options file is named *vendor*.opt, and
2. The options file is located in the same directory as the vendor-daemon binary.

A daemon options file consists of lines in the following format:

```
INCLUDE feature[:qualifier] \
        {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
INCLUDEALL {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
EXCLUDE feature[:qualifier] \
        {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
EXCLUDEALL {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
GROUP name user1 user2 ...
LINGER feature[:qualifier] seconds
MAX #lic feature[:qualifier] \
        {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
MAX_OVERDRAFT feature maximum
NOLOG {IN|OUT|DENIED|QUEUED}
REPORTLOG file
RESERVE #lic feature[:qualifier] \
        {USER|HOST|DISPLAY|GROUP|HOSTGROUP|INTERNET} name
TIMEOUT feature[:qualifier] timeout_in_seconds
TIMEOUTALL timeout_in_seconds
```

Lines beginning with a pound sign character (#) are ignored and can be used as comments. If the filename in the REPORTLOG line starts with a “+” character, the old report log file will be opened for append.

Any reference to a feature name can, optionally, specify a specific line that refers to a specific FEATURE or INCREMENT line (as of v5). The syntax is:

```
feature:attribute=value
```

For example:

```
INCLUDE f1:VERSION=2.0 USER daniel
```

If the license file contains two INCREMENT lines, one for v2 and one for v3, this INCLUDE line will only pertain to the v2 licenses. The attribute qualifier can any one of the following:

```
VERSION
HOSTID
EXPDATE
KEY
VENDOR_STRING
ISSUER
NOTICE
dist_info
user_info
asset_info
```

Feature names can also be PACKAGE names, and the operation applies to all the package components.

Lines can be up to 2000 characters. With v7+ clients and servers, newlines are ignored by licenses, so long as comment lines are prefixed with ‘#’ and FEATURE or INCREMENT appear at the beginning of a line. This is because mailers often insert newlines (though they don’t delete them). Prior to v7, ‘\’ line continuation character is required where newlines are added. Line continuation characters are still added by the license generators (lmcrypt).

Note: Versions prior to 4.0 did not support the backslash ‘\’ continuation character. Versions prior to FLEX*lm* v3.0 had a line limit of 200 characters.

EXCLUDE	Allows the end-user to deny certain users a particular feature. EXCLUDE overrides INCLUDE.
EXCLUDEALL	Allows the end-user to deny certain users all features.

GROUP	Defines a group of users for use in the other commands. Multiple GROUP lines for the same group name will have the effect of concatenating all members specified on all the GROUP lines. (Prior to FLEX*lm* v3.0, only the last GROUP line for a given name was effective.)
HOSTGROUP	Allows the specification of a group of hosts for use in the other commands. Multiple HOSTGROUP lines for the same group name will have the effect of concatenating all members specified on all the HOSTGROUP lines.
INCLUDE	Allows the end-user to specify a list of users who are allowed access to a particular feature. EXCLUDE overrides INCLUDE.
INCLUDEALL	Allows the end-user to specify a list of users who are allowed access to all features your daemon supports.
LINGER	Causes licenses to be held by the vendor daemon for a period after the application checks them in or exits.
MAX	Indicates maximum usage for a particular GROUP. You can also specify USER, HOST, HOST_GROUP, etc. as with all end-user options. If a checkout exceeds the maximum, the following error message will be displayed: `Checkout exceeds MAX specified in options file (-87,147)`
MAX_OVERDRAFT	Limits an OVERDRAFT to the specified amount.
NOLOG	causes messages of the specified type to be filtered out of the debug log output; useful to save disk space.
REPORTLOG	Specifies that a logfile be written suitable for use by the FLEX*admin* report writer. Users with the LM_PROJECT environment variable set will have this value logged in this file with each transaction, so reporting can be done by user-project.
RESERVE	Ensures that your application software will always be available to one or more users or on one or more host computer systems.
TIMEOUT	Allows idle licenses to return to the free pool, for use by someone else. This only works if supported by the application. TIMEOUT requires that the application not send heartbeats when it is idle. If the application uses FLEX*lm* timers (`LM_MANUAL_HEARTBEAT` not set), a TIMEOUT specification will be ineffective. If FLEX*lm* timers are disabled, then the application must ensure that it calls *HEARTBEAT()* regularly when active, and does not

	call it when inactive. If *HEARTBEAT()* is not called regularly when active, the TIMEOUT option can cause the client to lose its license.
TIMEOUTALL	Like TIMEOUT, but applies to all features.

All the INCLUDE and EXCLUDE family of options take an internet address in addition to USER, HOST, DISPLAY, and GROUP. The keyword is INTERNET and the address is specified as follows:

```
        a.b.c.d
```

where any of a, b, c, d can be “*”. For example:

```
        INCLUDEALL INTERNET 144.*.*.*
```

This allows any user from network number 144 to access any feature supported by this daemon.

The following options file would reserve a copy of feature compile for user pat, three copies for user lee, and a copy for anyone on a computer with the hostname of terry, and would cause QUEUED messages to be omitted from the log file. In addition, user joe would not be allowed to use the compile feature:

```
        RESERVE 1 compile USER pat
        RESERVE 3 compile USER lee
        RESERVE 1 compile HOST terry
        EXCLUDE compile USER joe
        NOLOG QUEUED
```

If this data were in file `/usr/local/flexlm/options/local.options`, then you would modify the license file VENDOR line as follows:

```
VENDOR XXX options=/usr/local/flexlm/options/local.options
```

SEE ALSO

- Section 8.2, “Configuring Your Vendor Daemon,” on page 50

13.2 License Administration Tools—lmutil (UNIX)

All license administration tools are contained in the single executable lmutil. lmutil contains the following utility programs:

- lmcksum
- lmdiag
- lmdown
- lmhostid
- lminstall
- lmremove
- lmreread

- lmswitchr
- lmstat

lmutil behavior is determined by its first argument, or its argv[0] name. lmutil renamed to lmstat will behave the same as `"lmutil lmstat"`. The installation creates hard links from lmutil to all program names listed when you install your FLEX*lm* kit. You should also create the hard links when your software is installed on your customer's system.

All utilities take the following arguments:

-v	print version and exit
-c *license_file*	operate on "license file"

13.3 License Administration Tools — LMUTIL (Windows, NT)

LMUTIL.EXE command line program similar to the UNIX version previously described is provided. It is accessed by LMUTIL.EXE *Function*, where *Function* is lmstat, lmdiag, etc. A Windows Graphics Interface program called LMTOOLS.EXE is also provided. LMTOOLS has the same functionality as LMUTIL.EXE but is graphically-oriented. Simply run the program and choose a button for the functionality required. Refer to the following sections for information about the options of each feature.

Note: The lmdown, lmremove, and lmreread commands are "privileged". If you have started lmgrd with the "-p -2" switch, you must be a "license administrator" to run any of these three utilities. A "license administrator" is a member of the UNIX "lmadmin" group, or, if the lmadmin group does not exist, a member of group 0. In addition, `lmgrd -x` can disable lmdown and/or lmremove.

13.3.1 lmcksum

lmcksum performs a checksum of a license file. This is useful to verify data entry errors at your customer's location. lmcksum will print a line-by-line checksum for the file as well as an overall file checksum. If the license file contains "`cksum=nn`" attributes, the bad lines will be automatically indicated.

lmcksum will ignore all fields that do not enter into the license key computation; thus, the server node name and port number, as well as the daemon pathname and options file names are not checksummed. In addition, lmcksum will treat non-case-sensitive fields correctly (in general, lmcksum is not case-sensitive). lmcksum takes the "-k" switch to force the license key checksum to be case-sensitive.

lmcksum takes an optional daemon name; if specified, only license file lines for the selected daemon are used to compute the checksums.

By default, lmcksum operates on `license.dat` in the current directory. Specify *-c license_file* if you want to checksum another license file. Example output is:

```
lmcksum - Copyright (C) 1989, 1994 GLOBEtrotter Software, Inc.
lmcksum: using license file "/usr/local/flexlm/licenses/license.dat"
189: SERVER speedy 08002b32b161 2837
166: VENDOR demo /u/gsi/lmgr/src/testsuite/demo
8: FEATURE f1 demo 1.000 01-jan-99 0 3B2BC33CE4E1 "" 08002b32b161
109: (overall file checksum)
```

13.3.2 lmdiag

lmdiag allows you to diagnose problems when you cannot check out a license.

```
lmdiag [-c license_list] [-n] [feature[:specification=value]]
```

where:	**is the:**
-c *license_list*	path to the file(s) to diagnose. If more than one file, use colon separator of Unix, or semi-colon on PC.
-n	run in non-interactive mode; lmdiag will not prompt for any input in this mode. In this mode, extended connection diagnostics are not available.
feature	diagnose this feature only.
specification=value	If a license file contains multiple lines for a particular feature, you can select a particular line for lmdiag to report on. For example: `lmdiag f1:HOSTID=12345678` will attempt a checkout on the line with the hostid limited to 12345678. *specification* can be one of the following: VERSION, HOSTID, EXPDATE, KEY, VENDOR_STRING, ISSUER.

If no *feature* is specified, lmdiag will operate on all features in the license file(s) in your path. lmdiag will first print information about the license, then attempt to check out each license. If the checkout succeeds, lmdiag will indicate this. If the checkout fails, lmdiag explains the reason for the failure. If the checkout fails because lmdiag cannot connect to the license server, then you have the option of running “extended connection diagnostics”.

Extended connection diagnostics attempt to connect to each port on the license server node and can detect if the port number in the license file is incorrect. lmdiag will indicate each port number that is listening, and if it is an lmgrd process, lmdiag will indicate this as well. If lmdiag finds the vendor daemon for the feature being tested, then it will indicate the correct port number for the license file to correct the problem.

13.3.3 lmdown

lmdown allows a graceful shutdown of all license daemons (both lmgrd and all vendor daemons) on all nodes. The syntax is:

```
        % lmdown [-c license-list] [-vendor name] [-q] [-all]
```

where:	**is the:**
license-list	path to the file(s) to shutdown
name	If -vendor *name* is used, only this vendor daemon will be shutdown, and lmgrd will not be shutdown.
-q	Does not prompt "Are you sure?"
-all	Shuts down all license servers in the license-list without prompting.

If you wish to restrict the use of lmdown to license administrators, start lmgrd with the "`-2 -p`" switch. It is reasonable to restrict the execution of lmdown, since shutting down the servers will cause loss of licenses. To disable lmdown, the license administrator can use *lmgrd -x lmdown*.

To stop and restart a single vendor daemon, use *lmdown -vendor name* (and then you can, for example edit the options file), then *lmreread -vendor name*, which restarts a vendor daemon if it's not already running.

SEE ALSO

- Section 8.1,"lmgrd," on page 50
- Section 13.3.7,"lmreread," on page 85

13.3.4 lmhostid

lmhostid is used to print the correct hostid value on any machine supported by FLEX*lm*. The syntax is:

```
        lmhostid [type]
```

where:	**is the:**
type	the type of hostid to print. *Type* must be one of `-flexid` (dongle) `-vsn` (volume serial number) `-ether` (ethernet address) `-long` (32-bit integer) `-internet` (internet address in #.#.#.# format) -cpu (32-bit Pentium III CPU-ID) -cpu64 and -cpu96 also available. The 64 and 96-bit versions are potentially more unique than the 32-bit.

These arguments are only useful on Windows, NT, SCO and HP700. On the SCO and HP700, they exist for backwards compatibility with older hostid types. On Windows and NT, these are the 4 hostid types provided by FLEX*lm* — the default is *ether.*

The output from *lmhostid* will be similar to the following:

```
lmhostid - Copyright (C) 1989, GLOBEtrotter Software, Inc.
The FLEXlm host ID of this machine is "1200abcd"
```

SEE ALSO

- FLEX*lm* Reference Manual section on machine-specific hostids.

13.3.5 lminstall

New in version 6, lminstall is designed primarily for typing in decimal format licenses to generate a readable format license file

```
lminstall [-i {infile | -}] [-o outfile] [-maxlen n]\
        [-overfmt {2, 3, 4, 5, 5.1, or 6}] [-odecimal]
```

Normally, users will simply type lminstall. The user is first prompted for the name of the output license file. The default name is today’s date in yyyyddmm.lic format. The file should be moved to your application’s default license file directory, if specified. Otherwise, the user can use xx_LICENSE_FILE to specify the directory where the *.lic files are located.

Decimal format input is verified by checksum of each line.

To finish entering, type Q on a line by itself, or enter 2 blank lines.

If “infile” is a dash (‘-’), it takes input from stdin.When ‘-i’ is used, default output is stdout; otherwise if -o is not specified, lminstall prompts the user for an output file name.

LMINSTALL AS A CONVERSION TOOL:

lminstall can alternatively be used to convert licenses between decimal and readable format, and between different versions of FLEX*lm* license formats.

To convert from readable to decimal:

```
% lminstall -i infile -o outfile -odecimal
```

To convert to FLEX*lm* Version 2 format:

```
% lminstall -i infile -o outfile -verfmt 2
```

Conversion errors are reported as necessary. lminstall has a limit of 1000 lines of input.

To enforce a maximum line length of 50 (to discourage mailers from inserting newlines):

```
% lminstall -maxlen 50
```

13.3.6 lmremove

lmremove allows the system administrator to remove a single user's license for a specified feature. This is sometimes required in the case where the licensed user was running the software on a node that subsequently crashed — in these cases, due to a limitation in the way TCP works, it can take several hours for the license server to detect that the user is gone. This situation will sometimes cause the license to remain unusable.

Note: If the application is active when it is removed with *lmremove*, it will simply checkout the license again (assuming the applications FLEX*lm* timers are enabled or *HEARTBEAT()* is called). lmremove therefore cannot be used to "steal" licenses.

---

lmremove will allow the license to return to the pool of available licenses. The syntax is:

```
lmremove [ -c file_list ] feature user host display
```

The *user host display* information must be obtained from the output of `lmstat -a`.

lmremove removes all instances of *user* on *host* on *display* from usage of *feature*. If the optional -c *file* is specified, the indicated file is used as the license file. The end-user system administrator should protect the execution of *lmremove* since removing a user's license can be disruptive.

An alternate usage, which makes use of the license handle in the vendor daemon, is:

```
lmremove [-c file_list] -h feature serverhost port handle
```

This variation uses the serverhost, port, and license handle, as reported by `lmstat -a`. Consider this example `lmstat -a` output:

```
joe cloud7 /dev/ttyp5 (v1.000) (cloud9/7654 102), start Fri 10/29 18:40
```

In this example, the serverhost is "cloud9", the port is "7654", and the license handle is 102. To remove this license, issue the following command:

```
lmremove -h f1 cloud9 7654 102
```

or

```
lmremove f1 joe cloud7 /dev/ttyp5
```

When removing by handle, if licenses are grouped as duplicates, all duplicate licenses will also be removed.

SEE ALSO

- Section 13.3.8,"lmstat," on page 85

13.3.7 lmreread

lmreread causes the license daemon to reread the license file and start any new vendor daemons that have been added. In addition, all currently running daemons will be signaled to re-read the license file for changes in feature licensing information. If the optional daemon name is specified, only the named daemon will re-read the license file (in this case, lmgrd does not re-read the license file either).

The syntax is:

```
        lmreread [-vendor name] [ -c license-file-list ] [-all]
```

where:	**is the:**
license-list	path to the file(s) to find the lmgrd server.
name	If -vendor *name* is used, only this vendor daemon will reread the license file. If the vendor daemon is not running, lmgrd will start it.
-all	rereads all servers in the *license-file-list*

To disable lmreread, the license administrator can use *lmgrd -x lmreread*.

To stop and restart a single vendor daemon, use *lmdown -vendor name* (and then you can, for example edit the options file), then *lmreread -vendor name*, which restarts a vendor daemon if it's not already running.

Note: If the -c option is used, the license file specified will be read by lmreread, not by lmgrd*;* lmgrd re-reads the file it read originally. Also, lmreread cannot be used to change server node names or port numbers. Vendor daemons will not re-read their option files as a result of lmreread.

SEE ALSO

- Section 8.1,"lmgrd," on page 50
- Section 13.3.3,"lmdown," on page 82

13.3.8 lmstat

License administration is simplified by the lmstat utility. lmstat allows the user of FLEX*lm* to instantly monitor the status of all network licensing activities. lmstat allows a license administrator to monitor license management operations including:

- Daemons that are running
- Users of individual features
- Users of features served by a specific VENDOR

Except for the *-i* option, *lmstat* prints information that it receives from the license server. Therefore, if no server is running because all features are *uncounted*, *lmstat* will not show anything. Prior to v5, lmstat only showed information for *counted* features.

The syntax is:

```
        lmstat  [-a] [-S daemon] [-f feature] [-i feature]
                [-s [server]] [-t value] [-c license_file_list] [ -A ]
```

-a	Displays everything
-A	Lists all active licenses only
-c *license_file_list*	Uses *license_file(s)*
-S [*daemon*]	Restricts output to one *daemon*, and the features and users of that *daemon*
-f [*feature_name*]	Lists users of *feature*(s)
-i [*feature_name*]	Prints information about the named *feature*, or all features if no feature name is given. Note that lmstat -i does *not* communicate with the server, and therefore reports raw data from the license file, which may differ from what the server actually supports.
-s [*server_name*]	Displays status of *server* node(s)
-t *value*	Sets lmstat timeout to *value*

Notes:
- lmstat -a is a potentially expensive command. With many active users, this can generate a lot of network activity, and therefore should not be used too often.
- lmremove requires the output of "`lmstat -a`."

13.3.9 lmswitchr

lmswitchr switches the FLEX*admin* (REPORTLOG) log file for the specified feature.

The syntax is:

```
        lmswitchr [-c license_file_list] { feature | daemon } new-file
```

SEE ALSO

- Section 13.1, "End-User Options File," on page 76 for REPORTLOG

13.3.10lmver

lmver reports the FLEX*lm* version of a library or binary. The syntax is:

```
        lmver [filename]
```

If a filename is specified, the FLEX*lm* version incorporated into this file is displayed; otherwise lmver looks for the library file *liblmgr.a* to detect its version.

13.4 Switching the Debug Log File Under UNIX

The FLEX*lm* daemons create an ascii Debug log file on stdout. There are several processes in a parent-child hierarchy which are sharing the same file pointer, so this log file cannot be changed after the vendor daemons have been started, since each process has a copy of the current offset, etc.

There is another way to switch the log file output data however; this involves piping the stdout of lmgrd to a shell script that appends each line to a file. This is done as follows:

Instead of the “normal” startup:

```
% lmgrd > LOG
```

Start lmgrd this way:

```
% lmgrd -z | sh -c ’while read line; do echo "$line" >> LOG ; done’
```

With this startup method, the output file “LOG” can be renamed and a new log file will be created. You could even make “LOG” a symbolic link and change the value of the link to “switch” the log file.

Chapter 14

End-User Installation Instruction Template

To ensure that your customers can use your FLEX*lm*-managed product easily and successfully, use the information in this chapter as a guideline about what they need and what information they need to know.

14.1 Binaries Your Customers Will Require

When your application software is built with the calls to FLEX*lm*, you will need to ship the following four (or five) files in addition to the files that you normally ship in your installation kit:

vendor.lic	The license file, customized for your customer.
lmgrd (or lmgrd.exe)	The License Manager daemon (license daemon).
lmutil (or lmtools.exe)	FLEX*lm* utility program(s).
lcmflxa.dll	If you’re using the LCM feature.
flckflxa.dll	If you’re using the FLEXlock feature.
xyzd (or xyzd.exe)	Your vendor daemon.

14.2 Information Every Customer Needs to Know

In addition to installing your software, your customer will need to do the following steps.

14.2.1 Install the license file

We recommend that the application specify a default location directory in the installed “product hierarchy,” for example “*installed-path*/licenses”, where *installed-path* is determined by the end-user during installation, and licenses is a directory. At run time *installed-path* is determined (many applications do this with an environment variable or global internal character string), and the default location directory is specified in *CHECKOUT()*, *lp_checkout()* or *lc_set_attr(key,* LM_A_LICENSE_DEFAULT,...*)*.

The license file be installed in that directory with a “.lic” suffix. A useful system is to name the file the date of installation, in *yyyymmdd* format. For example, if the license is generated 10 January, 1998, it can be installed as

```
        installed-path/licenses/19980110.lic
```

Since your application has specified a directory, it does not need to know the exact name of the file, only that it ends with ".lic", and lies in the directory.

We recommend the license file be delivered by email. If delivered by email, the entire email message can be saved in the specified location; email headers and extraneous text are automatically ignored by FLEX*lm*.

If delivered in decimal format, the end-user should use lminstall. lminstall will request the license location, and will default to a license file called "*yyyymmdd*.lic". This file should then be moved to the *installed-path*/licenses directory.

LCM ON WINDOWS

LCM ("License Certificate Manager") automatically handles internet-based license fulfillment. This works in combination with GLOBE*trotter* Software's Globetrack Web-Licensing.

14.2.2 If licenses are *counted*, install lmgrd and vendor-daemon

These can be installed wherever the end-user prefers. lmgrd will need to find the vendor-daemon in the same directory, or in it's *$PATH*. Otherwise, the user will need to edit the license file to add the path to the vendor-daemon to the VENDOR line.

14.2.3 If licenses are *counted*, start lmgrd

First, make sure the vendor-daemon is in the same directory, or lmgrd's $PATH, or it's in the same directory as lmgrd, or the user has edited the license file to include the path to the vendor daemon on the VENDOR line.

Start the license daemon as follows:

```
% lmgrd -c license_file_path(Unix)
C> lmgrd -app -c license_file_list (NT and Windows 95)
```

where *license_file_list* is the full pathname to the license file or a delimited license-file list.

14.2.4 If licenses are *counted*, install lmgrd to start at boot.

On Unix, edit the appropriate boot script, which may be `/etc/rc.boot`, `/etc/rc.local, /etc/rc2.d/Sxxx, /sbin/rc2.d/Sxxxx, etc.` Remember that these scripts are run in /bin/sh, so do not use the *csh* ">&" redirection syntax.

Each Unix operating system can have some quirks in doing this, but the following script has been successfully tested for HP700 systems. See the notes following for a full explanation.

```
/bin/su daniel -c ‘echo "starting lmgrd" > \
        /home/flexlm/v5.12/hp700_u9/boot.log’

/bin/nohup /bin/su daniel -c "umask 022; \
        /home/flexlm/v5.12/hp700_u9/lmgrd -c \
                /home/flexlm/v5.12/hp700_u9/license.dat >>& \
                /home/flexlm/v5.12/hp700_u9/boot.log"

        /bin/su daniel -c ‘echo "sleep 5" >> \
                /home/flexlm/v5.12/hp700_u9/boot.log’
        /bin/sleep 5


        /bin/su daniel -c ‘echo "lmdiag" >>\
                /home/flexlm/v5.12/hp700_u9/boot.log’

        /bin/su daniel -c ‘/home/flexlm/v5.12/hp700_u9/lmdiag -n -c\
                /home/flexlm/v5.12/hp700_u9/license.dat >> \
                /home/flexlm/v5.12/hp700_u9/boot.log’
        /bin/su daniel -c ‘echo "exiting" >>\
                /home/flexlm/v5.12/hp700_u9/boot.log’
```

Please note the following about how this script was written:

- All paths are specified in full, since no paths can be assumed at boot time.
- Since no paths are assumed, the vendor daemon must be in the same directory as lmgrd, or the VENDOR lines must be edited to include the full path to the vendor-daemon binary file.
- The “su” command is used to run lmgrd as a non-root user, “daniel”. We recommend that lmgrd not be run as root, since it can be a security risk to run any program that does not require root permissions, and lmgrd does not require root permissions.
- Daniel has a csh login, so all commands executed as daniel must be in csh syntax. All commands not executed as daniel must be in /bin/sh syntax, since that’s what’s used by the boot scripts.
- The use of “nohup” and “sleep” are required on some operating systems, notably HPUX and Digital Unix, for obscure technical reasons. These are not needed on Solaris and some other operating systems, but are safe to use on all.
- lmdiag is used as a diagnostic tool to verify that the server is running and serving licenses.

Note: On IBM RS6000 systems, /etc/rc cannot be used, because TCP/IP is not installed when this script is run. Instead, /etc/inittab must be used. Add a line like this to /etc/inittab after the lines which start networking:

```
        rclocal:2:wait:/etc/rc.local > /dev/console 2>&1
```
