End Users Guide

VERSION 8.1

FEBRUARY 2002

GLOBEtrotter®
a macrovision company

Printed in the USA.
February 2002

Table of

Contents

Preface

Welcome to FLEX*lm*®, the *de facto* standard network license manager used by over 2000 leading software vendors to control the use of their software products. If you are a system administrator or user, chances are one or more of the products currently on your network is licensed by FLEX*lm*.

About This Manual

This manual explains FLEX*lm* for administrators and end users and describes how to use the tools which are part of the standard FLEX*lm* distribution kit. GLOBE*trotter* Software also provides the SAM*suite*™ and SAM*report*™ asset management tools for more advanced license server control and reporting. SAM*wrap* is another GLOBE*trotter* product that allows end users to provide FLEX*lm* license management for applications without embedded FLEX*lm* license management. Please contact GLOBE*trotter* Software by email at info@globes.com or on the Internet at http://www.globetrotter.com for more information about SAM*suite*, SAM*report*, or SAM*wrap*.

Keep in mind that certain topics (such as password encryption) are vendor-specific and proprietary so they cannot be documented in any detail. Also, because FLEX*lm* does not enforce a particular licensing strategy, each vendor's implementation can have subtle differences. If you do not find out what you need to know here, you should contact your vendor's technical support group.

Versions of FLEX*lm*

This manual covers features of interest to license administrators and end users in FLEX*lm* versions 1.0 through 8.1.

Related Documents from GLOBE*trotter* Software

The *SAMsuite Users Guide* describes the SAM*suite* license administration tool for managing FLEX*lm*-enabled applications, the *SAMreport Users Guide* describes the SAM*report* license usage reporting tool, and the *SAMwrap Users Guide* describes SAM*wrap* for administrators who wish to provide FLEX*lm* license management for applications shipped without embedded FLEX*lm* license management.

The *FLEXlm Programmers Guide* and *FLEXlm Reference Manual* are for programmers responsible for incorporating FLEX*lm* into their products.

Typographic Conventions

The following typographic conventions are used in this manual:

- The first time a new term is used it is presented in *italics.*
- Commands and path, file, and environment variable names are presented in a `fixed_font`.
- Other variable names are in an *`italic_fixed_font`*.
- API function calls are in a sans-serif font.

Contacting Technical Support

GLOBE*trotter* Technical Support is available to customers and prospects with current support contracts. Please include the following information with your inquiry:

- Product Name
- Product Version
- Operating System Name and Version

Support Centers

Contact the support center in your area:

- North America
  - Telephone: (408) 445-8100, 8:30am-5:00pm PST
  - Email: support@globes.com, 24 hours
  - Fax: (408) 445-7760, 24 hours
  - Online: http://www.globes.com
  - GTwebSupport

- Europe
  - Telephone: +44 (1928) 579 789, 8:30am-5:00pm GMT
  - Email: support@globes-europe.com, 24 hours
  - Fax: +44 (1928) 579 799, 24 hours
  - Online: http://www.globes.com/europe
- Japan
  - Telephone: +81-35774-6253, 9:30am-6:00pm JST
  - Email: support-japan@globes.com, 24 hours
  - Fax: +81-35774-6269, 24 hours

All other customers and prospects please contact the North American office.

Chapter 1

Introduction and Overview

This chapter explains the basics of floating (network) licensing and gives a quick overview of the components of FLEX*lm*. It explains where license administrators have control and where end users have control. Section 1.6, "Getting Started Checklist," tells both license administrators and end users how to start managing FLEX*lm*.

1.1 Introduction to FLEX*lm*

FLEX*lm* is the most popular license manager used in the software industry. FLEX*lm* is best known for its ability to allow software licenses to be available (or float) anywhere on a network, instead of being tied to specific machines. Floating licensing benefits both users and license administrators. Users can make more efficient use of fewer licenses by sharing them on the network. License administrators can control who uses the licensed application and the node(s) where the licenses will be available. See Section 2.4, "Types of License Files," for details about the different licensing models supported by FLEX*lm*.

1.2 How to Use This Manual

This manual is written for two different audiences:

- Administrators of FLEX*lm* licenses and license servers
- End users of FLEX*lm*-enabled applications

1.2.1 License Administrator

If you are a license administrator, read these chapters:

Chapter:	**Explains:**
Preface	Overview of this manual.

Chapter:	Explains:
Chapter 1, ”Introduction and Overview”	FLEX*lm* basics: license manager and vendor daemons; the license file; configuring FLEX*lm*; the license request process.
Chapter 2, ”The License File”	The license file format; setting the path at start-up; different types of licensing policies.
Chapter 3, ”Multiple License Files”	Using license files from independent software vendors.
Chapter 4, ”Selecting Server Nodes”	Selecting which machines will run the license servers; resources required by the servers; multiple servers; quorums; redundant license servers.
Chapter 5, ”The Options File”	Creating and editing the options file.
Chapter 6, ”License Administration Tools”	Managing FLEX*lm* using GLOBE*trotter*-supplied utilities.
Chapter 7, ”Mobile Licensing”	Licensing to allow working on a computer disconnected from the license server.

In addition, you can refer to Appendix B, “Troubleshooting Guide,” which contains a list of common problems and their solutions and Appendix D, “Frequently Asked Questions.”

1.2.2 End User

If you are an end user, read these chapters:

Chapter:	Explains:
Preface	Overview of this manual.
Chapter 1, ”Introduction and Overview”	FLEX*lm* basics: license and vendor daemons; the license file; configuring FLEX*lm*; the license request process.

Chapter:	Explains:
Chapter 2, ”The License File”	The license file format; setting the path at start-up; different types of licensing policies.
Chapter 7, ”Mobile Licensing”	Licensing to allow working on a computer disconnected from the license server.

In addition, you can refer to Appendix B, “Troubleshooting Guide,” which contains a list of common problems and their solutions.

1.3 FLEX*lm* Components

The four main components of FLEX*lm* are:

- License manager daemon, `lmgrd`
- Vendor daemon
- Client application program
- License file

1.3.1 The License Manager Daemon (lmgrd)

The *license manager daemon* (`lmgrd`) handles the initial contact with the client application programs, passing the connection on to the appropriate vendor daemon. It also starts and restarts the vendor daemons.

1.3.2 The Vendor Daemon

In FLEX*lm*, counting (floating) licenses are granted by running processes. There is one process for each vendor who has a FLEX*lm*-licensed product on the network. This process is called the *vendor daemon*. The vendor daemon keeps track of how many licenses are checked out, and who has them. If the vendor daemon terminates for any reason, all users lose their licenses (though this does not mean the applications suddenly stop running). Users normally regain their license automatically when `lmgrd` restarts the vendor daemon, though they may exit if the vendor daemon remains unavailable.

Client programs communicate with the vendor daemon, usually through TCP/IP network communications. The client application and the daemon processes (the *license server*) can run on separate nodes on your network, across any size wide-area network. Also, the format of the traffic between the client and the vendor daemon is machine independent, allowing for

heterogeneous networks. This means the license server and the computer running an application can be either different hardware platforms or even different operating systems (Windows and UNIX, for example).

1.3.3 The License File

Licensing data is stored in a text file called the *license file*. The license file is created by the software vendor, and edited and installed by the license administrator. It contains information about the server nodes and vendor daemons, and at least one line of data (called FEATURE or INCREMENT lines) for each licensed product. Each FEATURE line contains a license key or signature based on the data in that line, the hostids specified in the SERVER line(s), and other vendor-specific data. Client applications that are licensed with a node-locked, uncounted license need only read a valid license file to run—they do not need a license server.

In some environments, the licensing information for several vendors may be combined into a single license file.

Most applications have an expected location for the license file, documented by that application. End users can usually override this location by setting the environment variable `LM_LICENSE_FILE` to point elsewhere, or by following instructions supplied with the licensed application. If your site has software from multiple vendors with incompatible license files (due to different sets of servers), you can keep the data in separate files and set the `LM_LICENSE_FILE` variable to reference multiple files.

It's strongly recommended that you keep a link (on UNIX) or copy of the license file in the vendor's expected license location, so that users will not need to set `LM_LICENSE_FILE` to run their applications. If the licenses are counted (floating) this license should have a USE_SERVER line directly after the SERVER line. For details, see Chapter 2, "The License File." See also Appendix C, "FLEXlm Environment Variables."

1.3.4 The Application Program

The application program using FLEX*lm* is linked with the program module (called the FLEX*lm* client library) that provides the communication with the license server. On Windows, if dynamically linked, this module is called `lmgr`*`xxx`*`.dll`, where *xxx* indicates the FLEX*lm* version. During execution, the application program communicates with the vendor daemon to request a license.

1.4 The License Request Process

When you run a counted FLEX*lm*-licensed application, the following occurs:

1. The license module in the client application finds the license file, which includes the host name of the license server node and port number of the license manager daemon, `lmgrd`.
2. The client establishes a connection with the license manager daemon (`lmgrd`) and tells it what vendor daemon it needs to talk to.
3. `lmgrd` determines which machine and port correspond to the master vendor daemon and sends that information back to the client.
4. The client establishes a connection with the specified vendor daemon and sends its request for a license.
5. The vendor daemon checks in its memory to see if any licenses are available and sends a grant or denial back to the client.
6. The license module in the application grants or denies use of the feature, as appropriate.

Uncounted features (where the license count is 0) do not require a license server, and the FLEX*lm* client library routines in the application grant or deny usage based solely upon the contents of the license file.

1.5 Configuring FLEX*lm*

Most of the parameters of FLEX*lm* are configurable by the license administrator. The license administrator can set the:

- Location of the license file (though it's recommended that a copy or link of the license remains at the location where the application expects it)
- Location of all executables
- Location of all log files
- TCP/IP port number used by the license manager process, `lmgrd`

In addition, the license administrator can reserve licenses for specific users, nodes, or groups, and control other license-related options. Changing parameters is discussed in Chapter 5, "The Options File."

Note: Refer to your vendor's documentation before attempting to change file names, locations, or contents.

1.6 Getting Started Checklist

The following sections provide a quick overview of how to set up and use licensing for FLEX*lm*-licensed products. By scanning the list, you should be able to quickly find the areas of interest.

1.6.1 Installing Licensed Software

As a license administrator you are responsible for setting up licensing on your system or network. This section tells you how to do that. If you are an end user of the application and you will not be involved in installing it, then go to Section 1.6.2, “Notes for End Users.”

Remember that the installation guide for your application software is the final word on installing and configuring FLEX*lm*. Generally, however, installing FLEX*lm* licensing requires the following steps:

1. Select your license server nodes and get their hostids. See Appendix A, “Hostids for FLEXlm-Supported Machines.”
2. Give the hostids to your software vendor and get a license file (or the data to enter in the license file) in return.
3. Consider combining the new license file with any existing license files. See Chapter 3, “Multiple License Files.”
4. Determine if an options file is desired, and if so, set it up.
5. Determine where to install the FLEX*lm* utility programs such as `lmgrd` and `lmutil` (`lmstat`/`lmdown`/etc.) and install them, unless your vendor’s installation script does so for you.
6. Start `lmgrd` (the license manager daemon) manually; you may also want to set it up to start automatically at boot time. See Section 2.1.1, “Starting the License Server at System Startup.”

These steps are discussed briefly below.

LICENSE SERVER NODE AND HOSTIDS

Before running any FLEX*lm*-licensed program using floating licenses, you will need to set up your license server node (or nodes). You must select which node or nodes to run your license servers on and provide the hostids of those machines to your software vendor. For pointers on selecting your server machine, see Chapter 4, “Selecting Server Nodes.”

You can get the hostid of the server machine by running FLEX*lm*'s `lmhostid` utility on that machine. If you don't have `lmhostid`, you can get the hostid of your machine by using the appropriate command as described in Appendix A, "Hostids for FLEXlm-Supported Machines."

Using the hostid of your server machines, your vendor will send you a license file that enables the application software.

LICENSE FILES AND LMGRD

Once you have received a license file from your vendor, you must install it on your system and start up the license manager daemon, `lmgrd`.

- Your software vendor may have selected a default location for your license file. If not, you can use any location you wish. For more details, see Chapter 2, "The License File."
- To start `lmgrd` automatically at boot time, you will have to modify your system files (UNIX) or use LMTOOLS (Windows). For details, see Section 2.1.1, "Starting the License Server at System Startup."

ADMINISTRATION TOOLS

GLOBE*trotter* Software supplies administration tools to your software vendor. The vendor usually includes these utilities with their product. You can download the latest versions from http://www.globetrotter.com.lmgrd.htm. See Chapter 6, "License Administration Tools," for more information about how to use the FLEX*lm* utilities.

OPTIONS FILES

The options file controls various options such as reservations and timeouts of licenses. Most users run without an options file, but you may decide you want to use some options. For example, many administrators use an option to limit the quantity and content of logged messages. To set up an options file, see Chapter 5, "The Options File."

1.6.2 Notes for End Users

As a user of a FLEX*lm*-licensed application, you may need to know a few things to use the system effectively. The main things you need to know are:

- How to tell an application which license file to use
- How to query the system to find out who is using a license

How to Specify a License File

The license file determines what features are available to a program. It also contains information telling the application how to connect to the license server.

For information about the standard way of specifying a license file for an application, see Chapter 2, “The License File.”

Getting Information About Licenses

To find out who is using a license run `lmstat`, described in Chapter 6, “License Administration Tools.”

Chapter 2

The License File

The license file contains all site-specific information required by FLEX*lm*. This information includes:

- Server names and hostids
- Vendor names and paths to vendor daemon executables
- Feature information

In general, the license file, or a copy of it, must be accessible to every machine that runs a FLEX*lm*-licensed application, and to each machine designated as a license server. If the license file contains counted (also called “floating”) licenses, before you can use the application you have to start the license manager daemon (`lmgrd`) using the following syntax:

```
lmgrd -c license_file_path -l debug_log_path
```

where *`license_file_path`* is the full path to the license file and *`debug_log_path`* is the full path to the debug log file.

2.1 Specifying Location of the License File

Most software vendors recommend a specific location for your license file. If you are running the application on multiple nodes, you have these options for making your license available on all the machines:

- Place the license file in a partition which is available (via NFS on UNIX systems) to all nodes in the network that need the license file.
- Copy the license file to all of the nodes where it is needed.
- Set `LM_LICENSE_FILE` to *`port`*`@`*`host`*, where *`host`* and *`port`* come from the SERVER line in the license file. With v6.0+, you can use `@`*`host`*, if the license file SERVER line uses a default port or specifies a port in the default port range (27000-27009).

- On Windows (v6.0+), if the application can't find the license file, the user is presented with a dialog that asks for the license file location, the name of the system running the license server, or allows the user to type in the license directly.

Since the vendor daemon keeps track of license usage, and since the license file contains encrypted data to protect it against modification, you may move and copy the license file as much as necessary.

For counted licenses, no matter which option you choose, you must first install `lmgrd` and the vendor daemon.

With a FLEX*lm* v6.0+ vendor daemon and `lmgrd`, the license path can be a list of files, separated by colons on UNIX or semi-colons on Windows. If there is a directory in this list, all files named `*.lic` in that directory are used.

Note: You can only start `lmgrd` on the server node specified in the license file.

Note: If you are running three-server redundant license servers, you should have separate copies of the license file (as well as the binaries for `lmgrd` and the vendor daemons) on *each* server node. If you do not do this, you lose all the advantages of having redundant servers, since the file server holding these files becomes a single point of failure.

2.1.1 Starting the License Server at System Startup

If any licenses in the license file are counted (license count > 0), then the license server must be started before the product can be used.

UNIX

To start the license manager daemon (`lmgrd`), execute a command similar to the following.

If you are running in the C shell:

```
lmgrd_path -c license_file_path -l debug_log_path &
```

If you are using either the Korn or Bourne shell:

```
nohup lmgrd_path -c license_file_path -l debug_log_path 2>&1 &
```

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, because no paths can be assumed at boot time.
- Because no paths are assumed, the vendor daemon must be in the same directory as `lmgrd`, or the DAEMON lines must be edited to include the full path to the vendor daemon.

- The `su` command is used to run `lmgrd` as a non-root user, “daniel.” It is recommended that `lmgrd` not be run as “root,” since it can be a security risk to run any program as “root” 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 is what is used by the boot scripts.
- The use of `nohup` and `sleep` are required on some operating systems, notably HP-UX 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
```

Note: This will not start the daemon until you reboot your license server machine.

WINDOWS

From LMTOOLS, start `lmgrd`. You can optionally indicate that this should be started at system startup.

2.1.2 Setting the Path with an Environment Variable

Most applications specify a location where they expect to find the license file. Many will automatically install the license file. You should rarely, if ever, be required to specify where the license file is located with an environment variable. However, you can change the license file location with `LM_LICENSE_FILE`, or if a location is not set by the application, you can set one.

Use the environment variable `LM_LICENSE_FILE` to set the location of the license file. For example in the C shell:

```
setenv LM_LICENSE_FILE license_file_path
```

In the Korn and Bourne shells:

```
LM_LICENSE_FILE=license_file_path
export LM_LICENSE_FILE
```

On Windows 95, add the following line to `C:\autoexec.bat`:

```
SET LM_LICENSE_FILE=license_file_path
```

On Windows NT, use the System Control Panel to change the global environment, adding *`license_file_path`* to `LM_LICENSE_FILE`, where *`license_file_path`* is the full path to the license file. This can also be a *`port@host`* setting, where *`port`* and *`host`* are the port number and host name from the SERVER line in the license file. v6.0+ applications can use simply *`@host`*, if the server uses a default port number.

In FLEX*lm* v6.0+, applications will accept an environment variable (or Windows Registry) named *`VENDOR`*`_LICENSE_FILE`, where *`VENDOR`* is the vendor daemon name, e.g., `GSI_LICENSE_FILE`.

With `lmgrd` and `lmutil` (`lmstat`, `lmdown`, etc.), the `-c` option overrides the setting of the `LM_LICENSE_FILE` environment variable. See Section 3.1.3, “Using Separate License Files on the Same Server Node,” for more information about `LM_LICENSE_FILE`.

Note: Some applications do not recognize the `LM_LICENSE_FILE` environment variable.

See also Appendix C, “FLEXlm Environment Variables.”

2.2 License File Format

License files usually begin with a SERVER line (or three lines for three-server redundant servers) followed by one or more DAEMON lines, followed by one or more FEATURE or INCREMENT lines. In some cases the license file requires no SERVER line and no DAEMON line. See Section 4.4, “Counted vs. Uncounted Licenses,” for more information. Since FLEX*lm* v6.0, the DAEMON line can be called VENDOR. Wherever DAEMON appears, VENDOR can be used, if the `lmgrd` and vendor daemon are both >= FLEX*lm* v6.0.

You can modify these data items in the license file:

- Node names on the SERVER line(s)
- Port numbers on the SERVER line(s)

- Paths on the DAEMON line(s)
- Options file paths on the DAEMON line(s)
- Optional port numbers on the DAEMON line(s) (for firewall support only)
- USE_SERVER line (FLEX*lm* v5.0+ only)
- Values in *keyword=value* pairs on FEATURE lines, if *keyword* is all lowercase

Long lines normally use the "\" line-continuation character to break up long lines (though this is not required with v7.0+ applications). FLEX*lm* v2 did not support the line-continuation character, although this rarely matters since optional attributes weren't supported then either.

In FLEX*lm* v8.0+, 8-bit Latin-based characters are fully supported in license files, options files, log files, and client environments.

Note: Everything else is used to compute the license key or signature, and should be entered exactly as supplied by your software vendor.

2.2.1 SERVER Lines

The SERVER line specifies the node name and hostid of the license server and the port number of the license manager daemon (`lmgrd`). Normally a license file has one SERVER line. Three SERVER lines mean that you are using a three-server redundant license server. The absence of a SERVER line means that every FEATURE or INCREMENT line in the license file is uncounted. For more information about uncounted features, see Section 2.2.4, "FEATURE or INCREMENT Lines." License administrators do not have the option of deleting SERVER lines from a license file because the hostids from the SERVER lines are computed into the license key or signature on every FEATURE and INCREMENT line. For more information about redundant servers, see Chapter 4, "Selecting Server Nodes."

The format of the SERVER line is:

```
SERVER host hostid [port]
```

where:

host	The system host name or IP address. String returned by the UNIX `hostname` or `uname -n` command. On NT, `ipconfig /all`; on Windows 95, `winipcfg /all` return the host name. If the application uses FLEX*lm* v5 or higher, this can be an IP address (in *###.###.###.###* format).
hostid	Usually the string returned by the `lmhostid` command. This can only be changed by your software supplier.
port	TCP port number to use. A valid number is any unused port number between 0 and 64000. On UNIX, choose a port >1024, since those <1024 are privileged port numbers. The port number is optional if `lmgrd`, the vendor daemon, and the application are v6.0+ (if no port number is specified, one of the default ports in the range of 27000 and 27009 will be used).

Example:

```
SERVER enterprise 0122345 21987
```

2.2.2 DAEMON (or VENDOR) Lines

The DAEMON line specifies the daemon name and path. `lmgrd` uses this line to start the vendor daemon, and the vendor daemon reads it to find its options file. The format of the DAEMON line is shown below.

Note: Since FLEX*lm* v6.0, the DAEMON line can be called VENDOR. VENDOR can be used if the `lmgrd` and vendor daemon are both at least FLEX*lm* v6.0.

```
{DAEMON|VENDOR} vendor [vendor_daemon_path]\
   [[options=]options_file_path] [[port=]port]
```

where:

vendor	Name of the vendor daemon used to serve some feature(s) in the file. This name cannot be changed by the administrator.

vendor_daemon_path	Path to the executable for this daemon. Generally the license administrator is free to install the daemon in any directory. (It is recommended, however, that it be installed in a local directory on the license server node.) If the vendor daemon is v6.0+, this path is optional. If left out, `lmgrd` will look for the vendor daemon binary in the current directory, the path specified in `lmgrd's` $PATH environment variable, or in the directory where `lmgrd` is located. If the vendor daemon path is blank, then the `options=` and `port=` strings are required if options or port number are specified.
options_file_path	Full path to the end-user options file for this daemon. (See Chapter 5, “The Options File.”) FLEX*lm* does not require an options file. The keyword `options=` requires a v5.0+ vendor daemon. If the vendor daemon is v6.0+, the options file need not be specified on this line. If it is called *vendor*`.opt` (where *vendor* is the vendor daemon name) and located in the same directory as the license file, the vendor daemon will automatically find and use it.
port	Vendor daemon port number. This requires a v5.0+ `lmgrd`. **Note:** This is for firewall support only and is otherwise not recommended. If a port number is specified on the DAEMON line, there is a delay restarting the vendor daemon until all the clients have closed their connections to the vendor daemon.

v6.0+:

```
VENDOR sampled
```

pre-v6.0:

```
DAEMON sampled /etc/sampled \
  /a/b/sampled/licenses/sampled.opt
```

2.2.3 USE_SERVER Line (v5.0+ only)

USE_SERVER takes no arguments and has no impact on the server. When the application sees USE_SERVER, it ignores everything in the license file except preceding SERVER lines, and the checkout validation occurs at the vendor daemon. USE_SERVER is recommended since it improves performance when a license server is used. For uncounted features, USE_SERVER can be used to force logging of usage by the daemons.

2.2.4 FEATURE or INCREMENT Lines

A FEATURE line describes the license required to use a product. An INCREMENT line can be used in place of a FEATURE line, as well as to incrementally add licenses to a prior FEATURE or INCREMENT line in the license file.

Only the first FEATURE line for a given feature will be processed by the vendor daemon. If you want to have additional copies of the same feature (for example, to have multiple node-locked, counted features), then you must use multiple INCREMENT lines. INCREMENT lines form license groups based on the feature name, version, and node-lock hostid. If the feature name, version, and node-lock hostid (and optionally, the vendor string, if the vendor specified this) match a prior INCREMENT or FEATURE line, the new number of licenses is added to the old number. If any of the three do not match, a new group of licenses is created in the vendor daemon, and this group (called a *license pool*) is counted independently from others with the same feature name. INCREMENT is not available for pre-v2.61 FLEX*lm* clients or servers. A FEATURE line does not give an additional number of licenses, whereas an INCREMENT line always gives an additional number of licenses.

Note: There is a rarely used option in FLEX*lm* which causes FEATURE lines to function as INCREMENT lines. This option is called ls_use_all_feature_lines. You will have to ask your vendor if they use this option. If they do, then all FEATURE lines behave exactly as INCREMENT lines.

A FEATURE line placed after another FEATURE or INCREMENT line will be ignored (unless ls_use_all_feature_lines is set).

Nothing in a FEATURE/INCREMENT line is editable, except for values in `keyword=value` pairs where `keyword` is all lowercase.

Older formats are understood by new clients and servers, but the new formats are more flexible. A FEATURE/INCREMENT line must have a license key or a SIGN= signature—or it could have both.

v2.0 format:

```
{FEATURE|INCREMENT} feature vendor \
feat_version exp_date num_lic license_key \
"vendor_str" [feat_hostid]
```

v3.0+ format:

```
{FEATURE|INCREMENT} feature vendor feat_version \
exp_date num_lic [license_key] [optional_attributes] \
[SIGN=sign]
```

where:

feature	Name given to the feature by the vendor.
vendor	Name of the vendor daemon; also found in the DAEMON line. The specified daemon serves this feature.
feat_version	Version of this feature that is supported by this license.
exp_date	Expiration date of license, e.g., 7-may-1996. **Note:** If the year is 0 (or 00, 000, 0000) then the license never expires. Dates before 2000 can be two- or four-digit years. After 2000 they must be four-digit years. The expiration date is fully year-2000 compliant. FLEX*lm* v6 (or later) applications only: the keyword “permanent” can be used instead of the date 1-jan-0.

num_lic	Number of concurrent licenses for this feature. If the number of licenses is set to 0, the licenses for this feature are uncounted and no `lmgrd` is required but a hostid on the FEATURE line is required. See Section 4.4, “Counted vs. Uncounted Licenses.” FLEX*lm* v6 (or later) applications only: the keyword “uncounted” can be used instead of a license count of `0`.
license_key	License key to authenticate this FEATURE line. Two identical-looking FEATURE or INCREMENT lines may have different license keys if the start dates are different.
SIGN=*sign*	SIGN= signature to authenticate this FEATURE line. (v7.1+)

The following fields are all optional (except for "*vendor_str*" in the v2 format). For optional fields of the *keyword=value* syntax, if the *keyword* is lowercase, it can be modified and the license will remain valid.

"*vendor_str*"	Vendor-defined string, enclosed in double quotes. This string can contain any characters except a quote. Required field in the v2 format.
feat_hostid	String returned by `lmhostid`. Used only if the feature is to be bound to a particular host, whether its use is counted or not. Numeric hostids are case insensitive. See Appendix A, “Hostids for FLEXlm-Supported Machines,” for more information.
BORROW[=n]	Enables license borrowing for a particular FEATURE/INCREMENT line. *n* is the maximum number of hours that the license can be borrowed for. The default maximum borrow period is 168 hours, or one week. (v8.0+)

`DUP_GROUP=...`

Duplicate grouping parameter can be specified in the license in FLEX*lm* v4.0 and later. The syntax is:

```
DUP_GROUP=NONE|SITE|[UHDV]
```

```
U = DUP_USER
H = DUP_HOST
D = DUP_DISPLAY
V = DUP_VENDOR_DEF
```

Any combination of UHDV is allowed, and the DUP_MASK is the OR of the combination. For example, DUP_GROUP=UHD means the duplicate grouping is (DUP_USER|DUP_HOST|DUP_DISPLAY), so for a user on the same host and display, additional uses of a feature do not consume additional licenses.

`FLOAT_OK [=`*`server_hostid`*`]`

Enables mobile licensing via FLEX*id* with FLOAT_OK for a particular FEATURE/INCREMENT line. This FEATURE/INCREMENT line must also be node-locked to a FLEX*id*. (V8.0+)
When `FLOAT_OK=`*`server_hostid`* is specified on a FEATURE line:

- The *`server_hostid`* must refer to the same host that appears on the SERVER line of the license file.
- The license server can only be run on the machine with the hostid that `lmhostid` returns equal to the *`server_hostid`* specified with FLOAT_OK.

HOSTID= *`feat_hostid`*

Same as *`feat_hostid`* above, but in the *`keyword=value`* pair syntax (FLEX*lm* v3.0 and later).

`HOST_BASED[=`*`n`*`]`	Host names must be specified in INCLUDE statements in the end-user options file, and the number of hosts is limited to *`num_lic`*, or the number specified in *`=n`*.
`ISSUED=` *`dd-mmm-yyyy`*	Date issued.
`ISSUER="..."`	Issuer of the license.
`NOTICE="..."`	A field for intellectual property notices.
`OVERDRAFT=n`	The overdraft policy allows your vendor to specify a number of additional licenses which users will be allowed to use, in addition to the licenses they have purchased. This allows your users to not be denied service when in a “temporary overdraft” state. Usage above the license limit will be reported by the SAM*report* reporting tool.
`PLATFORMS="..."`	Usage is limited to the listed platforms.
`SN=`*`serial_num`*	Serial number, used to identify FEATURE or INCREMENT lines.
`START=` *`dd-mmm-yyyy`*	Start date.
`SUPERSEDE=` *`"f1 f2 ..."`*	If this appears, all licenses issued before the date specified in `ISSUED=` are *superseded* by this line and become ineffective.
`TS_OK`	FLEX*lm* detects when a node-locked uncounted license is running under Windows Terminal Server. To run on Terminal Server remote nodes, `TS_OK` must be added to the FEATURE line. Without `TS_OK`, a user running on a Terminal Server client will be denied a license. (v8.0+)

`USER_BASED[=n]`	Users must be specified in INCLUDE statements in the end-user options file, and the number of users are limited to *num_lic*, or the number specified in *=n*.
`VENDOR_STRING= "..."`	Same as "*vendor_str*" above, but in *keyword=value* pair syntax.

The following attributes can be changed or deleted by end users. This is indicated by a lowercase *keyword*.

`asset_info= "..."`	Additional information provided by the license administrator for asset management.
`ck=nnn`	Obsolete in v8.0.
`dist_info= "..."`	Additional information provided by the software distributor.
`user_info= "..."`	Additional information provided by the license administrator.
`vendor_info= "..."`	Additional information provided by the software vendor.
`LINGER=n`	The lingering interval for this license (encrypted, but not used)

Examples:

```
FEATURE xyz_app xyzd 2.300 31-dec-2005 20 123456789012 "xyz"
INCREMENT f1 xyzd 1.000 1-jan-0 5 901234567890 \
     HOSTID=INTERNET=195.186.*.* NOTICE="Licensed to XYZ corp"
```

2.2.5 PACKAGE Lines

The purpose of the PACKAGE line is to support two different licensing needs:

- To license a product SUITE, or
- To provide a more efficient way of distributing a license file that has a large number of features, which largely share the same FEATURE line arguments.

A PACKAGE line, by itself, does not license anything—it requires a matching FEATURE/INCREMENT line to license the whole package. A PACKAGE line can be shipped by your software vendor with a product, independent of any licenses. Later, when you purchase a license for that package, one or more corresponding FEATURE/INCREMENT lines will enable the PACKAGE line.

Example:

```
PACKAGE package vendor [pkg_version] COMPONENTS=pkg_list \
  [OPTIONS=SUITE] SIGN=pkg_sign
```

where:

package	Name of the package. The corresponding FEATURE/INCREMENT line must have the same name.
vendor	Name of the vendor daemon that supports this package.
pkg_version	Version of the package (optional in v7.1+). If specified, the enabling FEATURE/INCREMENT line must have the same version.
pkg_sign	License key or SIGN= signature.
pkg_list	List of package components. The format is: *feature*[:*version*[:*num_lic*]] Packages must consist of at least one component. Version and count are optional, and if left out, their values come from the corresponding FEATURE/INCREMENT line. *num_lic* is only legal if OPTIONS=SUITE is not set—in this case the resulting number of licenses will be *num_lic* on the COMPONENTS line multiplied by the number of licenses in the FEATURE/INCREMENT line. Examples: `COMPONENTS="comp1 comp2 comp3 comp4"` `COMPONENTS="comp1:1.5 comp2 comp3:2.0:4"`

`OPTIONS=SUITE`	This is what distinguishes a package suite from a package used to ease distribution. With `OPTIONS=SUITE`, the corresponding feature of the same name as the package is checked out in addition to the component feature being checked out. If `OPTIONS=SUITE` is not set, then the corresponding feature of the same name as the package is removed once the package is enabled; is not checked out when a component feature is checked out.

Examples:

```
PACKAGE suite xyzd 1.0 3B24B2F508CB \
  COMPONENTS="comp1 comp2" OPTIONS=SUITE
FEATURE suite xyzd 1.0 1-jan-0 5 4193E6ABCCCB
```

This is a typical `OPTIONS=SUITE` example. There are two features, “comp1” and “comp2,” which are each version 1.0, each with five non-expiring licenses available. When “comp1” or “comp2” is checked out, “suite” will also be checked out.

```
PACKAGE suite xyzd 1.0 2CBF44FCB9C1 \
     COMPONENTS="apple:1.5:2 orange:3.0:4"
FEATURE suite xyzd 1.0 1-jan-2005 3 321E78A17EC1 SN=123
```

In this example, the component version overrides the feature version, and the number of licenses available for any component is the product of the three licenses for “suite” and the number of licenses for that component. The result is equivalent to:

```
FEATURE c1 xyzd 1.5 1-jan-2005 6 0D3AD5F26BEC SN=123
FEATURE c2 xyzd 3.0 1-jan-2005 12 EB16C5AE61F0 SN=123
```

With FLEX*lm* v6 (or later) applications only the PACKAGE lines can be stored in a separate file which need never be edited.

2.2.6 UPGRADE Lines

```
UPGRADE feature vendor from_feat_version to_feat_version \
     exp_date num_lic [options ... ] SIGN=sign
```

All the data is the same as for a FEATURE or INCREMENT line, with the addition of the *from_feat_version* field. An UPGRADE line removes up to the number of licenses specified from any old version (>= *from_feat_version*) and creates a new version with that same number of licenses.

For example, the two lines:

```
INCREMENT f1 xyzd 1.000 1-jan-2005 5 SIGN=9BFAC0316462
UPGRADE f1 xyzd 1.000 2.000 1-jan-2005 2 SIGN=1B9A308CC0F7
```

would result in three licenses of v1.0 of “f1” and two licenses of v2.0 of “f1.”

An UPGRADE line will operate on the closest preceding FEATURE or INCREMENT line with a version number that is >= *from_feat_version*, and < *to_feat_version*.

Note that UPGRADE lines do not work for node-locked, uncounted licenses before v6.

2.3 Sample License File

This is an example of a license file for a single vendor with two features.

```
SERVER excellent_server 17007ea8 1700
VENDOR xyzd
FEATURE xyz_app1 xyzd 1.000 01-jan-2005 10 1EF890030EAB
FEATURE xyz_app2 xyzd 1.000 01-jan-2005 10 084561FE98B3
```

The license file above would allow the license server “excellent_server” with the hostid “17007ea8” to serve ten floating licenses for “xyz_app1” and “xyz_app2” to any user on the network.

2.4 Types of License Files

License files are created by the software vendor. License files can specify floating (concurrent) usage, node-locked (both counted and uncounted), and any combination of floating, counted, and uncounted.

2.4.1 Floating (Concurrent) Licenses

A *floating license* means anyone on the network can use the licensed software, up to the limit specified in the license file (also referred to as *concurrent usage* or *network licensing*). Floating licenses have no hostids on the individual FEATURE lines. Floating licenses requires an `lmgrd` and a vendor daemon to be running to count the concurrent usage of the licenses.

An example of a license file that provides floating licenses is:

```
SERVER lulu 17001234
VENDOR xyzd
FEATURE f1 xyzd 1.00 1-jan-2005 2 key1 ""
FEATURE f2 xyzd 1.00 1-jan-2005 6 key2 ""
FEATURE f3 xyzd 1.00 1-jan-2005 1 key3 ""
```

This license file specifies that two licenses for feature “f1,” six licenses for feature “f2,” and one license for feature “f3” are available anywhere on the network that can access the license server “lulu.” `lmgrd` uses one of the default FLEX*lm* ports.

2.4.2 Node-Locked Licenses

Node-locking means the licensed software can only be used on one node. A node-locked license has a hostid on any FEATURE line that is node-locked to a particular host. There are two types of node-locked licenses; uncounted and counted.

If the number of licenses is set to 0, then the license is uncounted and unlimited use is permitted on the specified node. This configuration does not require an `lmgrd` or a vendor daemon because it is not going to count the concurrent usage of the features.

The following license file allows unlimited usage of feature “f1” on the nodes with hostids of “12001234” and “1700ab12”:

```
FEATURE f1 xyzd 1.000 1-jan-2005 0 key1 "" 12001234
FEATURE f1 xyzd 1.000 1-jan-2005 0 key2 "" 1700ab12
```

Alternately, in FLEX*lm* v5.0 or later, these two FEATURE lines could have been issued by your software vendor with a *hostid list*:

```
FEATURE f1 xyzd 1.000 1-jan-2005 0 key HOSTID="12001234 1700ab12"
```

If these were the only FEATURE lines in this license file, no `lmgrd` daemon would be necessary and you should not start one.

The following license file allows three licenses for feature “f1” to be run, but only on the node with hostid “1300ab43.” In this case, the daemons should be run on the same node that runs the software, since there is no reason to run the daemons on another node.

```
SERVER lulu 1300ab43 1700
DAEMON xyzd /etc/xyzd
FEATURE f1 xyzd 1.00 1-jan-2005 3 key HOSTID=1300ab43
```

2.4.3 Mixed Node-Locked and Floating Licenses

Uncounted node-locked and concurrent usage licenses can be mixed in the same license file.

The following license file allows unlimited use of feature “f1” on nodes “17001111” and “17002222,” while allowing two other licenses for feature “f1” to be used anywhere else on the network:

```
SERVER lulu 17001234 1700
DAEMON xyzd C:\flexlm\xyzd.exe
FEATURE f1 xyzd 1.00 1-jan-2005 0 key1 HOSTID=17001111
FEATURE f1 xyzd 1.00 1-jan-2005 0 key2 HOSTID=17002222
FEATURE f1 xyzd 1.00 1-jan-2005 2 key3
```

This configuration requires an `lmgrd` and a vendor daemon because the concurrent usage of the two licenses on the third FEATURE line is counted.

2.5 Decimal Format

The decimal format was introduced in v6. Users with older products can still use the decimal format, but they will require a copy of the `lminstall` command (which is part of `lmutil`). The `lminstall` utility allows the user to type in a decimal line, which is then converted to the readable format, and saved in the specified location. The mixed node-locked and floating example above looks as follows in decimal format:

```
xyzd-f1-01761-55296-37046-04544-00017-06551-18072-57346-18754-136
xyzd-f1-01761-55296-37046-08896-00034-235
xyzd-f1-00481-55296-17590-2
```

A simple demo license in readable format:

```
FEATURE f1 xyzd 1.00 1-jan-2005 0 key1 HOSTID=DEMO
```

converted to decimal:

```
xyzd-f1-00737-55296-1825
```

Note that by default `lminstall` converts to v6 format. It can convert to a format compatible with older versions by using `lminstall -overfmt 2` (or `3`, `4`, `5`, `5.1`, `6`, `7`, `7.1`, or `8` depending on the FLEX*lm* version).

2.6 License File Order

In some cases, the ordering of lines in a license file can be crucial. Version 7.0+ vendor daemons and clients automatically internally sort the lines so that in most cases the optimal result is achieved. For earlier versions of FLEX*lm*, note the following suggestions, which are all based on the fact the checkouts are attempted on lines in the order they appear in the license file:

- Place FEATURE lines before INCREMENT lines for the same feature.

  The rule regarding FEATURE lines is that only the first counted FEATURE line is observed by the license server, and that if there is a FEATURE line and INCREMENT lines, the FEATURE line must appear first. (A few large, older FLEX*lm*-licensed companies have FEATURE lines that behave identically to INCREMENT lines, and only the rules applying to INCREMENT apply to them.)

- Where multiple counted FEATURE lines exist for the same feature, make sure the desired FEATURE line appears first.

  All but the first is ignored.

- Place node-locked, uncounted lines before floating lines for the same FEATURE.

  Otherwise, the floating license may be consumed when a node-locked would have been used, resulting in denial for other users.

- The placement of a USE_SERVER line affects behavior.

  A USE_SERVER line is recommended. Normally, the USE_SERVER line is placed immediately after the SERVER line. However, if there are uncounted licenses which you don’t want to fail if the server is unavailable, these should be placed at the top of the file, with the USE_SERVER line following them. This only works, however, if each user that needs the uncounted license has direct access to a current copy of the file. The advantage to placing USE_SERVER right after the SERVER line is users don’t need up-to-date copies of the license file.

Chapter 3

Multiple License Files

Since more than 2000 vendors have chosen FLEX*lm* as their license manager, chances are good that you will have to administer FLEX*lm* licenses from more than one vendor or multiple products from the same vendor.

3.1 Overview of Combining License Files

When you are running FLEX*lm*-licensed products from multiple vendors, you may need to take steps to prevent licensing conflicts during installation. There are three ways you can accomplish this:

- Multiple license server nodes, each running one `lmgrd` and one license file
- One license server node running one `lmgrd` and several license files
- One license server node running multiple `lmgrd`s and multiple license files

Note that before v6, each `lmgrd` could read only a single license file. In the first option mentioned above, you will have more license servers to monitor; in the third option you have only one server but multiple `lmgrd`s to administer.

If all applications and vendor daemons are FLEX*lm* v6.0+, `lmgrd` can process multiple license files, even when the hostids in each license file are different, so long as they refer to the same machine. For example, on UNIX:

```
lmgrd -c license_file_path_1:license_file_path_2 ...
```

Your product's license file(s) define the license server(s) by host name and hostid in the SERVER line(s) in the license file. If you have two or more products whose license servers run on the same node (as specified by the SERVER lines in the license files), you may be able to combine the license files into a single license file. You can combine two license files under the following conditions:

- The number of SERVER lines in each file is the same.
- The hostid field of each SERVER line in one file *exactly* matches the hostid field of each SERVER line in the other file.

Some possible reasons license files may not be compatible are:

- License files are set up to run on different server nodes, so hostids are different.
- One file is set up for single server (has only one SERVER line), the other is set up for a three-server redundant license server (has multiple SERVER lines).
- One vendor uses a custom hostid algorithm, so the hostids on the SERVER lines are different, even though the files are for the same machine.

If your license files are compatible as described above, then you have the option of combining license files and running a single `lmgrd`, as described below in Section 3.1.1, "Combining License Files from Multiple Vendors." If the license files are not compatible, then you must keep the license files separate and run separate copies of `lmgrd` for each license file, as described in Section 3.1.3, "Using Separate License Files on the Same Server Node."

Note that you are not required to combine compatible license files; you always have the option of running separate `lmgrd`s, and there is virtually no performance or system-load penalty for running separate `lmgrd` processes.

3.1.1 Combining License Files from Multiple Vendors

If your license files are compatible, you can combine them with any text editor. To combine license files, read all of the compatible license files into one file, then edit out the extra SERVER lines so that only one set of SERVER lines remains. Write out this data, and you have your combined license file. If you combine license files from multiple vendors, it is a good idea to keep a copy of the combined license file in each vendor's default license file location. This way, users can avoid having to set `LM_LICENSE_FILE`, because each package finds it's license information in the default place. On UNIX, you can do this with a symbolic link from each default location to the location of the combined license file.

3.1.2 FLEX*lm* Version Component Compatibility

When you combine license files for two different FLEX*lm*-licensed products, it may be the case that those products do not use the same version of FLEX*lm*. FLEX*lm* is designed to handle this situation. There are two basic compatibility rules for FLEX*lm*:

- A newer `lmgrd` can be used with an older vendor daemon, but a newer vendor daemon might not work properly with an older `lmgrd`.

- A newer vendor daemon (or `lmgrd`) can be used with an older client program, but a newer client program might not work properly with an older vendor daemon.

From these two compatibility rules come the simple rules for selecting which version of administration tools to use:

- Always use the newest version of `lmgrd` and the newest version of each vendor daemon.
- Use the newest FLEX*lm* utilities.

For specific application programs, you can use either the new or the old version (of course, the vendor daemon for that application must be at least as new as the application itself). See also Section H.1, “Version Compatibility and Components.”

3.1.3 Using Separate License Files on the Same Server Node

You must run a separate copy of `lmgrd` for each license file. When you run multiple copies of `lmgrd`, there are two details to remember:

1. The port number on the SERVER line of each license file must be unique. You can use a standard text editor to change the port number in each license file so that they are all different.
2. You must make sure that you are using a compatible version of `lmgrd` when you start it up for a particular license file. This can be done by using an explicit path to `lmgrd`.

When running client programs (such as a licensed application), you can set the `LM_LICENSE_FILE` environment variable to point to multiple license files. For example, you may have a license file from vendor “ABC” and a license file from vendor “XYZ” with incompatible servers. You can place the license file from vendor “ABC” into:

```
/usr/flexlm/abc.lic
```

and the license file from vendor “XYZ” into:

```
/usr/flexlm/xyz.lic
```

then set the `LM_LICENSE_FILE` environment variable to point to both of them. Each name in `LM_LICENSE_FILE` should be separated by a colon (“ : ”) on UNIX systems and a semicolon (“ ; ”) on Windows and Windows/NT systems.

In the C shell:

```
setenv LM_LICENSE_FILE /usr/flexlm/abc.lic:/usr/flexlm/xyz.lic
```

In the Korn and Bourne shells:

```
LM_LICENSE_FILE=/usr/flexlm/abc.lic:/usr/flexlm/xyz.lic
export LM_LICENSE_FILE
```

3.2 Using LM_LICENSE_FILE License-File List

If products use different license server nodes, each set of license servers requires separate license files. (When multiple software vendors use the same set of license server nodes, the technique described above in Section 3.1, “Overview of Combining License Files,” can be used to combine license files.) The resulting (multiple) license files can be installed in convenient locations. On UNIX you would set the `LM_LICENSE_FILE` environment variable as follows:

`setenv LM_LICENSE_FILE` *`lfpath1`*`:`*`lfpath2`*`:...`

where *`lfpath1`* is the path to the first license file, *`lfpath2`* is the path to the second license file, etc.

Note: Use a colon (“ : ”) to separate the license file names on UNIX and on Windows use a semicolon (“ ; ”).

Each application queries each license file in the order it is listed in `LM_LICENSE_FILE`. If the license server serving the license file listed in *`lfpath1`* is unreachable, the other files listed in `LM_LICENSE_FILE` allow a user to obtain a license from another server. *`lfpathn`* can also be *`port@host`*, using the port number and host name from the SERVER line in the license file.

See also:

- Section 4.3.1, “Redundancy via License-File List”
- Appendix C, “FLEXlm Environment Variables”
- Section 2.1.2, “Setting the Path with an Environment Variable”

Chapter 4

Selecting Server Nodes

This chapter helps you decide which nodes to use as license server nodes.

4.1 Resources Used by the Server

This section discusses the resources used by the license server. When you select a server node, you may need to take into account the system limits on these resources. For small numbers of licenses (under about 100), most of these items should not be a problem on any workstation.

4.1.1 Sockets

When using TCP, a single vendor daemon can support as many users as the per-process system limit for file descriptors, which ranges from 256 on SunOS 4.x to 4000 on DEC Alpha. When no more file descriptors are available to a daemon, additional vendor daemons are spawned to allow for extra file descriptors, though this is not recommended. When using UDP, there is no limit to the number of end users per vendor daemon process, because they can share a single socket in the vendor daemon (UDP has other drawbacks, and TCP is preferred). If there are more than 250 concurrent clients from a SunOS vendor daemon, it may be a good idea to move the server to a different OS, since all other OSs support more file descriptors. If there are more than 1000 concurrent clients being supported by a single vendor daemon, then it's probably good to split the license file into more than one file, from different servers, to lighten the networking traffic (which will require the vendor to agree to issue new licenses). Clients can check out licenses from multiple servers using a license-file list via the `LM_LICENSE_FILE` environment variable.

Each client connected to a license server uses one socket. The total number of sockets used by the license server programs is slightly larger than the total number of simultaneous clients.

On older SCO systems, the default number of sockets may be set fairly low; if you choose to run a server on such a machine, you may need to reconfigure your kernel to have more sockets.

The number of sockets available for Windows 95 clients is about 60. In general, NT is preferred for server systems, where there is no such limit, and the operating system is better designed for server processes.

4.1.2 CPU Time

For small numbers of clients, the license servers use very little CPU time. The servers might have only a few seconds of CPU time after many days.

For a large number of clients (who are each exchanging heartbeat messages with the server), or for high checkout/checkin activity levels (hundreds per second), the amount of CPU time consumed by the server may start to become significant, although, even here, CPU usage is normally not high. In this case, you may need to ensure that the server machine you select will have enough CPU cycles to spare.

Note: GLOBE*trotter* Software has rarely encountered a situation where CPU cycles were an issue.

4.1.3 Disk Space

The only output files created by the license servers are the debug and report log files. The report log files are used to generate accurate usage reports by SAM*report*. These log files contain one line for each checkout and one line for each checkin. If you have a lot of license activity, these log files will grow very large. You will need to consider where to put these files and how often to delete or prune them. The license administrator can opt not to log messages to the debug log file if disk space is at a premium. See Section 6.6.1, “Managing Debug Log Output,” and Section 6.6.2, “Managing Report Log Output,” for more details.

Note that the log files should be local files on the server machine(s) to avoid networking dependencies.

4.1.4 Memory

The FLEX*lm* daemons use little memory. On SunOS, `lmgrd` uses approximately2 MB and the vendor daemons use approximately 2 MB each, although memory usage increases in the vendor daemon with the size of the license file and the number of concurrent users.

4.1.5 Network Bandwidth

FLEX*lm* sends relatively small amounts of data across the network. Each transaction, such as a checkout or checkin, is typically satisfied with less than 1 KB of data transferred. This means that FLEX*lm* licensing can be effectively run over slow networks (such as dial-up SLIP lines) for small numbers of clients.

For a large number of clients (hundreds), each of which will be exchanging heartbeat messages with the vendor daemon, the network bandwidth used may start to become significant. In this case you should run client and server on the same local area network, which may require splitting licenses between two files for two servers. Users can use a license-file list in the `LM_LICENSE_FILE` environment variable to have effective access to both servers.

In high-traffic networks, with FLEX*lm* clients older than v5, you may also want to avoid setting `LM_LICENSE_FILE` to a *`port@host`* address. Instead, the license administrator should place a copy of the license file in a file system local to the application. See Section 2.1, “Specifying Location of the License File.”

4.2 Remote Mounted Disks

GLOBE*trotter* Software recommends that you do not use remote mounted disks when you run the license server. In other words, it is recommended that `lmgrd`, the vendor daemons, the license file, and the debug and report log files are all on locally mounted disks. If any of these files is on a remote mounted disk, you double the points of failure which could lead to a temporary loss of all of your licenses. When all files are mounted locally, the licenses will be available as long as the server machine is up; but when the files are on a different machine, then the loss of either the license server machine or the file server machine will cause the licenses to be unavailable.

4.3 Redundant License Servers

If all the end-user data is on a single file server, then there is no need for redundant license 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 for one reason or another. Redundant license server nodes can be any supported server nodes—it is not required that they be the same architecture or operating system.

FLEX*lm* supports two methods of redundancy: redundancy via a license-file list in the `LM_LICENSE_FILE` environment variable and a set of three redundant license servers.

With `LM_LICENSE_FILE` list redundancy, each one of a group of license servers serves a subset of the total licenses. The end user sets `LM_LICENSE_FILE` to a list of license files, where each license file refers to one of the license servers. The application then tries each server in the list, in order, until it succeeds or gets to the end of the list.

With three-server redundancy, if any two of the three license servers are up and running (two out of three license servers is referred to as a *quorum*), the system is functional and hands out its total complement of licenses.

See also Section 3.2, “Using LM_LICENSE_FILE License-File List.”

4.3.1 Redundancy via License-File List

This is best explained by example. If ten licenses are desired for both “f1” and “f2,” the vendor would issue two sets of licenses with a count of 5 for each of “f1” and “f2.” The server nodes (unlike three-server redundancy) can be physically distant.

The license files would look like:

License 1 for “chicago”

```
SERVER chicago 17007ea8 1700
DAEMON xyzd /etc/mydaemon
FEATURE f1 xyzd 1.000 01-jan-2005 5 26C7DD9C0186
FEATURE f2 xyzd 1.000 01-jan-2005 5 8CE46C57041D
```

License 2 for “tokyo”

```
SERVER tokyo 17a07e08 1700
DAEMON xyzd /etc/mydaemon
FEATURE f1 xyzd 1.000 01-jan-2005 5 16BE40E1D98D
FEATURE f2 xyzd 1.000 01-jan-2005 5 6DB6F3E402DF
```

The user in Chicago could set `LM_LICENSE_FILE` to:

```
1700@chicago:1700@tokyo
```

The user in Tokyo could set `LM_LICENSE_FILE` to:

```
1700@tokyo:1700@chicago
```

Remember to separate the license file names with a colon (“ : ”) on UNIX and with a semicolon (“ ; ”) on Windows. The application attempts the first server in the list, and if that fails for any reason, the second server is tried.

4.3.2 Three-Server Redundancy

These three-server redundant servers should have excellent communications and should be on the same subnet. Often this means that the three servers should be located physically close to each other. This form of redundancy requires that the servers exchange heartbeats periodically, and poor communications can cause poor performance. You should never configure redundant servers with slow communications or dial-up links.

Three-server redundancy does not provide load-balancing. Use `LM_LICENSE_FILE` list instead for this type of redundancy. This is because with three-server redundancy, only one of the three servers is “master,” capable of issuing licenses. Since all clients must contact the “master,” all clients must have reliable networking to a single node.

4.3.3 Comparing Three-Server to License-File List

ARE THERE ANY DRAWBACKS TO USING THE LICENSE-FILE LIST FOR REDUNDANCY?

Yes. By default, once a *license job* has successfully checked out a license from one host, all subsequent checkouts must be satisfied from the same host. If the application requires more than one license, this could result in a license denial when the license is available on another server. An application can bypass this restriction if it is coded with the use of multiple FLEX*lm* license jobs. Only your application vendor can tell you if their application is programmed in this manner.

If the application supports license queueing, all licenses are only queued from the first host on the list.

Finally, if one server becomes unavailable, some licenses will be unavailable.

WHEN IS IT RECOMMENDED TO USE A LICENSE-FILE LIST FOR REDUNDANCY RATHER THAN THREE-SERVER REDUNDANT SERVERS?

- When there’s less system administration available to monitor license servers.
- When load-balancing is needed for clients located far apart, e.g., London and Tokyo. You can make servers available locally, with remote servers available as backup.
- License-file list is more forgiving if you lose quorum.
- License-file list is not limited to three servers (any number will work).

- Clients do not require reliable networking to a single node with license-file list, so this is recommended where networking itself requires redundancy.

4.4 Counted vs. Uncounted Licenses

The license file determines whether a license server is needed. If all the FEATURE (or INCREMENT) lines have a license count of 0 (unlimited) or “uncounted” (FLEX*lm* v6 or later only), then no server is needed. This type of license is called uncounted. Alternatively, if any FEATURE lines have a non-zero license count, then a server is required to count those licenses. If a vendor wants to use FLEX*lm* without a server, they must issue uncounted licenses.

With FLEX*lm* v5 or later, the license server can serve uncounted licenses as well. This is done so that the report log file will include transactions for all license requests, which can then be reported on by SAM*report*. To do this, include a SERVER line in the license file, and put the USE_SERVER line immediately after the SERVER line. The vendor daemon will serve the uncounted licenses, and the USE_SERVER line indicates to applications that they will be authorized by the server.

Chapter 5

The Options File

The options file allows the license administrator to control various operating parameters of FLEX*lm*. Users can be identified by their user name, host name, display, IP address, or PROJECT (which is set with the `LM_PROJECT` environment variable).

Specifically, the license administrator can:

- Allow the use of features
- Deny the use of features
- Reserve licenses
- Control the amount of information logged about license usage
- Enable a report log file

Options files allow you, as the license administrator, to be as secure or open with licenses as you like.

Lines in the options file are limited to 2048 characters. The "\" character is a continuation character in options file lines.

Identifying users by `LM_PROJECT` *in the options file* requires a v7.0+ vendor daemon.

5.1 Creating an Options File

To create an options file:

1. Use the appropriate options listed in Section 5.2, "Options File Syntax," to create the options file using any text editor. You can put the options file anywhere; however, it is recommended that the options file for vendor "XYZ" (whose vendor daemon is `xyzd`) be placed in:

   UNIX: `/a/b/xyzd/licenses/xyzd.opt`

   Windows: `C:\a\b\xyzd\licenses\xyzd.opt`

2. Add the path to the options file in the license file as the fourth field on the DAEMON line for the application's vendor daemon. For example:

   ```
   DAEMON xyzd /etc/xyzd /a/b/xyzd/licenses/xyzd.opt
   ```

   would enable the `xyzd` vendor daemon to look at the specified options file. In FLEX*lm* v5 or later, you can put `options=` before the path. In FLEX*lm* v6 or later, adding the options path to the license file is optional. Instead, name the file *`vendor`*`.opt`, where *`vendor`* is the vendor daemon name, and place it in the same directory as the license used by `lmgrd`, and it will automatically be used at server startup.

5.2 Options File Syntax

Below is an overview of the options file syntax. See Section 5.5, "Options File Examples," for examples and additional information.

Each line of the file controls one option. The options are:

BORROW_LOWWATER	Set the number of BORROW licenses that cannot be borrowed.
DEBUGLOG	Writes debug log information for this vendor daemon to the specified file (v8.0+ vendor daemon).
EXCLUDE	Deny a user access to a feature.
EXCLUDE_BORROW	Deny a user the ability to borrow BORROW licenses.
EXCLUDEALL	Deny a user access to *all* features served by this vendor daemon.
GROUP	Define a group of users for use with any options.
HOST_GROUP	Define a group of hosts for use with any options (v4.0+).
INCLUDE	Allow a user to use a feature.
INCLUDE_BORROW	Allow a user to borrow BORROW licenses.

INCLUDEALL	Allow a user to use *all* features served by this vendor daemon.
MAX	Limit usage for a particular feature/group—prioritizes usage among users.
MAX_OVERDRAFT	Limit overdraft usage to less than the amount specified in the license.
NOLOG	Turn off logging of certain items in the debug log file.
REPORTLOG	Specify that a report log file suitable for use by the SAM*report* license usage reporting tool be written.
RESERVE	Reserve licenses for a user or group of users/hosts.
TIMEOUT	Specify idle timeout for a feature, returning it to the free pool for use by another user.
TIMEOUTALL	Set timeout on all features.

You can include comments in your options file by starting each comment line with a pound sign "#." Everything in an options file is case sensitive. Be sure that user names and feature names, for example, are entered correctly.

FEATURE SPECIFICATION

When there are multiple pools of licenses for a single feature, you will need to specify a particular pool of licenses in the options file. You can specify options for a particular FEATURE/INCREMENT line of a given feature name, as follows:

```
feature:keyword=value
```

For example:

```
f1:VERSION=2.0
```

A colon (:) is a valid feature name character if you are using a v8.0+ vendor daemon. If you use a colon in your feature names, you can specify a pool of licenses with the following alternative syntax using quotation marks and spaces:

```
"feature keyword=value"
```

You can specify a feature by any of the following fields:

VERSION=, HOSTID=, EXPDATE=, KEY=, SIGN=, VENDOR_STRING=, ISSUER=, NOTICE=, dist_info=, user_info=, asset_info=

In FLEX*lm* v5.11 or later, you can use a package name in place of a feature name, and the option will apply to all of the components in the package.

The following option keywords restrict who may use licenses or where licenses may be used: EXCLUDE, EXCLUDEALL, INCLUDE, INCLUDEALL, MAX, and RESERVE. These options take a *type* argument, which specifies whether the restriction is based on USER, HOST, DISPLAY, INTERNET, or PROJECT:

- USER—UNIX user name of the user executing the program
- HOST—machine where the application is executing

  Anywhere a host name can be used in an options file, an IP-address of the format `#.#.#.#` can be used instead (v8.0+ vendor daemon). The IP-addresses can contain wildcards.
- DISPLAY—display where the application is displayed

  DISPLAY is often confusing. Historically, the FLEX*lm* default on UNIX was `/dev/tty`*xx* (which is always `/dev/tty` when an application is run in the background). Some newer applications now have it set to the X-Display name.
- INTERNET—IP address of the machine where the application is executing (wildcards can be used in the IP address)
- PROJECT—`LM_PROJECT` environment variable set by the user who is executing the program (v7.0+ vendor daemon)

On PCs, the USER, HOST and DISPLAY name are all set to the PC's host name.

The types listed above take a list of members, separated by spaces. For example:

```
EXCLUDE coolsoft USER joe barbara susan
```

Other valid types are GROUP and HOST_GROUP. These allow you to define a group of users or hosts on a separate line in the options file, then use those group names instead of an explicit list. For example:

```
GROUP stars joe barbara susan
EXCLUDE coolsoft GROUP stars
```

5.2.1 BORROW_LOWWATER

```
BORROW_LOWWATER feature[:keyword=value] n
```

Sets the number of licenses for a BORROW feature that cannot be borrowed.

feature	Name of feature being affected.
n	Number of licenses that cannot be borrowed via license borrowing.

For example, if a feature "f1" has a count of 10 and borrowing is enabled in the application and on the FEATURE line:

```
FEATURE f1 ... 10 ... BORROW SIGN=...
```

the following line in the options file will allow only 7 licenses to be borrowed.

```
BORROW_LOWWATER f1 3
```

5.2.2 DEBUGLOG

```
DEBUGLOG debug_log
```

Specifies a location for the debug log output for one vendor daemon. *debug_log* should be the full path of the debug log. Note that this affects only the vendor daemon associated with this options file. The debug log output of `lmgrd` and any other vendor daemons in the same license file is unaffected (v8.0+ vendor daemon).

SEE ALSO:

- Section 6.14, "lmswitch"

5.2.3 EXCLUDE

```
EXCLUDE feature[:keyword=value] type {list | group_name}
```

Excludes a list or pre-defined group of users, etc., from the list of who is allowed to use the feature. EXCLUDE overrides INCLUDE.

feature	Name of the feature being affected.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to exclude.

group_name	Name of the group to exclude.

To exclude the user “hank” from the list of users able to use feature “f1”:

```
EXCLUDE f1 USER hank
```

5.2.4 EXCLUDE_BORROW

```
EXCLUDE_BORROW feature[:keyword=value] type {list | group_name}
```

Excludes a list or pre-defined group of users, etc., from the list of who is allowed to borrow licenses for this BORROW feature. EXCLUDE_BORROW overrides INCLUDE_BORROW.

feature	Name of the feature being affected.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to exclude.
group_name	Name of the group to exclude.

To exclude the user “fred” from the list of users able to borrow feature “f1” assuming the feature has the BORROW attribute:

```
EXCLUDE_BORROW f1 USER fred
```

5.2.5 EXCLUDEALL

```
EXCLUDEALL type {list | group_name}
```

Excludes a list or pre-defined group of users, etc., from the list of who is allowed to use all features served by this vendor daemon.

type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to exclude.
group_name	Name of the group to exclude.

To exclude any user on the machine “chaos” from using all features served by this vendor daemon:

```
EXCLUDEALL HOST chaos
```

5.2.6 GROUP

```
GROUP group_name user_list
```

Defines a group of users for use in INCLUDE, INCLUDEALL, EXCLUDE, EXCLUDEALL, and RESERVE option lines.

group_name	Name of the group being defined.
user_list	List of user names in that group.

To define the group “Hackers” consisting of “bob,” “howard,” and “james”:

```
GROUP Hackers bob howard james
```

Multiple GROUP lines for the same group name will add all the specified users into the group.

Note: USER_GROUP is an alias for GROUP.

5.2.7 HOST_GROUP

```
HOST_GROUP group_name host_list
```

Defines a group of hosts for use in INCLUDE, INCLUDEALL, EXCLUDE, EXCLUDEALL, and RESERVE option lines. Multiple HOST_GROUP lines will add all the specified hosts into the group.

group_name	Name of the group being defined.
host_list	List of host names in that group.

To define the host group “Pacific” consisting of “tokyo,” “seattle,” and “auckland”:

```
HOST_GROUP Pacific tokyo seattle auckland
```

As of v8.0, anywhere a host name can be used in an options file, an IP-address can be used instead.

5.2.8 INCLUDE

```
INCLUDE feature[:keyword=value] type {list | group_name}
```

Includes a list or pre-defined group of users, etc., in the list of who is allowed to use licenses for this feature. Anyone not in an INCLUDE statement will not be allowed to use that feature. EXCLUDE overrides INCLUDE.

feature	Name of the feature being affected.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to include.
group_name	Name of the group to include.

To include user “bob” in the list of users able to use feature “f1”:

```
INCLUDE f1 USER bob
```

Note: INCLUDE is required for USER_BASED or HOST_BASED features. The system administrator specifies which users are allowed to use the product, via INCLUDE, and the license limits the number of users that can be INCLUDEd.

5.2.9 INCLUDE_BORROW

```
INCLUDE_BORROW feature[:keyword=value] type {list | group_name}
```

Includes a list or pre-defined group of users, etc., in the list of who is allowed to borrow the BORROW feature. Anyone not in an INCLUDE_BORROW statement will not be allowed to borrow licenses. EXCLUDE_BORROW overrides INCLUDE_BORROW.

feature	Name of the feature being affected.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to include.
group_name	Name of the group to include.

To include user “tom” in the list of users able to borrow feature “f1”:

```
INCLUDE_BORROW f1 USER tom
```

5.2.10 INCLUDEALL

```
INCLUDEALL type {list | group_name}
```

Includes a list or pre-defined group of users, etc. in the list of who is allowed to use all features served by this vendor daemon. Anyone not in an INCLUDEALL statement will not be allowed to use these features.

type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to include.
group_name	Name of the group to include.

To allow the user “jane” to use all features served by this vendor daemon:

```
INCLUDEALL USER jane
```

5.2.11 MAX

(v5.11+ vendor daemon only.)

```
MAX num_lic feature[:keyword=value] type {list | group_name}
```

Limits usage for a group or user.

num_lic	Usage limit for this user or group.
feature	Feature this limit applies to.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to limit.
group_name	Name of the group to limit.

5.2.12 MAX_OVERDRAFT

```
MAX_OVERDRAFT feature[:keyword=value] num_lic
```

Limits OVERDRAFT license usage below the OVERDRAFT allowed by the license file.

5.2.13 NOLOG

```
NOLOG { IN | OUT | DENIED | QUEUED }
```

Suppresses logging the selected type of event in the debug log file.

To turn off logging of checkins:

```
NOLOG IN
```

To turn off logging of checkouts and queued requests two separate NOLOG lines are required:

```
NOLOG DENIED
NOLOG QUEUED
```

Note: License administrators might use this option to reduce the size of the debug log file.

See also Section 6.14, “lmswitch.”

5.2.14 REPORTLOG

```
REPORTLOG [+]report_log_path
```

REPORTLOG specifies the report log file for this vendor daemon. It is recommended preceding the *report_log_path* with a + character to append logging entries, otherwise the file will be overwritten each time the daemon is started.

Note: SAM*report*, a separate product available from GLOBE*trotter*, is used to process FLEX*lm* report log files. SAM*report* can process only report log files, not debug log files.

REPORTING ON PROJECTS WITH LM_PROJECT

The SAM*report* report writer can report on “projects.” A project is set up by having all users working on a project set their LM_PROJECT environment variable (or registry on Windows) to a string that describes the project. SAM*report* can then group usage by project, as defined by what LM_PROJECT was set to when the application was run.

See also Appendix C, “FLEXlm Environment Variables.”

5.2.15 RESERVE

```
RESERVE num_lic feature[:keyword=value] type
  {list | group_name}
```

Reserves licenses for a specific user.

num_lic	Number of license to reserve for this user or group.
feature	Feature this reservation applies to.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP.
list	List of *type* members to reserve licenses for.
group_name	Name of group to reserve licenses for.

To reserve one license of feature “f1” for user “mel”:

```
RESERVE 1 f1 USER mel
```

If you want to reserve a license for *each* of several users or groups, you must use a separate RESERVE line for each user or group.

Note: Any licenses reserved for a user are dedicated to that user. Even when that user is not actively using the license it will be unavailable to other users. However, a RESERVEd license will not cause an overdraft to be reported by SAM*report* if the license is not actually in use.

5.2.16 TIMEOUT

```
TIMEOUT feature[:keyword=value] seconds
```

Sets the time after which an inactive license is reclaimed by the vendor daemon.

feature	Name of the feature.
seconds	Number of seconds after which inactive license is reclaimed.

To set the timeout for feature “f1” to one hour (3600 seconds):

```
TIMEOUT f1 3600
```

TIMEOUT checks in the licenses if the process has been “idle” for a period longer than the specified time period. The daemon declares a process idle when it has not heard from the process (the client sends heartbeats). The application must explicitly declare itself idle for this to work, or (on UNIX) the application must be stopped (^Z). That is, unless the application explicitly supports this feature, it will not work. Contact your software vendor for information about how they implemented this feature in their product.

The application vendor can also disable the timeout feature, in which case the TIMEOUT option has no effect. The vendor can set a minimum value for the timeout. If you specify a timeout value smaller than the minimum, the minimum is used. The default minimum value is 900 seconds (15 minutes).

If you do not specify a timeout value in your options file, then there will be no timeout for that feature. With a pre-v5 vendor daemon, licenses are only freed by TIMEOUT when a new request for a license would require a license that can be freed with TIMEOUT. With v5.0+, licenses are freed when they time out.

5.2.17 TIMEOUTALL

(v5.11+ vendor daemon only.)

```
TIMEOUTALL seconds
```

Same as TIMEOUT, but applies to all features.

5.3 How the Vendor Daemon Uses the Options File

When the vendor daemon is started by `lmgrd`, the vendor daemon reads its options file. There can only be one options file per vendor daemon and each vendor daemon needs its own options file. For any changes in an options file to take effect, the vendor daemon must read its options file. `lmreread` causes the vendor daemon to reread its options file (v8.0+ vendor daemon); if you are using earlier versions of FLEX*lm*, the vendor daemon must be restarted.

5.4 Rules of Precedence in Options Files

Before you can use options to utilize licenses effectively you must understand the options file precedence. INCLUDE and EXCLUDE statements can be combined in the same options file and control access to the same features. When doing so, keep in mind the following:

- If there is only an EXCLUDE list, everyone who is not on the list will be allowed to use the feature.
- If there is only an INCLUDE list, only those users on the list will be allowed to use the feature.
- If neither list exists, everyone is allowed to use the feature.
- The EXCLUDE list is checked before the INCLUDE list; someone who is on both lists will not be allowed to use the feature.

Once you create an INCLUDE or EXCLUDE list, everyone else is *implicitly* "outside" the group. This feature allows you, as an administrator, the ability to control licenses without having to *explicitly* list each user that you wish to allow or deny access to. In other words, there are two approaches; you can either:

- Give most users access and list only the exceptions, or
- Severely limit access and list only the those users that have access privileges

5.5 Options File Examples

The following information gives some examples of options files intended to illustrate ways to effectively control access to your licenses.

5.5.1 Simple Options File Example

```
RESERVE 1 compile USER robert
RESERVE 3 compile HOST mainline
EXCLUDE compile USER lori
NOLOG QUEUED
```

This options file would:

- Reserve one license for the feature "compile" for the user "robert."
- Reserve three licenses for the feature "compile" for anyone on a computer with the host name "mainline."
- Prevent the user "lori" from using the "compile" feature on any node on the network.
- Cause QUEUED messages to be omitted from the debug log file.

The sum total of the licenses reserved must be less than or equal to the number of licenses specified in the FEATURE line. In the example above, there must be a minimum of four licenses on the "compile" FEATURE line. If fewer licenses are available, only the first set of reservations (up to the license limit) is used.

If this data were in file */a/b/*xyzd/licenses/xyzd.opt, then you would modify the license file DAEMON line as follows:

```
DAEMON xyzd /etc/xyzd /a/b/xyzd/licenses/xyzd.opt
```

5.5.2 Limiting Access for Multiple Users

Each INCLUDE, INCLUDEALL, EXCLUDE, EXCLUDEALL, and RESERVE line must have a single user name (or group) listed. To affect more than one user name create a GROUP. For example to exclude “bob,” “howard,” and “james” from using the feature called “toothbrush,” create the following options file:

```
EXCLUDE toothbrush USER bob
EXCLUDE toothbrush USER howard
EXCLUDE toothbrush USER james
```

However, there is an easier way. Create a GROUP and exclude the list of users from using the feature. Like the previous example, the following options file would exclude “bob,” “howard,” and “james” from using the feature called “toothbrush”:

```
# First define the group "Hackers"
GROUP Hackers bob howard james
# Then exclude the group
EXCLUDE toothbrush GROUP Hackers
```

Now when you want to allow or deny access to any feature to that group, you have an alias list to make it simple.

With a FLEX*lm* v4.0+ vendor daemon, you can use HOST_GROUP to allow, deny, or reserve licenses for multiple hosts. For example, to exclude all users logged in on the hosts “fred” and “barney” from using a feature called “f1,” add these lines to your options file:

```
HOST_GROUP writers fred barney
EXCLUDE f1 HOST_GROUP writers
```

Note: See Section 5.2.6, “GROUP,” and Section 5.2.7, “HOST_GROUP,” for more information about defining groups.

5.5.3 EXCLUDE Example

```
#First Define the group "painters"
GROUP painters picasso mondrian klee
EXCLUDE spell GROUP painters
EXCLUDE spell USER bob
EXCLUDE spell INTERNET 123.123.123.*
```

This options file would:

- Prevent the users “picasso,” “mondrian,” and “klee” from using the feature “spell” on any machine on the network.
- Prevent the user “bob” from using the feature “spell” on any machine on the network.
- Prevent any user logged into a host with an IP address in the range 123.123.123.0 through 123.123.123.255 from using the feature “spell.”
- Allow any other user, as long as they are not on the excluded IP addresses, *and* they are not a member of the “painters” GROUP, *and* they are not “bob,” to use feature “spell” (by implication).

Note that “bob” could have been added to the group “painters.” However, “painters” might be used for some other purpose in the future so the license administrator chose to handle “bob” as a special case here. In this case, the two EXCLUDE statements concatenate to create a list of four users.

5.5.4 INCLUDE Example

```
INCLUDE paint USER picasso
INCLUDE paint USER mondrian
INCLUDE paint HOST bigbrush
```

This options file would:

- Allow the user “picasso” to use the feature “paint” on any machine on the network.
- Allow the user “mondrian” to use the feature “paint” on any machine on the network.
- Allow any user, as long as they are on the host “bigbrush,” to use feature “paint.”
- Deny access to the feature “paint” to anyone except “picasso,” “mondrian,” or anyone from the host “bigbrush” (by implication).

Chapter 6

License Administration Tools

FLEX*lm* provides utilities for the license administrator to help manage the licensing activities on the network. These utilities are:

- `lmborrow` — Supports license borrowing.
- `lmdiag` — Diagnoses license checkout problems.
- `lmdown` — Gracefully shuts down all license daemons (both `lmgrd` and all vendor daemons) on the license server node (or on all three nodes in the case of three-server redundant servers).
- `lmhostid` — Reports the hostid of a system.
- `lminstall` — Converts license files between different formats.
- `lmnewlog` — Moves existing report log information to a new file name and starts a new report log file with existing file name. Requires a v7.1+ vendor daemon.
- `lmpath` — Allows users direct control over license file settings. Requires v7.0+ applications.
- `lmremove` — Releases a hung license to the pool of free licenses.
- `lmreread` — Causes the license daemon to reread the license file and start any new vendor daemons. Requires v8.0 `lmreread`.
- `lmstat` — Displays the status of a license server.
- `lmswitch` — Controls debug log location and size. Requires v8.0+ vendor daemon.
- `lmswitchr` — Switches the report log to a new file name.
- `lmver` — Reports the FLEX*lm* version of a library or binary file.

This chapter also contains command-line syntax for `lmgrd`.

6.1 Running Administration Tools

All FLEX*lm* utility programs (except `lmgrd`) are packaged as a single executable called `lmutil`. `lmutil` can either be installed as the individual commands (either by creating links to the individual command names, or making copies of `lmutil` as the individual command names), or the commands can be run as `lmutil` *`command`*, for example, `lmutil lmstat`, or `lmutil lmdown`. On Windows systems, the `lmutil` *`command`* forms of the commands are available. There is also a graphical user interface for these commands on Windows—see Section 6.17, "License Administration Tools—LMTOOLS for Windows."

6.2 Universal lmutil Arguments

The following are valid arguments for most `lmutil` utilities:

`-c` *`license_file_path`*	Most `lmutil` utilities need to know the path to the license file. This can be specified with a `-c` *`license_file_path`* argument, or by setting the `LM_LICENSE_FILE` environment variable. Otherwise, the default location is used. Version 7.0+ utilities also honor all *`VENDOR`*`_LICENSE_FILE` environment variables. Some utilities can take more than one license file path in a license-file list separated by colons on UNIX and semi-colons on Windows.
`-v`	Prints the FLEX*lm* version of the utility.
`-verbose`	Prints longer description for all errors found. The output from the utilities may be harder to read with this option, but is useful for diagnostics. (v6.0+ only)

6.3 lmborrow

`lmborrow` supports borrowing of licenses with the BORROW attribute and must be run on the machine where licenses will or have been borrowed. It sets `LM_BORROW` in either the registry (Windows) or in `$HOME/.flexlmrc` (UNIX).

To initiate borrowing, the user can run `lmborrow` from the command line or through LMTOOLS:

```
lmborrow {vendor | all} enddate [time]
```

where:

vendor	The vendor daemon name that serves the licenses to be borrowed, or `all` specifies all vendor daemons in that license server.
enddate [*time*]	Date the license is to be returned in *dd-mmm-yyyy* format. *time* is optional and is specified in 24-hour format (`hh:mm`) in the client's local time. If *time* is unspecified, the checkout lasts until the end of the given end date.

For example:

```
lmborrow xyzd 20-aug-2001 13:00
```

To borrow licenses for the desired vendor, *on the same day and the same machine* that the user runs `lmborrow`, run the application(s) to check out the license(s). If you run the application(s) more than once that day, no duplicate licenses are borrowed. No licenses are borrowed if the application is run on a day different than the date borrowing was to be initiated.

To print information about features borrowed on a machine:

```
lmborrow -status
```

This is useful for finding out which features have been borrowed while the machine is disconnected from the network.

To clear the `LM_BORROW` setting in the registry or `$HOME/.flexlmrc`:

```
lmborrow -clear
```

Clearing the LM_BORROW setting will prevent any more licenses from being borrowed. A user might run `lmborrow -clear` after he has borrowed licenses for features that will be used offline if—before disconnecting from the network—he wants to run an application that checks out additional features served by *vendor* that should not be borrowed. Clearing LM_BORROW does *not* clear borrowing information for already-borrowed licenses.

6.4 lmdiag

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

Usage is:

```
lmdiag [-c license_file_list] [-n] [feature[:keyword=value]]
```

where:

`-c` *license_file_list*	Diagnose the specified file(s).
`-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.
keyword=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 "12345678." *keyword* 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 list. `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` will give you 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."

These extended 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.

See also Section B.2, "FLEXLM_DIAGNOSTICS."

6.5 lmdown

The `lmdown` utility allows for the graceful shutdown of all license daemons (both `lmgrd` and all vendor daemons) on all nodes.

Usage is:

```
lmdown -c license_file_list [-vendor vendor] [-q] [-all]
[-force]
```

where:

`-c` *`license_file_list`*	Use the specified license file(s). Note that specifying `-c` *`license_file_list`* is always recommended with `lmdown`
`-vendor` *`vendor`*	Shut down only this vendor daemon. `lmgrd` will always continue running if this option is specified. Requires v6.0 `lmdown` and `lmgrd` (the vendor daemon can be any version).
`-q`	Don't prompt or print a header. Otherwise `lmdown` asks "Are you sure? [y/n]: ."
`-all`	If multiple servers are specified, automatically shuts down all of them. Otherwise, only one is shutdown. `-q` is implied with `-all`. (v7.0+)

`-force`	If licenses are borrowed, `lmdown` will run only from the machine where the license server is running, and then only if the user adds `-force`. (All components v8.0+)

You may want to protect the execution of `lmdown`, since shutting down the servers causes users to lose their licenses. See the `-local`, `-2 -p`, or `-x` options in Section 6.6, "lmgrd," for details about securing access to `lmdown`.

If `lmdown` encounters more than one server (for example if `-c` specifies a directory with many `*.lic` files), a choice of license servers to shut down is presented.

Note: On UNIX, do *not* use `kill -9` to shut down the license servers. On Windows, if you must use the Task Manager to kill the FLEX*lm* service, be sure to end the `lmgrd` process first, then all the vendor daemon processes.

To stop and restart a single vendor daemon, use `lmdown -vendor` *`vendor`*, then use `lmreread -vendor` *`vendor`* to restart the vendor daemon.

When shutting down a three-server redundant license server, there is a one-minute delay before the servers shut down. `lmdown` will shut down all three license servers of a set of redundant license servers. If you need to shut down one of a set of redundant license servers (not recommended because you are left with two points of failure), you must kill both the `lmgrd` and vendor daemon processes on that license server machine.

See also Section 6.12, "lmreread."

6.6 lmgrd

`lmgrd` is the main daemon program for FLEX*lm*. When you invoke `lmgrd`, it looks for a license file which contains information about vendors and features. On UNIX systems, it is strongly recommended that `lmgrd` be run as a non-privileged user (not root).

Usage is:

```
lmgrd [-c license_file_list] [-l debug_log_path]
  [-2 -p] [-local] [-nfs_log] [-x lmdown] [-x lmremove] [-z ]
  [-v]
```

where:

`-c` *`license_file_list`*	Use the specified license file(s).
`-l` *`debug_log_path`*	Write debugging information to file *`debug_log`*. This option uses the letter `l`, not the numeral 1.
`-2 -p`	Restricts usage of `lmdown`, `lmreread`, and `lmremove` to a FLEX*lm* administrator who is by default root. If there a UNIX group called "lmadmin," then use is restricted to only members of that group. If root is not a member of this group, then root does not have permission to use any of the above utilities. If `-2 -p` is used when starting `lmgrd`, no user on Windows can shut down the license server with `lmdown`.
`-local`	Restricts the `lmdown` command to be run only from the same machine where `lmgrd` is running.
`-nfs_log`	It is not recommended to write a debug log to an NFS-mounted or PC network-mounted disk. If you choose to write to a mounted disk and the speed of the license server is too slow, you can use this flag to cache debug info before it is written out (when approximately 1kb of data is reached).

`-x lmdown`	Disable the `lmdown` command (no user can run `lmdown`). If `lmdown` is disabled, you will need to stop `lmgrd` via `kill` *`pid`* (UNIX) or stop the `lmgrd` and vendor daemon processes through the Windows Task Manager or Windows service. On UNIX, be sure the `kill` command does not have a `-9` argument. (v4.0+ `lmgrd`)
`-x lmremove`	Disable the `lmremove` command (no user can run `lmremove`). (v4.0+ `lmgrd`)
`-z`	Run in foreground. The default behavior is to run in the background. If `-l` *`debug_log_path`* is present, then no windows are used, but if no `-l` argument specified, separate windows are used for `lmgrd` and each vendor daemon.
`-v`	Prints `lmgrd` version number and copyright and exits.

6.6.1 Managing Debug Log Output

A license server always generates debug log output. Some of the debug log output describes events specific to `lmgrd` and some of the debug log output describes events specific to each vendor daemon. As `lmgrd` and its vendor daemons run for a period of time, the volume of this output can become quite large. As it gets older, the value of the debug log output decreases; therefore, it may be necessary for you to separate old debug log output from current output so you can either archive or delete the old output.

For performance, it is suggested that each debug log file be a on disk that is local to the machine that is running `lmgrd` and its vendor daemons. However, if the debug log file must be on a mounted disk and you find that the license server is too slow, you can start `lmgrd` with the `-nfs_log` option to improve performance.

See Appendix F, “The Debug Log File,” for a description of the debug log output format.

Capturing Debug Log Output for a License Server

By default, `lmgrd` and the vendor daemons it manages write debug log output to standard out. To put this debug log output in a file, you can either redirect the output of the license server to a file or start `lmgrd` with the `-l` *`debug_log_path`* option.

Capturing Debug Log Output for a Particular Vendor Daemon

The debug log output from different vendor daemons in the same license server can be written to their own files (v8.0+ vendor daemon). There are two ways to do this:

- Add the DEBUGLOG line to the options file for each vendor daemon. See Section 5.2.2, “DEBUGLOG,” for more details.
- Invoke `lmswitch` on the vendor daemon. See Section 6.14, “lmswitch,” for more details.

Even if you redirect the debug log output for every vendor daemon managed by a particular `lmgrd`, that `lmgrd` will still write its own debug log output to standard out.

Redirecting Debug Log Output for a Running Vendor Daemon

The debug log output for a particular vendor daemon can be redirected to a different file. There are two ways to do this:

- Change the DEBUGLOG line to the options file for the vendor daemon and reread its option file by invoking `lmreread`. See Section 5.2.2, “DEBUGLOG,” for more details.
- Invoke `lmswitch` on the vendor daemon. See Section 6.14, “lmswitch,” for more details.

Limiting Debug Log Output for a Vendor Daemon

By default, debug log output contains all events. To limit the events that are logged for a particular vendor daemon, add a NOLOG line to the options file of that vendor daemon. See Section 5.2.13, “NOLOG,” for more details. One of the reasons you may want to limit the events that are logged is to reduce the size of the debug log output.

6.6.2 Managing Report Log Output

Report log output is written by a vendor daemon; `lmgrd` does not write to a report log. However, a vendor daemon does not write report log output by default. Report log output is not human readable and is only used by the SAM*report* and FLEX*bill* products. Therefore, unless you are using either of these two GLOBE*trotter* products (or intend to use them in the future), there is no reason to enable report logging. As a vendor daemon runs for a period of time, report log output can become quite large. Therefore, it may be necessary to move report log output into different files over time, each file containing license activity over a particular period of time.

Because of a report log's format, standard file compression tools can significantly reduce the size of a report log file when it is no longer being written.

To avoid corruption and for performance, it is suggested that each vendor daemon write to its own report log file on a disk that is local to the machine that is running the vendor daemon.

ENABLING REPORT LOG OUTPUT FOR A VENDOR DAEMON

There are two ways to enable report logging for a particular vendor daemon either before or after starting the license server.

- Add the REPORTLOG line to the options file for that vendor daemon. See Section 5.2.14, "REPORTLOG," for more details.
- Invoke `lmswitchr` on the vendor daemon. See Section 6.15, "lmswitchr," for more details.
- Invoke `lmnewlog` on the vendor daemon. Requires a v7.1+ vendor daemon. See Section 6.9, "lmnewlog," for more details.

REDIRECTING REPORT LOG OUTPUT FOR A VENDOR DAEMON

The report log output for a particular vendor daemon can be moved into separate files, each file representing activity over a period of time. There are three ways in which to do this whether the vendor daemon is running or not:

- You can change the REPORTLOG line in the vendor daemon's options file and reread its option file by invoking `lmreread` (v8.0+ vendor daemon).
- Invoke `lmswitchr` on the vendor daemon. See Section 6.15, "lmswitchr," for more details.
- Invoke `lmnewlog` on the vendor daemon. Requires a v7.1+ vendor daemon. See Section 6.9, "lmnewlog," for more details.

6.7 lmhostid

The `lmhostid` utility reports the hostid of a any machine whose platform is supported by FLEX*lm*. The default hostid type is displayed for a platform, unless an optional hostid type is specified and supported by that platform.

Usage is:

```
lmhostid [-n] [type]
```

where *type* is one of:

`[-internet]` (optional on all platforms)

`[-vsn] [-flexid]`

`-n`	No header is printed, only the hostid is printed.
`-internet`	IP address in `###.###.###.###` format.
`-vsn`	Volume Serial Number of the Windows `C:\` drive.
`-flexid`	GLOBE*trotter* dongle-based hostid. (Windows)

The output of this command looks as follows:

```
lmhostid - Copyright (c) 1989, 1997 Globetrotter Software, Inc.
The FLEXlm hostid of this machine is "69021c89"
```

See also Appendix A, “Hostids for FLEXlm-Supported Machines.”

6.8 lminstall

Introduced in v6.0, `lminstall` is designed primarily for typing in decimal format licenses to generate a readable format license file.

Usage is:

```
lminstall [-i in_lic_file ] [-maxlen n] [-e err_file]\
  [-o out_lic_file] \
  [-overfmt {2 | 3 | 4 | 5 | 5.1 | 6 | 7 | 7.1 | 8}] [-odecimal]
```

Normally, to convert from decimal to readable format, `lminstall` is used with no arguments; you are prompted for the name of the output license file. The default file name is today’s date in *yyyymmdd*`.lic` format. The file should be moved to the application’s default license file directory, if specified

by the software vendor. Otherwise, use the LM_LICENSE_FILE or *VENDOR*_LICENSE_FILE environment variables to specify the directory where the *.lic files are located.

Decimal format input is verified by a checksum of each line.

To finish entering, type q on a line by itself or enter two blank lines.

When an input file is specified with no output file specified, output goes to stdout; if neither input nor output file is specified, lminstall assumes that input will come from stdin and prompts the user for an output file name.

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

To convert from readable to decimal:

```
lminstall -i in_lic_file -o out_lic_file -odecimal
```

To convert to FLEX*lm* v5.1 format:

```
lminstall -i in_lic_file -o out_lic_file -overfmt 5.1
```

To enforce a maximum line length of, for example, 50 characters:

```
lminstall -maxlen 50
```

Conversion errors are reported as necessary and can be written to a file by specifying -e *err_file*. lminstall has a limit of 1000 lines of input.

6.9 lmnewlog

The lmnewlog utility switches the report log file by moving the existing report log information to a new file, then starting a new report log with the original report log file name. If you rotate report logs with lmnewlog instead of lmswitchr, you will not have to change the file name in the REPORTLOG line of the vendor's options file. Requires a v7.1+ vendor daemon.

Usage is:

```
lmnewlog [-c license_file_list] feature renamed_report_log
```

or:

```
lmnewlog [-c license_file_list] vendor renamed_report_log
```

where:

-c *license_file_list*	Use the specified license file(s).
feature	Any feature in this license file.

vendor	Vendor daemon in this license file.
renamed_report_log	New file path where existing report log information is to be moved.

6.10 lmpath

The `lmpath` utility allows direct control over FLEX*lm* license path settings. It can be used to add to, override, or get the current license path settings. Requires a v7.0+ application.

Note: `lmpath` works by setting the FLEX*lm* registry entry on Windows or `$HOME/.flexlmrc` on UNIX.

To add a license file or license-file list to the current license-file list for one or all vendor daemons, use:

```
lmpath -add {vendor | all} license_file_list
```

To override the existing license-file list for one or all vendor daemons, use:

```
lmpath -override {vendor | all} license_file_list
```

where *vendor* is the vendor daemon name (or type `all` for all vendors), and *license_file_list* is a colon-separated list on UNIX and a semi-colon-separated list on Windows.

If you try to add a path that is in the current license-file list, no changes will be made to the list.

To display the current license path settings, use:

```
lmpath -status
```

You will see, for example:

```
lmpath - Copyright (C) 1989-2001 Globetrotter Software, Inc.
Known Vendors:
_____________
demo:   ./counted.lic:./uncounted.lic
_____________
Other Vendors:
_____________
/usr/local/flexlm/licenses/license.lic
```

Note that where the path is set to a directory, all the `*.lic` files are listed separately.

6.11 lmremove

`lmremove` is needed only when a client node crashes, because that's the only condition where a license is not automatically freed immediately (v6.0+ applications will usually free these licenses automatically within three minutes to two hours). The `lmremove` utility allows you to remove a single user's license for a specified feature. If the application is active, it will re-checkout the license shortly after it is freed by `lmremove`.

Usage is:

```
lmremove [-c license_file_list] feature user user_host display
```

or

```
lmremove [-c license_file_list] -h feature server_host port handle
```

where:

`-c license_file_list`	Specify license file(s).
feature	Name of the feature checked out by the user.
user	Name of the user whose license you are removing, as reported by `lmstat -a`.
user_host	Name of the host the user is logged into, as reported by `lmstat -a`.
display	Name of the display where the user is working, as reported by `lmstat -a`.
server_host	Name of the host on which the license server is running.
port	Port number where the license server is running, as reported by `lmstat -a`.
handle	License handle, as reported by `lmstat -a`.

The *user*, *user_host*, *display*, *server_host, port*, and *handle* information must be obtained from the output of `lmstat -a`.

`lmremove` removes all instances of *user* on *user_host* and *display* from usage of *feature*. If the optional `-c` *license_file_list* is specified, the indicated file(s) is used as the license file. You should protect the execution of `lmremove`, because removing a user's license can be disruptive. See the `-x` or `-2 -p` options in Section 6.6, "lmgrd," for details about securing access to `lmremove`.

The `-h` variation uses the *server_host*, *port*, and license *handle*, as reported by `lmstat -a`. Consider this example `lmstat -a` output:

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

In this example, the user is "joe," the user host is "nirvana," the display is "/dev/typp5," the server host is "cloud9," the port is "7654," and the license handle is "102."

To remove this license, issue one of the following commands:

```
lmremove f1 joe nirvana /dev/ttyp5
```

or

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

When removing by handle, if licenses are grouped as duplicates, all duplicate licenses will also be removed. If license lingering is set and `lmremove` is used to reclaim the license, `lmremove` will start, but not override, the license's linger time.

6.12 lmreread

The `lmreread` utility causes the license daemon to reread the license file and start any new vendor daemons that have been added. In addition, all running daemons will be signaled to reread the license file and their end-user options files for changes in feature licensing information or option settings.

A v8.0+ vendor daemon will reread the path to its option file and will reread its options file as a result of `lmreread`. `lmreread` will also allow server machine host names to be reread, but still cannot be used to change server port numbers.

If the optional vendor daemon name is specified, only the named daemon will reread the license file and its end-user options file (in this case, `lmgrd` will not reread the license file).

Usage is:

```
lmreread [-c license_file_list] [-vendor vendor] [-all]
```

where:

`-c license_file_list`	Use the specified license file(s).
`-vendor vendor`	Only this one vendor daemon should reread the license file. `lmgrd` will restart the vendor daemon if necessary. Requires v6.0+ `lmreread` and `lmgrd` (the vendor daemon can be any version).
`-all`	If more than one `lmgrd` is specified, instructs all `lmgrd`s to reread.

You may want to protect the execution of `lmreread`. See the `-2 -p` and `-x` options in Section 6.6, “lmgrd,” for details about securing access to `lmreread`.

To stop and restart a single vendor daemon, use `lmdown -vendor vendor`, then use `lmreread -vendor vendor`, which restarts the vendor daemon.

Note: If you use the `-c license_file_list` option, the license file(s) specified will be read by `lmreread`, not by `lmgrd`; `lmgrd` rereads the file it read originally.

6.13 lmstat

The `lmstat` utility helps you monitor the status of all network licensing activities, including:

- Daemons that are running
- Users of individual features
- Users of features served by a specific vendor daemon
- BORROW licenses borrowed

`lmstat` prints information that it receives from the license server, therefore it will not return any information about users of node-locked uncounted or DEMO licenses, unless the server’s license file includes the node-locked

licenses and the client is not reading the license file (via `@host`, `port@host` or USE_SERVER). Queued users and licenses shared due to duplicate grouping are also not returned by `lmstat`.

Usage is:

```
lmstat [-a] [-A] [-c license_file_list] [-f [feature]]
  [-S [vendor]]
```

where:

`-a`	Display all information about.
`-A`	List all active licenses.
`-c license_file_list`	Use the specified license file(s).
`-f [feature]`	List users of *feature*.
`-S [vendor]`	List all users of *vendor*'s features.

`lmremove` requires the output of `lmstat -a`. The output of `lmstat -a` looks similar to:

```
License server status: 27000@myhost1
License file(s) on myhost:
install_dir/flexlm/v8.1/sun4_u5/counted.lic:
myhost: license server UP (MASTER) v8.1
Vendor daemon status (on myhost1):

demo: UP v8.1
Feature usage info:
Users of f1:  (Total of 4 licenses available)
  "f1" v1.0, vendor: demo
  floating license
    daniel myhost2 19.16.18.26 (v1.0) (myhost1/27000 102), start Fri
         5/3 7:29
```

where:

daniel	*user*	User name.
myhost2	*user_host*	Host where user is running.
19.16.18.26	*display*	Display where user is running.

v1.0	*version*	Version of feature.
myhost1	*server_host*	Host where license server is running.
27000	*port*	Port on *server_host* where license server is running.
102	*handle*	License handle.
start Fri 5/3 7:29	*checkout_time*	Time that this license was checked out.

Note: `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.

6.14 lmswitch

The `lmswitch` utility switches the debug log file written by a particular vendor daemon by closing the existing debug log for that vendor daemon and starting a new debug log for that vendor daemon with a new file name. It will also start a new debug log file written by that vendor daemon if one does not already exist. Requires a v8.0+ vendor daemon.

Usage is:

```
lmswitch [-c license_file_list] vendor new_debug_log
```

where:

`-c` *license_file_list*	Use the specified license file(s).
vendor	Vendor daemon in this license file.
new_debug_log	Path to new debug log file.

By default, debug log output from `lmgrd` and all vendor daemons started by that `lmgrd` get written into the same debug file. `lmswitch` allows companies to keep separate log files for different vendors and control the size of their debug log file.

If debug log output is not already directed to a separate file for this vendor daemon, `lmswitch` tells the vendor daemon to start writing its debug log output to a file, *`new_debug_log`*. If this vendor daemon is already writing to its own debug log, `lmswitch` tells the vendor daemon to close its current debug log file and start writing its debug log output to *`new_debug_log`*. The effect of `lmswitch` continues only until the vendor daemon is shut down or (as of v8.0) its options file is reread via `lmreread`. When the vendor daemon is restarted or its options file is reread, it will look for a DEBUGLOG line in the options file to determine whether or not to write its debug log output into its own file and, if so, what file to write.

SEE ALSO:

- Section 5.2.2, “DEBUGLOG”

6.15 lmswitchr

The `lmswitchr` utility switches the report log file by closing the existing report log and starting a new report log with a new file name. It will also start a new report log file if one does not already exist.

Usage is:

```
lmswitchr [-c license_file_list] feature new_report_log
```

or with v5.0+ vendor daemon:

```
lmswitchr [-c license_file_list] vendor new_report_log
```

where:

`-c` *`license_file_list`*	Use the specified license file(s).
`feature`	Any feature in this license file.
`vendor`	Vendor daemon in this license file.
`new_report_log`	Path to new report log file.

If report logging is not enabled for the vendor daemon, `lmswitchr` tells it to start writing its report log output to *`new_report_log`*. If report logging is already enabled for the vendor daemon, `lmswitchr` tells the vendor daemon to close its report log file and start writing its new report log output to *`new_report_log`*. The effect of `lmswitchr` continues only until the vendor daemon is shut down or (as of v8.0), its options file is reread via `lmreread`. When the vendor daemon is restarted or its options file is reread, it will look for a REPORTLOG line in the options file to determine whether or not to write report log output to a file and, if so, what file to write.

SEE ALSO:

- Section 6.9, “lmnewlog”

6.16 lmver

The `lmver` utility reports the FLEX*lm* version of a library or binary file.

Usage is:

```
lmver filename
```

where *`filename`* is the name of an executable file built with FLEX*lm*.

For example if you have an application called “spell,” type:

```
lmver spell
```

Alternatively, on UNIX systems, you can use the following commands to get the FLEX*lm* version of a binary:

```
strings filename | grep Copy
```

6.17 License Administration Tools—LMTOOLS for Windows

For the 32-bit Windows platforms, an LMTOOLS program is provided.

With LMTOOLS, you can start, stop and configure FLEX*lm* license servers, get system information, including hostids, get server status, and more.

LMTOOLS has two modes in which to configure a license server:

- Configuration using a license file
- Configuration using services

6.17.1 Configuration Using License File

Operations are performed on a particular license file. In this mode, you cannot start the `lmgrd` process, but you can do everything else. In the first tab, you need to select a license file.

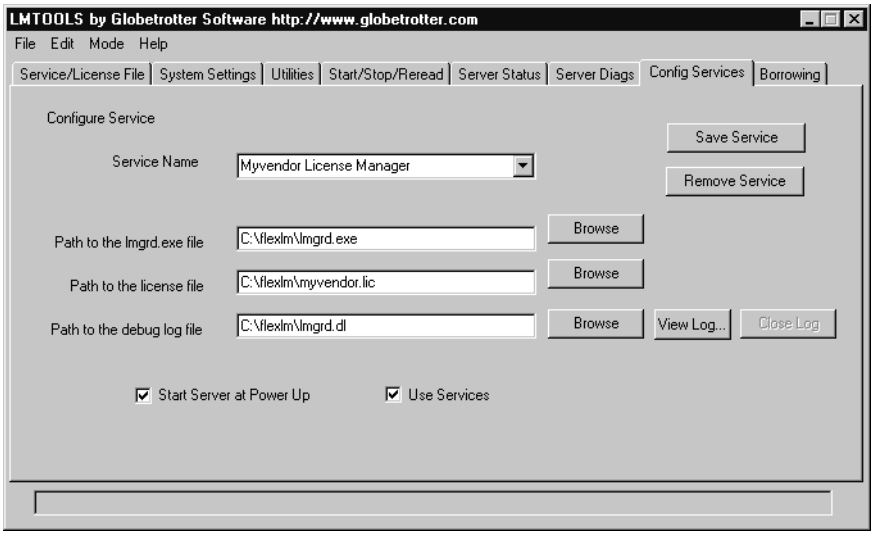
6.17.2 Configuration Using Services

Operations are performed on a service, which allows starting `lmgrd` processes. On NT/2000, you configure a Windows Service. On Windows 95/98, you configure a FLEX*lm* service that behaves similarly to a Windows service.

To configure a license server as a service, you must have Administrator privileges:

1. Run LMTOOLS.
2. Click the Configuration using Services radio button, then click the Config Services tab.
3. In the Service Name, type the name of the service that you want to define, for example, Myvendor License Manager.
4. In the Path to the lmgrd.exe file field, enter or browse to `lmgrd.exe` for this license server.
5. In the Path to the license file field, enter or browse to the license file for this license server.
6. In the Path to the debug log file, enter or browse to the debug log file that this license server writes.
7. To make this license server a Windows service, check the Use Services check box (otherwise, it will be a FLEX*lm* service).

8. If the license server is a Windows service, it will start on boot if the Start Server at Power Up check box is checked.



9. To save the new Myvendor License Manager service, click the Save Service button. From now on, when this machine is rebooted, this license server will start automatically as a Windows service.

Chapter 7

Mobile Licensing

End users often want to use applications on computers that do not have a continuous connection to a FLEX*lm* license server. These situations include:

- Working on a laptop
- Using a computer both at work and at home
- Working from several different computers not connected to a license server

FLEX*lm* supports licenses that allow one of several kinds of mobile licensing:

- Node-locked to a laptop
- Node-locked to a FLEX*id* (Windows only)
- Node-locked to a FLEX*id* with FLOAT_OK keyword (Windows only)
- License borrowing with BORROW keyword

7.1 Node-Locked to a Laptop Computer

If a license is to be used exclusively on one laptop computer, that license can simply be node-locked to an address associated with that computer. The license file would reside on the laptop computer.

7.2 Node-locked to a FLEX*id* (Windows Only)

If a license is to be moved between different Windows machines, it can be node-locked to a FLEX*id* (a dongle that connects to a parallel or USB port). This license can be moved between machines by installing a copy of the license file on each machine and moving the FLEX*id* from one machine to another.

7.3 Node-Locked to a FLEX*id* with FLOAT_OK (Windows Only)

This method of license mobility has an advantage over simply using a license node-locked to a FLEX*id*, because the FLEX*id* can be attached to a license server machine and its license can float on the network. Licenses with a FLOAT_OK keyword that are node-locked to a FLEX*id* are supported only where both the client application and the license server are running on Windows.

A vendor issues a license file with a FEATURE line node-locked to a FLEX*id* and containing the FLOAT_OK keyword and a FLEX*id* for that FEATURE line. One FEATURE line containing the FLOAT_OK keyword and one FLEX*id* is needed for each instance of a license that can be mobile. When the FLEX*id* is attached to a license server, the license floats on the network. When the FLEX*id* is removed from the license server, the license is available only on the standalone computer.

This method supports parallel or USB FLEX*id*s. Because it is simpler to attach multiple USB dongles to a computer, USB FLEX*id*s may be preferable. All FLEX*lm* components must be v8.0 or higher.

7.3.1 Initiating FLEX*id* with FLOAT_OK Mobility

A vendor issues the end user a FLEX*id*, a FLEX*id* driver installer, and a license file that contains a FEATURE line node-locked to that FLEX*id* containing the FLOAT_OK keyword. A end user then:

1. Installs the license file on the license server machine
2. Attaches all of the FLEX*id*s to the license server machine
3. Installs the FLEX*id* driver on the license server machine
4. Starts the license server or rereads the license file

While the FLEX*id*s are attached to the license server machine, the node-locked licenses associated with them float on the network. Each of the FLOAT_OK uncounted node-locked FEATURE lines has a count of *one* while it is available on the network.

To transfer a license from the pool of floating licenses to a disconnected computer, the end user:

1. Copies the license file containing the FLOAT_OK node-locked FEATURE line from the license file on the license server machine to a license file on the client computer in the location where the licensed application expects to find its license file.
2. Moves the FLEX*id* matching the node-locked FEATURE line from the license server machine to the client computer. When the FLEX*id* is removed from the license server machine, this license is unavailable on the network.
3. Installs the FLEX*id* drivers on the client computer, if they are not already installed.
4. Disconnects the client computer from the network. Now the license is available on the computer with the FLEX*id*, even though that computer is disconnected from the network.

7.3.2 Returning a FLEX*id* with FLOAT_OK License

To return the license to the license server machine so it can float on the network again, the end user:

1. Removes the FLEX*id* from the client machine and replaces it on the license server machine.
2. Rereads the license file for the license server that serves the floating version of the license by running `lmreread`. When the FLEX*id* is returned to the license server machine, the FLOAT_OK license will not float on the network again until `lmreread` is run.

7.3.3 FLEX*id* with FLOAT_OK Example

The following is a sample license file issued to a customer site. It is shipped with two FLEX*id*s: `FLEXID=7-b28520b9` and `FLEXID=7-b2857678`.

```
SERVER myhost ANY
VENDOR xyzd
FEATURE f1 xyzd 1.0 permanent uncounted FLOAT_OK \
  HOSTID=FLEXID=7-b28520b9 SIGN=123456789012
FEATURE f1 xyzd 1.0 permanent uncounted FLOAT_OK \
  HOSTID=FLEXID=7-b2857678 SIGN=ABCDEF123456
```

The customer installs the license file and the two FLEX*id*s on the license server machine. When attached to the license server machine, each uncounted FLOAT_OK license floats on the network and allows a single use. Therefore, up to two users can use “f1” on the customer’s network, except on the license server machine itself, where the license use is disallowed.

If an end user wants to work at home, the end user installs a license file that contains the FEATURE line node-locked to `FLEXID=7-b28520b9` (this only needs to be done once), transfers the FLEX*id* `FLEXID=7-b28520b9` from the license server machine to the client computer, and installs the FLEX*id* driver on the client computer (this also only needs to be done once). The end user disconnects the client computer from the network and can use the transferred FLOAT_OK license on the client computer. The license server allows only the single remaining FLOAT_OK license to float on the network.

After returning the FLEX*id* to the license server machine, the end user (or the system administrator) runs `lmreread` so the returned license can float again.

7.4 License Borrowing with BORROW

If a license is to be used on a computer that is intermittently connected to a license server, that license can be issued as a floating license with a BORROW keyword. A BORROW license can be borrowed from a license server via a special checkout and used later to run an application on a computer that is no longer connected to the license server. License borrowing must be enabled by a vendor before an end user can borrow licenses and all FLEX*lm* components must be built with FLEX*lm* v8.0+.

With license borrowing, a vendor issues a floating license with a FEATURE line that contains the BORROW keyword. An end user specifies the expiration date a borrowed license will be returned and runs the application while connected to the network which writes borrowing information on the client computer. The license server keeps the borrowed license checked out. The client application automatically uses the local borrowing data to do checkouts during the borrow period. Borrowed licenses cannot be returned before the borrow period expires. When the borrow period ends, the local borrowing data no longer authorizes checkouts and the license server returns the borrowed license to the pool of available licenses. No clock synchronization is required between the license server machine and the client machine.

7.4.1 Initiating License Borrowing

If a vendor has enabled license borrowing by issuing a license file that contains a FEATURE line with the BORROW keyword, an end user can initiate license borrowing in one of three ways:

- Using the borrowing interface in application, if provided in the application
- Running the `lmborrow` utility to set `LM_BORROW`
- Setting the `LM_BORROW` environment variable directly

APPLICATION INTERFACE

The user can initiate license borrowing this way only if the application provides a borrowing interface.

RUNNING THE LMBORROW UTILITY

`lmborrow` is one of the `lmutil`/LMTOOLS utilities. To initiate borrowing, the user can run `lmborrow` from the command line or through LMTOOLS:

```
lmborrow {vendor|all} enddate [time]
```

where *`vendor`* is the vendor daemon that serves the licenses to be borrowed, or `all` specifies all vendor daemons in the license server. *`enddate`* is the date the license is to be returned in *`dd-mmm-yyyy`* format. *`time`* is optional and is specified in 24-hour format (*`hh`*`:`*`mm`*) in the client's local time. If *`time`* is unspecified, the checkout lasts until the end of the given end date.

For example:

```
lmborrow xyzd 20-aug-2001 13:00
```

SETTING THE LM_BORROW ENVIRONMENT VARIABLE DIRECTLY

The `lmborrow` utility is a user interface to set `LM_BORROW` in either the registry (Windows) or in `$HOME/.flexlmrc` (UNIX). `LM_BORROW` can also be set directly as an environment variable:

```
today:{vendor|all}:enddate[:time]
```

where:

`today`	Today's date in *`dd-mmm-yyyy`* format. Any checkouts done on this date will create local borrow information. If a checkout is done on a different date than this date, no local borrowing information will be created.

vendor	Vendor daemon that serves the licenses to be borrowed, or `all` specifies all vendor daemons in the license server.
enddate	Date the license is to be returned in *dd-mmm-yyyy* format.
time	Optional. *time* is specified in 24-hour format (*hh*:*mm*) in the client's local time. If *time* is unspecified, the checkout lasts until the end of the given end date.

For example:

```
LM_BORROW=15-aug-2001:xyzd:20-aug-2001:13:00
```

In this example, one or more licenses served by the `xyzd` vendor daemon are being borrowed on August 15, 2001, and will be returned at 1 pm on August 20, 2001.

BORROWING A LICENSE

To borrow a license for a desired feature, *on the same day and the same machine* that the end user runs `lmborrow` or sets `LM_BORROW` (and while still connected to the network), the end user runs the application to check out and borrow the license. If the end user runs the application more than once that day, no duplicate license is borrowed. No license is borrowed if the application is run on a day different than the date borrowing was set to be initiated.

For example, say that today you want to borrow a license for the PageWizard feature for a week. The PageWizard feature is served by the `xyzd` vendor daemon. Today, while you are connected to the network, run `lmborrow` or set `LM_BORROW` directly. For example:

```
lmborrow xyzd enddate
```

Today, after you run `lmborrow`, while you are connected to the network, run the application that checks out a license for the PageWizard feature. After the license is checked out, you can close the application and disconnect your machine from the network. The license that you just checked out will stay checked out from the license server until the borrow period expires—that license can now be used on your disconnected machine until the borrow period expires.

CHECKING BORROW STATUS

You can check the status of borrowed licenses on your computer by running `lmborrow -status`.

ENDING BORROWING

Once you have borrowed all the licenses that you need, you can prevent licenses for any additional features to be borrowed by running `lmborrow -clear`. `lmborrow -clear` does *not* clear the local information about licenses you have already borrowed. These licenses cannot be returned before the end of the borrow period. Once checked out, they remain checked out for the full borrow period. The borrow period cannot be renewed until the period has expired.

7.4.2 Support for License Borrowing

See the following sections for more information about the utilities and end-user options that support license borrowing:

- Section 6.3, “lmborrow”
- Section 6.5, “lmdown”
- Section 6.13, “lmstat”
- Section 5.2.1, “BORROW_LOWWATER”
- Section 5.2.4, “EXCLUDE_BORROW”
- Section 5.2.9, “INCLUDE_BORROW”

Appendix A

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 hostid, whereas all DEC machines do not. For this reason, the ethernet address is used on some machine architectures as the hostid. An ethernet address is a 6-byte quantity, with each byte specified as two hexadecimal digits. Specify all twelve hex digits when using an ethernet address as a hostid. For example, if the ethernet address is "8:0:20:0:5:ac," you would specify "0800200005ac" as the hostid.

A.1 Hostid Formats

Numeric, 32-bit hostids are normally used in hexadecimal format. On some systems, including HP and SGI, the system command returns the number in decimal format. Since v3.0 of FLEX*lm*, a "#" before the hostid indicates to FLEX*lm* that this is a decimal number. For example, if the HP `uname -i` command returns "2005771344," FLEX*lm* will accept "#2005771344." Or it can be converted to hexadecimal. On UNIX systems, you can convert to hex with the following script:

```
echo 2005771344 16o p | dc
778DA450
```

A.2 Expected FLEX*lm* Hostids

The program `lmhostid` will print the exact hostid that FLEX*lm* expects to use on any given machine. The following table lists alternate methods to obtain the required hostid for each machine architecture. FLEX*lm* also supports a group of special hostids and vendor-defined hostids.

Hardware Platform	Hostid	Type this command on the license server:	Example
AIX (RS/6000, PPC)	32-bit hostid	`uname -m` (returns 000276513100), then remove last two digits, and use remaining last eight digits	02765131
DEC Alpha	ethernet address	`netstat -i`	080020005532
HP	32-bit hostid	`uname -i` and convert to hex, or prepend with #	778DA450 or #2005771344
	ethernet address	`lanscan` (station address without leading “0x”)	0000F0050185
Linux	ethernet address	`/sbin/ifconfig eth0` and remove colons from HWaddr 00:40:05:16:E5:25	00400516E525
SCO	Hostid String	`uname -x` (Serial is SCO00354), then prefix with “ID_STRING=”	ID_STRING=SCO00354
SGI	32-bit hostid	`/etc/sysinfo -s`, convert to hex, or prefix #	69064C3C or #1762020412
SUN	32-bit hostid	`hostid`	170a3472

Hardware Platform	Hostid	Type this command on the license server:	Example
Windows	ethernet address	`lmutil lmhostid`	00B0A9DF9A32
	Disk serial number	DIR C: (look for "Volume Serial Number is", and remove "-")	DISK_SERIAL_NUM= 3e2e17fd
	FLEX*id* parallel or USB port hardware key (dongle)	`lmhostid -flexid`	FLEXID=7-b28520b9

	Pentium III+ CPU, v7.0d+ only. Use BIOS Setup to enable.	`lmhostid -cpu`
		`lmhostid -cpu96`
		(The 32-bit version is the last nine characters from the full id.)
			9077-5D77-0002-57C8-95D2-1D3D (96-bit)
			
			95D2-1D3D (32-bit)

A.3 Special FLEX*lm* Hostids

FLEX*lm* contains a number of "special" hostid types which apply to all platforms. These hostid types can be used on either a SERVER line or a FEATURE line, wherever a hostid is required. These are:

ANY	Locks the software to any node (i.e., does not lock anything).
DEMO	Similar to ANY, but only for use with uncounted FEATURE lines.
DISK_SERIAL_NUM=*SN*	Locks the software to a PC with C drive serial number *SN*. (Windows only).
DISPLAY=*display*	Locks the software to display *display*.

FLEXID=*id_string*	Locks the software to a PC with a FLEX*id* dongle with a serial number *id_string*. FLEX*id*s are made available by your vendor. Your vendor can also provide you with an installer that will install drivers for all FLEX*id*s. (Windows only).
HOSTNAME=*host*	Locks the software to computer host name *host*.
ID=*n*	Functionally equivalent to the "ANY" hostid—it will run on any node. The difference is that the license is unique and can be used to identify the customer. This hostid can be used to lock the license server (on the SERVER line) or the client (on the FEATURE/INCREMENT line). The number can have dashes included for readability—the dashes are ignored. Examples: ID=12345678 is the same as ID=1234-5678 is the same as ID=1-2-3-4-5-6-7-8
ID_STRING=*string*	Used on SCO systems for hostid.
INTERNET= *###.###.###.###*	Locks the software to an Internet IP address, or group of IP addresses. Wildcards are allowed. For example, 198.156.*.* means any host with a matching internet IP address. The main use is to limit usage access by subnet, implying geographic area. For this purpose, it would be used on the FEATURE/INCREMENT line, as a hostid lock.
USER=*user*	Locks the software to user name *user*.

EXAMPLES

```
FEATURE f1 demo 1.0 1-jan-2005 uncounted \
     HOSTID="HOSTNAME=globes HOSTNAME=mars" SIGN=AB28E0011DA1
```

or

```
FEATURE f1 demo 1.0 1-jan-2005 uncounted HOSTID=USER=joe \
     SIGN=EB78201163B0
```

Appendix B

Troubleshooting Guide

This appendix documents areas of FLEX*lm* that have given customers difficulty in the past.

B.1 General Debugging Hints

The following are tips for debugging:

- When you start the license server (`lmgrd`) be sure that you direct the output into a log file where you can examine it. The log file often contains useful information. You should examine it when you have a problem, and be prepared to answer questions about it when you talk to a support person.
- If the license server appears to have started correctly (which you should be able to determine from the log file), try running `lmstat -a` and `lmdiag` to see if that program has the same problem as your application.
- If your application is FLEX*lm* v4.1 or later (v5 or later on Windows), you can use the `FLEXLM_DIAGNOSTICS` environment variable. Set `FLEXLM_DIAGNOSTICS` to 1, 2, or 3. A setting of 3 gives more information than 2, 2 gives more information than 1 (in particular, the feature name that was denied).
- When you talk to a support person, you should be prepared to answer the following questions:
- What kind of machine is your license server running on? What version of the operating system? What machine and operating system is the application running on?
- What version of FLEX*lm* does the program use? Use the `lmver` script, or, on UNIX, execute the following command on your `lmgrd`, vendor daemon, and application:

  ```
  strings binary_name | grep Copy
  ```

  Alternatively, `lmgrd -v` gives the `lmgrd` version, and this works with the vendor daemon also.

- What error or warning messages appear in the log file? Did the server start correctly? Look for a message such as:

  ```
  server xyz started for: feature1 feature2.
  ```

- What is the output from running `lmstat -a`?
- Are you running other products which are also licensed by FLEX*lm*? Are you using a combined license file or separate license files?
- Are you using a three-server redundant license server (multiple SERVER lines in your license file)?

B.2 FLEXLM_DIAGNOSTICS

Note: Available only with applications using FLEX*lm* v4.1 or higher for UNIX, and v5.0 or higher with Windows. Also, applications may choose not to provide this functionality.

`FLEXLM_DIAGNOSTICS` is an environment variable that will cause the application to produce diagnostic information when a checkout is denied. The format of the diagnostic information may change over time.

To set `FLEXLM_DIAGNOSTICS`, on UNIX:

```
(csh): setenv FLEXLM_DIAGNOSTICS 1
(sh): FLEXLM_DIAGNOSTICS=1; export FLEXLM_DIAGNOSTICS
```

On Windows 3.1 and 95, add the following line to `C:\autoexec.bat`:

```
SET FLEXLM_DIAGNOSTICS=1
```

On NT, use the System Control Panel applet to change the global environment, adding `FLEXLM_DIAGNOSTICS` to 1.

On UNIX, the diagnostic output goes to stderr.

On Windows, if the application is v5.11 or higher, the output is a file in the current directory called `flex`*`pid`*`.log`, where *`pid`* is the application's process ID. If the application is v5.0, the output file is called `flex_err.log`.

B.2.1 Level 1 Content

If `FLEXLM_DIAGNOSTICS` is set to 1, then the standard FLEX*lm* error message will be presented, plus a complete list of license files that the application tried to use. For example:

```
setenv FLEXLM_DIAGNOSTICS 1
FLEXlm checkout error: Cannot find license file (-1,73:2) No such file
or directory
license file(s):  /usr/myproduct/licenses/testing.lic license.lic
```

B.2.2 Level 2 Content

If FLEXLM_DIAGNOSTICS is set to 2, then, in addition to level 1 output, the checkout arguments are presented. For example:

```
setenv FLEXLM_DIAGNOSTICS 2
FLEXlm checkout error: No such feature exists (-5,116:2) No such file or
directory
license file(s):  /usr/myproduct/licenses/testing.lic license.lic
lm_checkout("f1", 1.0, 1, 0x0, ..., 0x4000)
```

Note that the error message actually contains two separate problems, which both occurred during the checkout:

- There's no such feature in the license it did find
- It was unable to find the other license file, which is what produces the message "No such file or directory"

Following is a description of the arguments to lm_checkout()

`lm_checkout(`*`feature`*`, `*`version`*`, `*`num_lic`*`, `*`queue_flag`*`, ..., `*`dupgroup_mask`*`)`

where:

feature	The requested feature.
version	The requested version. The license file must contain a version >= the requested version.
num_lic	Number of licenses requested. Usually 1.
queue_flag	If 0, no queueing If 1, queue for license ("blocking" queue) If 2, queue for licenses, but return to application ("non-blocking" queue)
dupgroup_mask	Indicates duplicate grouping, also called license sharing. User, host, and display are as shown by `lmstat -a`.

B.2.3 Level 3 Content (FLEX*lm* v6.0+ only)

If FLEXLM_DIAGNOSTICS is set to 3, then, in addition to level 1 and 2 output, if a checkout is successful, information is printed explaining how the license was granted:

```
setenv FLEXLM_DIAGNOSTICS 3
app
Checkout succeeded: f0/14263EAEA8E0
License file: ./servtest.lic
No server used
app2
Checkout succeeded: f1/BC64A7B120AE
License file: @localhost
License Server: @localhost
app3
Checkout succeeded: f1/BC64A7B120AE
License file: servtest.lic
License Server: @speedy
```

Note that the feature name and license key are printed, along with the license file location (or host name if `@host` were used) and host name of the server, where applicable.

B.3 FLEX*lm* Troubleshooting List

B.3.1 Problem Description Format

Each problem is presented in three parts:

Symptom:	*A description of the problem.*
Cause:	A discussion of what causes the problem described in the “Symptom” section.
Solution:	Instructions on how to solve the problem.

You can scan through the list of problems to find any which appear to relate to your concerns. In order to solve your problem, you may have to use all or some of the solutions listed here.

B.3.2 Hostid Problems

Symptom: *When I run the license manager on my machine, it tells me it is the wrong hostid.*

Cause: The vendor daemon checks the hostid listed on the SERVER line in the license file; if it does not match the hostid of the machine it is running on, this message will be printed. Possible causes include: 1) you are trying to run the license server on a different machine from the machine the file was made for; 2) the hostid of the machine you are running on changed (for example, the dongle hostid was moved (Windows) or the CPU board was replaced); 3) the hostid in the license file was modified.

Solution: Verify that the hostid of the machine on which the vendor daemon (or node locked client program) is being run matches the hostid specified in the license file (on the SERVER line for the vendor or on the FEATURE line for a node-locked client). You can run the `lmhostid` program to see what FLEX*lm* thinks the hostid is. You may not modify the hostid in the license file. If the hostid of your server machine changes, you will have to get a new license file from your software vendor.

B.3.3 Connection Problems

Symptom: *The application program (or `lmstat`) can't connect to the server to check out a license.*

Cause: The FLEX*lm* routines in the application are unable to make a TCP connection to the server and port specified in the license file. Possible reasons for this are: 1) the wrong license file is being referenced by the application program; 2) the server machine specified in the license file is down; 3) the vendor daemon specified in the license file is not running; 4) the host name in the license file is not recognized by the system; 5) the network between the client machine and the server machine is down; 6) You are mixing FLEX*lm* v1.5 (or earlier) and v2.1 (or later) vendor daemons in one license file and have run `lmgrd` without the `-b` command line option; 7) TCP is not running on your machine.

Solution: The `lmdiag` utility is designed primarily to debug this problem, so first, try `lmdiag`. Verify that the application is using the proper license file. Verify that specified server machine is up and reachable by executing another command that uses TCP, such as `telnet`, from the client to the server. Verify that the vendor daemon is running (you can use `ps` on the server to look for it). Examine the license log file to see if any problems are reported, particularly messages indicating that the vendor daemon has quit. Run `lmstat -a` from the server machine to verify that the vendor daemon is alive. Run `lmstat -a` from the client machine to verify the connection from client to vendor daemon across the network. Try using `telnet` *`host port`* where *`host`* and *`port`* are the same as on the SERVER line in your license file.

B.3.4 Other Client Problems

Symptom: *When I run my application program (or vendor daemon), I get the error* `bad code` *or* `inconsistent encryption code`*.*

Cause: Possible causes for this are: 1) the license file was modified (either the hostid on a SERVER line or anything on the FEATURE line was changed); 2) the vendor used the wrong version of his license creation program to generate your license file (or there is a bug in that process).

Solution: You may not modify the license file (except for specific fields, see Chapter 2, “The License File”). If you need to change something in your license file, you must get a new license file from your software vendor.

Symptom: *When the second user tries to check out a license, the vendor daemon prints an error concerning* `Parameter mismatch` *in the debug log file and refuses the license. The application may give the error:* `Duplicate selection mismatch for this feature.`

Cause: The most likely cause of this problem is that you are simultaneously trying to run two different versions of the application program, and the software vendor has not specifically set up the new version for this kind of compatibility. Check the debug log file for a comm version mismatch warning message; this indicates that someone is running an older client than the license server daemon, `lmgrd`.

Solution: Run only the new version of the application (or only the old version) or get a copy of a v8.0+ vendor daemon.

B.3.5 Other Server Problems

Symptom: *I'm having trouble restarting my license servers on Solaris.*

Cause: The default Solaris TCP port delay of four minutes may cause problems when restarting a license server, especially a license server that is using a hard-coded port number (for example, each of the three-server redundant servers) rather than selecting from a set of default port numbers.

Solution: If you have a problem restarting license servers on Solaris, a root user can reset the port delay to 2.4 seconds (2400 milliseconds). As a long-term solution, it is recommended that one of the following lines be put in a boot script because the port delay is reset every time the machine reboots:
Solaris 2.6-:

```
ndd -set /dev/tcp tcp_close_wait_interval 2400
```

Solaris 2.7+:

```
ndd -set /dev/tcp tcp_time_wait_interval 2400
```

Symptom: *The license server keeps reporting "lost lock" errors in the log file and exiting.*

Cause: The lock file (normally placed in `/usr/tmp` on UNIX (`/var/tmp` in FLEX*lm* v8.1), `C:\flexlm` on Windows NT) is being removed by someone else. There could be another daemon running, or the license administrator (or a script he set up) could have deleted the file.

Solution: Check to see if there is more than one copy of the daemon running. On UNIX use a command like `ps -aux | grep` *`vendor`* to search for it. Check for more than one `lmgrd` running as well, since it will restart your vendor daemon when it is killed. If more than one `lmgrd` is running, kill them all (using the `kill` command, not `kill -9`, on UNIX or LMTOOLS on Windows NT), then kill any remaining vendor daemons (on UNIX, try a simple `kill`, if that fails then try `kill -9` the `lmgrd` and all vendor daemons) and start one fresh copy of `lmgrd`. On UNIX, check to see if there is a shell script running that cleans out `/tmp` (or `/usr/tmp` or `/var/tmp`). If so, try modifying it so that it does not delete zero-length files.

Appendix C

FLEX*lm* Environment Variables

Environment variables should never be required to use FLEX*lm*-licensed applications. Environment variables are normally used for debugging or for changing license default location.

C.1 How to Set Environment Variables

FLEX*lm* environment variables are set in two different ways:

- The normal `set` or `setenv` commands (or the System Control Panel on Windows NT)
- The registry (Windows v6.0+) or in `$HOME/.flexlmrc` (UNIX v7.0+), which functions like the registry for FLEX*lm* on UNIX.

C.1.1 Registry

On Windows, the FLEX*lm* registry location is:

```
HKEY_LOCAL_MACHINE→SOFTWARE→FLEXlm License Manager
```

On UNIX, the equivalent information is stored in `$HOME/.flexlmrc`. In this file, the syntax is *`variable=value`*.

C.1.2 Precedence

If the variable is `LM_LICENSE_FILE` or *`VENDOR`*`_LICENSE_FILE`, then both the environment and the registry are used, with the environment used first, and the registry appended to the path.

If it's a different variable, then if the environment set, only that is used, otherwise the registry is used. That is, the registry is only used if the environment is not set.

C.2 Environment Variables

Variable	Use (FLEX*lm* version introduced)
FLEXLM_BATCH	Windows only: prevents interactive pop-ups from appearing. Set to 1 if a batch application. (Version 7.0+ clients)
FLEXLM_DIAGNOSTICS	Used for debugging where applications don't print FLEX*lm* error message text. Set to 1, 2, or 3, depending on the amount of diagnostic information desired. See Section B.2, "FLEXLM_DIAGNOSTICS." (Version 5.0+ clients)
LM_LICENSE_FILE or *VENDOR*_LICENSE_FILE	Reset path to license file. Can be a license-file list, separated by " : " on UNIX and " ; " on Windows. If *VENDOR*_LICENSE_FILE used, *VENDOR* is the vendor daemon name used by this application. For example, GLOBE*trotter* Software products use GSI_LICENSE_FILE. Can be a file name, or *port@host*. See also Section 2.1.2, "Setting the Path with an Environment Variable." (*VENDOR*_LICENSE_FILE requires v6.0+ clients.)
LM_PROJECT	LM_PROJECT's value is logged in the report log file and later reported on by SAM*report*. Limited to 30 characters. (v5.0+ client required.) This can also be used to RESERVE, INCLUDE, etc. licenses with PROJECT. For example: RESERVE 1 f1 PROJECT airplane v5.0+ clients and v7.0+ vendor daemon are required for this feature.

Appendix D

Frequently Asked Questions

D.1 License File Questions

D.1.1 I've received FLEX*lm* license files from two different companies. Do I have to combine them?

You don't have to combine license files. Each license file that has any counted lines (the license count field is > 0) requires a server. It's perfectly OK to have any number of separate license files, with different `lmgrd` server processes supporting each file. Moreover, since `lmgrd` is a lightweight process, for sites without system administrators, this is often the simplest (and therefore recommended) way to proceed. With v6.0+ `lmgrd`/`lmdown`/`lmreread`, you can restart/stop/reread a single vendor daemon (of any FLEX*lm* version). This makes combining licenses more attractive than previously. Also, if the application is v6.0+, using *`dir/*`*`.lic` for license file management behaves like combining licenses without physically combining them.

D.1.2 When is it recommended to combine license files?

Many system administrators, especially for larger sites, prefer to combine license files to ease administration of FLEX*lm* licenses. It's purely a matter of preference.

D.1.3 Does FLEX*lm* handle dates in the year 2000 and beyond?

Yes. The FLEX*lm* date format uses a four-digit year. Dates in the 20th century (19xx) can be abbreviated to the last two digits of the year (xx), and use of this feature is quite widespread. Dates in the year 2000 and beyond must specify all four year digits.

D.2 FLEX*lm* Versions

D.2.1 I have products from several companies at various FLEX*lm* version levels. Do I have to worry about how these versions work together?

If you're not combining license files from different vendors, the simplest thing to do is make sure you use the tools that are shipped by each vendor.

`lmgrd` will always correctly support older versions of vendor daemons and applications, so it's *always* safe to use the latest version of `lmgrd` and the `lmutil`/LMTOOLS utilities. If you've combined license files from two vendors, you *must* use the latest version of `lmgrd`.

If you've received two versions of a product from the same vendor, you *must* use the latest vendor daemon they send you. An older vendor daemon with a newer client will cause communication errors.

Please ignore letters appended to FLEX*lm* versions, for example, the "d" in v2.4d. The appended letter indicates a patch, and does NOT indicate any compatibility differences. In particular, some elements of FLEX*lm* didn't require certain patches, so a v2.4 `lmgrd` will work successfully with a v2.4b vendor daemon. See also Section H.1, "Version Compatibility and Components."

D.2.2 I've received a new copy of a product from a vendor, and it uses a new version of FLEX*lm*. Is my old license file still valid?

Yes. Older FLEX*lm* license files are always valid with newer versions of FLEX*lm*.

D.2.3 I've received a new license file, and the format is different from the old one I had from the same vendor? Why? Are they compatible?

As of v3.0, FLEX*lm* has an optional new format for license files. FLEX*lm* products always understand older versions; therefore, the pre-v3.0 files are understood by every FLEX*lm* version. However, your old applications (pre-FLEX*lm* v3.0) will not be able to use the new license file. See also Section 2.2, "License File Format."

D.3 Using FLEX*lm*

D.3.1 Does FLEX*lm* work across the Internet?

Yes. A server on the Internet will serve licenses to anyone else on the Internet. This can be limited with the INTERNET= attribute on the FEATURE line, which limits access to a range of Internet addresses. You can also use the INCLUDE and EXCLUDE options in the daemon option file to allow (or deny) access to clients running on a range of Internet addresses.

D.3.2 Does FLEX*lm* work with Internet firewalls?

Many firewalls require that port numbers be specified to the firewall. FLEX*lm* v5 `lmgrd` supports this.

D.3.3 If my client dies, does the server free the license?

Yes, unless the client's whole system crashes, or becomes disconnected from the network. Assuming communications is TCP, the license is automatically freed immediately. If communications are UDP, then the license is freed after the UDP timeout, which is set by each vendor, but defaults to 45 minutes. UDP communications is normally only set by the end user, so TCP should be assumed. If the whole system crashes, then the license is not freed, and you should use `lmremove` to free the license.

D.3.4 What happens when the license server dies?

FLEX*lm* applications send periodic heartbeats to the server to discover if it has died. What happens when the server dies is then up to the application. Some will simply continue periodically attempting to re-checkout the license when the server comes back up. Some will attempt to re-checkout a license a few times, and then, presumably with some warning, exit. Some GUI applications will present pop-ups to the user periodically letting them know the server is down and needs to be restarted.

D.3.5 How do you tell if a port is already in use?

99.44% of the time, if it's in use, it's because `lmgrd` is already running on the port—or was recently killed, and the port isn't freed yet. Assuming this is not the case, then use `telnet` *`host port`*— if it says "Can't connect," it's a free port.

D.3.6 Does FLEX*lm* require root permissions?

No. There is no part of FLEX*lm*, `lmgrd`, vendor daemon, or application, that requires root permissions. In fact, it is *strongly recommended* that you do not run the license server (`lmgrd`) as "root," since root processes can introduce

security risks. If `lmgrd` must be started from the root user (for example, in a system boot script), it is recommended that you use the `su` command to run `lmgrd` as a non-privileged user:

```
su username -c "lmgrd_path -c license_file_path -l debug_log_path"
```

where *`username`* is a non-privileged user, and the paths are the correct paths to `lmgrd`, the license file, and the debug log file. You will have to ensure that the vendor daemons listed in the license file have execute permissions for *`username`*. The paths to all the vendor daemons in the license file are listed on each DAEMON line.

D.3.7 Is it OK to run lmgrd as "root" (UNIX only)?

It is not prudent to run any command, particularly a daemon, as root on UNIX, as it may pose a security risk to the operating system. Therefore, it is recommended that `lmgrd` be run as a non-privileged user (not "root"). For examle, if you are starting `lmgrd` from a boot script, use:

```
su username -c "umask 022; lmgrd..."
```

to run `lmgrd` as a non-privileged user.

D.3.8 Does FLEX*lm* licensing impose a heavy load on the network?

No, but partly this depends on the application, and end user's use. A typical checkout request requires five messages and responses between client and server, and each message is <150 bytes.

When a server is not receiving requests, it requires virtually no CPU time.

When an application, or `lmstat`, requests the list of current users, this can significantly increase the amount of networking FLEX*lm* uses, depending on the number of current users.

Also, prior to FLEX*lm* v5, use of *`port@host`* can increase network load, since the license file is downloaded from the server to the client. *`port@host`* should be, if possible, limited to small license files (say, <50 features). In v5, *`port@host`* actually *improves* performance.

D.3.9 Does FLEX*lm* work with NFS?

Yes. FLEX*lm* has no direct interaction with NFS. FLEX*lm* uses an NFS-mounted file like any other application.

D.3.10 Does FLEX*lm* work with ATM, ISDN, Token-Ring, etc.

In general, these have no impact on FLEX*lm*. FLEX*lm* requires TCP/IP. So long as TCP/IP works, FLEX*lm* will work.

D.3.11 Does FLEX*lm* work with subnets, fully qualified names, multiple domains, etc.?

Yes, although this behavior was improved in v3.0, and v6.0.

When a license server and a client are located in different domains, fully qualified host names have to be used. A "fully qualified host name" is of the form *node*`.`*domain*, where *node* is the local host name (usually returned by the `hostname` command or `uname -n`) and *domain* is the Internet domain name, e.g., "globes.com."

To ensure success with FLEX*lm* across domains, do the following:

1. Make the sure the fully qualified host name is the name on the SERVER line of the license file.
2. Make sure ALL client nodes, as well as the server node, are able to `telnet` to that fully qualified host name. For example, if the host is locally called "speedy," and the domain name is "corp.com," local systems will be able to logon to "speedy" via `telnet speedy`. But very often, `telnet speedy.corp.com` will fail locally. Note that this `telnet` command will always succeed on hosts in other domains (assuming everything is configured correctly), since the network will resolve "speedy.corp.com" automatically.
3. Finally, there must be an alias for "speedy" so it's also known locally as "speedy.corp.com." This alias is added to the `/etc/hosts` file, or if NIS/Yellow Pages are being used, then it will have to be added to the NIS database. This requirement goes away in version 3.0 of FLEX*lm*.

If all components (application, `lmgrd` and vendor daemon) are v6.0 or higher, no aliases are required; the only requirement is that the fully qualified domain name, or IP address, is used as a host name on the SERVER, or as a host name in the `LM_LICENSE_FILE` environment variable (*port*`@`*host* or `@`*host*).

D.3.12 Does FLEX*lm* work with NIS and DNS?

Yes. However, some sites have broken NIS or DNS, which will cause FLEX*lm* to fail. In v5 of FLEX*lm*, NIS and DNS can be avoided to solve this problem. In particular, sometimes DNS is configured for a server that's not current available (e.g., a dial-up connection from a PC). Again, if DNS is configured, but the server is not available, FLEX*lm* will fail.

On PCs, if a checkout seems to take three minutes and then fails, this is usually because the system is configured for a dial-up DNS server which is not currently available. The solution here is to turn off DNS.

Finally, host names must NOT have periods in the name. These are not legal host names, although PCs will allow you to enter them, and they will not work with DNS.

D.3.13 We are using FLEX*lm* over a wide-area network. What can we do to improve performance?

FLEX*lm* network traffic should be minimized. With the most common uses of FLEX*lm*, traffic is negligible. In particular, checkout, checkin, and heartbeats use very little networking traffic. There are two items, however, which can send considerably more data and should be avoided or used sparingly:

- `lmstat -a` should be used sparingly. `lmstat -a` should not be used more than, say, once every 15 minutes, and should be particularly avoided when there's a lot of features, or concurrent users, and therefore a lot of data to transmit; say, more than 20 concurrent users or features.
- Prior to FLEX*lm* v5, the *`port@host`* mode of the `LM_LICENSE_FILE` environment variable should be avoided, especially when the license file has many features, or there are a lot of license files included in `LM_LICENSE_FILE`. The license file information is sent via the network, and can place a heavy load. Failures due to *`port@host`* will generate the error `LM_SERVNOREADLIC (-61)`.

D.4 Windows Questions

D.4.1 What PC Platforms are supported?

FLEX*lm* v8.1 supports Windows 95/98, NT, 2000, ME, and XP.

D.4.2 Will the Server run on Windows 95?

Yes.

Appendix E

FLEX*lm* Error Codes

E.1 Error Message Format

FLEX*lm* error messages presented by applications have the following components:

- FLEX*lm* Error Number—a negative number starting at -1.
- FLEX*lm* Error Text—short sentence (< 80 characters) summarizing problem.
- FLEX*lm* Error Explanation (optional)—short paragraph (3-5 lines) explaining problem and possible solutions or workarounds.
- FLEX*lm* Minor Error Number—a positive number starting at 1. These numbers are unique error identifiers and are used by FLEX*lm* vendors for more advanced support assistance. Their meaning is not documented.
- System Error Number (optional)—a UNIX or Windows OS error code last set by the operating system.
- System Error Explanation (optional)—a short sentence (< 80 characters) explaining the system error.
- Other supporting information (optional)

Error messages were improved in v6. FLEX*lm* Error Explanation, and supporting information are only available in applications using v6.0+.

These error messages may occur in two formats available with FLEX*lm* or may appear in a format customized by the application.

E.1.1 Format 1 (short):

`FLEXlm error text (-lm_errno, minor_num[:sys_errno]) [sys_error_text]`

The system error information may be missing.

Example:

```
Can't connect to license server (-15,12:61) Connection refused
```

E.1.2 Format 2 (long—version 6.0+):

```
FLEXlm error text
 FLEXlm error explanation
[Optional Supporting information]
FLEXlm error: -lm_errno, minor_num. [System Error: sys_errno]
[“system_error_text”]
```

Example:

```
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.
Feature:       f1
Server name:   localhost
License path: @localhost:license.dat:./*.lic
FLEXlm error:  -15,12.  System Error: 61 “Connection refused”
```

E.2 Error Code Descriptions

Errors marked with an “ * ” indicates errors which should not appear in shipping software. These are errors intended to help the programmer incorporate FLEX*lm* in their product, and should be fixed before shipping.

Errors marked with “ + ” indicate errors due to an operating system failure.

Error	Description
-1	Cannot find license file.
-2	Invalid license file syntax.
-3	No server for this feature.
-4	Licensed number of users already reached.
-5	No such feature exists.
-6	No port number in license file and FLEXlm service does not exist. (pre-v6 only)
-7	No socket connection to license manager service.
-8	Invalid (inconsistent) license key or signature. The license key/signature and data for the feature do not match. This usually happens when a license file has been altered.

-9	Invalid host. The hostid of this system does not match the hostid specified in the license file.
-10	Feature has expired.
-11	Invalid date format in license file.
-12	Invalid returned data from license server.
-13	No SERVER lines in license file.
-14	Cannot find SERVER host name in network database. The lookup for the host name on the SERVER line in the license file failed. This often happens when NIS or DNS or the hosts file is incorrect. Workaround: Use IP address (e.g., 123.456.789.123) instead of host name.
-15	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 host name in the license file has been changed.
-16	Cannot read data from license server.
-17	Cannot write data to license server.
-18	License server does not support this feature.
-19	Error in select system call.
-20	[Obsolete]
-21	License file does not support this version.
-22	Feature checkin failure detected at license server.
-23	License server temporarily busy (new server connecting).
-24	Users are queued for this feature.
-25	License server does not support this version of this feature.
-26	Request for more licenses than this feature supports.

-27	[Obsolete]
-28	[Obsolete]
-29	Cannot find ethernet device.
-30	Cannot read license file.
-31	Feature start date is in the future.
-32	No such attribute.
-33	Bad encryption handshake with daemon.
-34	Clock difference too large between client and server.
-35	In the queue for this feature.
-36	Feature database corrupted in daemon.
-37	Duplicate selection mismatch for this feature. Obsolete with v8.0+ vendor daemon.
-38	User/host on EXCLUDE list for feature.
-39	User/host not on INCLUDE list for feature.
-40	Cannot locate dynamic memory.
-41	Feature was never checked out.
-42	Invalid parameter.
-43	* No FLEXlm key data supplied in initializing call.
-44	* Invalid FLEXlm key data supplied.
-45	* FLEXlm function not available in this version.
-46	[Obsolete]
-47	Clock setting check not available in daemon.
-48	* FLEXlm platform not enabled.
-49	* Date invalid for binary format.
-50	* FLEXlm key data has expired.

-51	* FLEXlm not initialized.
-52	FLEXlm vendor daemon did not respond within timeout interval.
-53	Checkout request rejected by vendor-defined checkout filter.
-54	No FEATURESET line in license file.
-55	Incorrect FEATURESET line in license file.
-56	Cannot compute FEATURESET data from license file.
-57	+ socket() call failed.
-58	[Obsolete]
-59	Message checksum failure.
-60	Server message checksum failure.
-61	Cannot read license file data from server.
-62	Network software (TCP/IP) not available.
-63	You are not a license administrator.
-64	lmremove request before the minimum lmremove interval.
-65	* Unknown VENDORCODE struct type passed to lm_init().
-66	* FLEXlm include file/library version mismatch.
-67	No licenses to borrow.
-68	License BORROW support not enabled.
-69	FLOAT_OK can’t run standalone on SERVER.
-71	Invalid TZ environment variable.
-72	* Old VENDORCODE (three-word) struct type passed to lm_init().
-73	Local checkout filter rejected request.

-74	Attempt to read beyond end of license file path.
-75	+ SYS$SETIMR call failed (VMS).
-76	Internal FLEXlm error - please report to GLOBE*trotter* Software.
-77	Bad version number - must be floating-point number with no letters.
-78	* FLEXlm internal error -78.
-79	* FLEXlm internal error -79.
-80	* FLEXlm internal error -80.
-81	* FLEXlm internal error -81.
-82	Invalid PACKAGE line in license file.
-83	FLEXlm version of client newer than server.
-84	USER_BASED license has no specified users - see server log.
-85	License server doesn’t support this request.
-86	License object already in use (Java only).
-87	Checkout exceeds MAX specified in options file.
-88	System clock has been set back.
-89	This platform not authorized by license.
-90	Future license file format or misspelling in license file. The file was issued for a later version of FLEX*lm* than this program understands.
-91	ENCRYPTION_SEEDS are non-unique.
-92	Feature removed during lmreread, or wrong SERVER line hostid.

-93	This feature is available in a different license pool. This is a warning condition. The server has pooled one or more INCREMENT lines into a single pool, and the request was made on an INCREMENT line that has been pooled.
-94	Attempt to generate license with incompatible attributes.
-95	Network connect to this_host failed. The license file indicates `this_host`, and the server is not running on this host. If it's running on a different host, `this_host` should be changed to the correct host.
-96	Server node is down or not responding. See the system administrator about starting the server, or make sure that you're referring to the right host (see `LM_LICENSE_FILE` environment variable).
-97	The desired vendor daemon is down. 1) Check the `lmgrd` log file, or 2) Try `lmreread`.
-98	This FEATURE line can't be converted to decimal format.
-99	The decimal format license is typed incorrectly.
-100	Cannot remove a linger license.
-101	All licenses are reserved for others. The system administrator has reserved all the licenses for others. Reservations are made in the options file. The server must be restarted for options file changes to take effect.
-102	[Unused]
-103	Terminal Server remote client not allowed.
-104	Cannot borrow that long.
-105	[Unused]
-106	License server out of network connections. The vendor daemon can't handle any more users. See the debug log for further information.

-107	[Unused]
-108	[Unused]
-109	[Unused]
-110	Dongle not attached, or can't read dongle. Either the hardware dongle is unattached, or the necessary software driver for this dongle type is not installed.
-112	Missing dongle driver. In order to read the dongle hostid, the correct driver must be installed. These drivers are available at http://www.globetrotter.com or from your software vendor.
-113	2 FLEXlock checkouts attempted. Only one checkout is allowed with FLEX*lock*-enabled apps.
-114	SIGN= keyword required, but missing from license. This is probably because the license is older than the application. You need to obtain a SIGN= version of this license from your vendor.
-115	Error in Public Key package.
-116	CRO not supported for this platform.
-117	BORROW failed.
-118	BORROW period has expired.
-119	lmdown and lmreread must be run on license server node.
-120	Cannot lmdown the server when licenses are borrowed.
-121	FLOAT_OK license must have exactly one dongle hostid.

Appendix F

The Debug Log File

FLEX*lm* daemons generate debug log files in the following format:

hh:*mm*:*ss* (*daemon*) *message*

where:

hh:*mm*:*ss*	Time that the message was logged.
daemon	Either `lmgrd` or the vendor daemon name. In the case where a single copy of the daemon cannot handle all of the requested licenses, an optional "_" followed by a number indicates that this message comes from a forked daemon.
message	The text of the message.

The debug log files can be used to:

- Diagnose configuration problems
- Diagnose daemon software errors

Note: A debug log file cannot be used for usage reporting with SAM*report.*

F.1 Informational Messages

Connected to *host*

This daemon is connected to its peer on *host*.

CONNECTED, master is *host*

The license daemons log this message when a quorum is up and everyone has selected a master.

DENIED: *num_lic feature* to *user*

user was denied access to *num_lic* licenses of *feature*.

EXITING DUE TO SIGNAL *nnn*
EXITING with code *nnn*

All daemons list the reason that the daemon has exited.

EXPIRED: *feature*

feature has passed its expiration date.

IN: "*feature*" *user* (*num_lic* licenses)

user has checked in *num_lic* licenses of *feature*.

Lost connection to *host*

A daemon can no longer communicate with its peer on node *host*, which can cause the clients to have to reconnect, or cause the number of daemons to go below the minimum number, in which case clients may start exiting. If the license daemons lose the connection to the master, they will kill all the vendor daemons; vendor daemons will shut themselves down.

Lost quorum

The daemon lost quorum, so will process only connection requests from other daemons.

MULTIPLE *vendor* servers running. Please kill, and restart license daemon.

The license daemon has detected that multiple licenses for vendor daemon *vendor* are running. The user should kill all *vendor* processes and restart the license daemon.

OUT: "*feature*" *user* (*num_lic* licenses)

user has checked out *num_lic* licenses of *feature*.

RESERVE *feature* for USER *user*
RESERVE *feature* for HOST *host*

A license of *feature* is reserved for either *user* or *host*.

REStarted *vendor* (internet port *nnn*)

Vendor daemon *vendor* was restarted at Internet port *nnn*.

Retrying socket bind (address in use)

The license servers try to bind their sockets for approximately six minutes if they detect "address in use" errors.

Selected (EXISTING) master *host*.

This license daemon has selected an existing master *host* as the master.

SERVER shutdown requested.

A daemon was requested to shut down via a user-generated `kill` command.

Server started on *host* for: "*feature_list*"

A (possibly new) server was started for the features listed.

Shutting down *vendor*

The license manager daemon is shutting down the vendor daemon *vendor*.

SIGCHLD received. Killing child servers.

A vendor daemon logs this message when a shutdown was requested by the license daemon.

Started *vendor*

The license manager daemon logs this message whenever it starts a new vendor daemon.

Trying to connect to *host*

The daemon is attempting a connection to *host*.

F.2 Configuration Problem Messages

***host*: Not a valid server host, exiting**

This daemon was run on an invalid host name.

***host*: Wrong hostid, exiting**

The hostid is wrong for *host*.

BAD CODE for *feature*

The specified feature name has a bad license key or signature. It was probably typed in wrong, or modified by the end user.

CANNOT OPEN options file *file*

The options file specified in the license file could not be opened.

Couldn't find a master

The daemons could not agree on a master.

License daemon: lost all connections

This message is logged when all the connections to a server are lost, which often indicates a network problem.

Lost lock, exiting
Error closing lock file
Unable to re-open lock file

The vendor daemon has a problem with its lock file, usually because of an attempt to run more than one copy of the daemon on a single node. Locate the other daemon that is running via a `ps` command, and kill it with `kill -9`.

No DAEMON line for *`vendor`*

The license file does not contain a DAEMON or VENDOR line for *`vendor`*.

No DAEMON lines, exiting

The license daemon logs this message if there are no DAEMON or VENDOR lines in the license file. Because there are no vendor daemons to start, there is nothing for the license daemon to do.

No features to serve!

A vendor daemon found no features to serve. This could be caused by a corrupted or incorrectly entered license file.

UNSUPPORTED FEATURE request: *`feature`* by *`user`*

The user has requested a feature that this vendor daemon does not support. This can happen for a number of reasons: the license file is bad, the feature has expired, or the daemon is accessing the wrong license file.

Unknown host: *`host`*

The host name specified on a SERVER line in the license file does not exist in the network database (probably `/etc/hosts`).

F.3 Daemon Software Error Messages

accept: message

An error was detected in the “accept” system call.

Can’t allocate server table space

A malloc error. Check swap space.

Connection to *host* TIMED OUT

The daemon could not connect to *host*.

Illegal connection request to *vendor*

A connection request was made to *vendor*, but this vendor daemon is not *vendor*.

read: error message

An error in a “read” system call was detected.

select: message

An error in a “select” system call was detected. This is usually a sign of a system networking failure.

Server exiting

The server is exiting. This is normally due to an error.

Appendix G

Platform-Specific Notes

G.1 Novell Netware

On Netware systems, there is no `lmgrd` and the vendor daemon is run directly.

The lock file on Netware is normally placed in `SYS:\SYSTEM\FLEXLM`.

G.2 VMS

On VMS systems, there is no `lmgrd` and the vendor daemon is run directly.

The recommended location for the `lmutil` tools is `SYS$COMMON:[SYSMGR]` on VMS.

In `LM_LICENSE_FILE` or a license-file list, use a space (“ ”) to separate file names.

If you get the error message `socket bind: permission denied (13)`, the daemon needs to bind the socket address in order to be able to listen for connections from clients. This is done in the system name table, so it requires the SYSNAM privilege.

Appendix H

FLEX*lm* Versions

H.1 Version Compatibility and Components

In general, always use the latest `lmgrd` and `lmutil`/LMTOOLS, which are available from http://www.globetrotter.com, and you'll automatically enjoy many of the enhancements available in the most recent versions of FLEX*lm*. However, some enhancements require a vendor daemon built with a newer version of FLEX*lm*, and yet others require a client application built with a newer version of FLEX*lm*. Contact your software vendor for the most current version of their vendor daemon.

The rules about FLEX*lm* version compatibility can be summarized as:

Version of `lmutil`/LMTOOLS must be >=

Version of `lmgrd`, which must be >=

Version of vendor daemon, which must be >=

Version of client application, which must be >=

Version of license file format

Except for the license file, you can use `lmver` to discover the version of all these components. For the vendor daemon, `lmgrd`, and `lmutil`, you can also use the `-v` argument to print the version.

H.2 How to Tell the License File Version

The following rules apply to individual FEATURE, INCREMENT or UPGRADE lines. It's possible to have a mix of versions in a single file. Only the features that a particular application checks out determine the version of the license for that feature.

Version 2	Blank quotes or a quoted string at the end of the FEATURE line.
>= Version 3	INCREMENT or UPGRADE line.

>= Version 4	OVERDRAFT, DUP_GROUP, INTERNET, or PACKAGE appear.
>= Version 5	SUPERSEDE, ISSUED, USER_BASED, HOST_BASED, or SN appear.
>= Version 6	START appears.
>= Version 7.1	SIGN= keyword appears.
>= Version 8	BORROW, FLOAT_OK, and TS_OK appear.

H.3 Version Summary

V1.0—1988

First FLEX*lm* Release, containing all the basic FLEX*lm* features

V1.5—FEBRUARY 1990

First widely used version including DEMO

V2.1—MARCH 1991

- Improved TIMEOUT support
- Improved ethernet hostid support

V2.21—NOVEMBER 1991

- Added support for many platforms and some platform-specific improvements, such as hostid
- Hostid ANY added

V2.26—MARCH 1992 (USED ONLY BY SUN)

- Added license lingering

V2.4—DECEMBER 1992

- Added “use-all-feature-lines” capability for incremental license distribution
- Enhanced vendor customization routines
- Enhanced end-user options file
- Added new hostid types: USER, HOSTNAME, and DISPLAY
- Added `port@host` to locate license file —downloads license file from server

V2.61—MARCH 1993 (USED ONLY BY SUN)

- Added INCREMENT and UPGRADE lines to license file

V3.0—MAY 1994

- INCREMENT and UPGRADE behavior changed and improved
- Added UDP protocol support
- Added `uname -i` hostid for HP
- Added multiple jobs for enhanced support of `LM_LICENSE_FILE` environment variable as a license-file list
- New, optional license file format with *`keyword=value`* syntax for optional new features, including: asset_info, ISSUER, and NOTICE, " \ " license file continuation character, 2048 character limit per feature

V4.0—DECEMBER 1994

- Removed use of floating point, for enhanced reliability
- FEATURE line additions: ck, OVERDRAFT, DUP_GROUP, INTERNET hostid
- PACKAGE line
- License Finder
- `lmdiag` and `FLEXLM_DIAGNOSTICS` for end-user diagnostics

V4.1—MAY 1995

- Performance improvements and new platform support

V4.1—PATCH RELEASE 6, OCTOBER 1995

- PC patch release for Windows 95 with various performance improvements

V5.0—MARCH 1996

- Improved *`port@host`* behavior—client application doesn't read license file
- Automatic *`port@host`* via USE_SERVER line in license file
- Hostid lists—lock a feature to several hostids
- New FEATURE attributes: SN (serial number), USER_BASED, HOST_BASED, MINIMUM, SUPERSEDE, ISSUED (issued date), CAPACITY (charging based on system capacity)
- Optional avoidance of NIS and DNS via IP address instead of host name
- Improved report log file format
- Server, upon startup, notifies of licenses that will expire within two weeks
- Improved end-user options file functionality

V5.11—FEBRUARY 1997

- SUPERSEDE lists, PLATFORMS= license attribute,
- new end-user options: MAX, TIMEOUTALL

- Windows control panel added
- Windows license generator `GENLIC` added

V5.12—APRIL 1997

- Performance improvements and new platform support

V6.0—SEPTEMBER 1997

- `lmgrd` can read multiple license files
- FLEX*lm* license directory support: `*.lic` automatically used
- License files require no editing for use at the end-user site
- Optional path on DAEMON/VENDOR line; $`PATH` environment variable used
- Decimal license format, with `lminstall` utility for typing in licenses
- FEATURE lines are shorter, easier to understand and type in
- PACKAGE lines can be shipped in separate files that never require user editing
- Default port numbers make SERVER line port number optional
- Default end-user options file path
- `this_host` host name supported on SERVER line
- *`VENDOR`*`_LICENSE_FILE` supported (e.g., `GSI_LICENSE_FILE`)
- `@`*`host`* supported where default port numbers are used
- Windows only: user prompted for license file or license server name
- License files are optionally case insensitive
- `lmdown` and `lmreread` accept `-vendor` *`vendor`* argument
- `START=`*`dd-mmm-yyyy`* optional license attribute

V6.1—JUNE 1998

- Performance improvements

V7.0—AUGUST 1999

- License Certificate Manager support for automatic license fulfillment
- Support for “try-before-you-buy” licensing
- License file handles inserted newlines from emailers
- License lines automatically optimally sorted
- Improved LMTOOLS interface for Windows
- `lmgrd`, when run at command line on Windows, runs in background by default

- Improved three-server redundancy reliability (v7.0 vendor daemon and `lmgrd`)
- `lmreread` and `lmdown` take `-all` argument to shut down or reread all `lmgrd`s
- Support registry (Windows) and `$HOME/.flexlmrc` (UNIX) for FLEX*lm* environment variables
- Automatically install license path in registry or `$HOME/.flexlmrc` after successful checkout
- Options support for `LM_PROJECT` with PROJECT
- Performance improvements, especially for Windows NT
- Intel Pentium III CPU-ID (v7.0d+, November 1999)

V7.1—AUGUST 2000

- Security enhancements
- SIGN= keyword in license
- `lmnewlog` utility (v7.1+ vendor daemon

V7.2—DECEMBER 2000

- Performance enhancements

V8.0—OCTOBER 2001

- `lmborrow` (v8.0+ components), `lmpath` (v7.0+ vendor daemon), `lmswitch` (v8.0+ vendor daemon) utilities
- `lmreread` rereads end-user options file and SERVER host name
- License borrowing with BORROW keyword

V8.1—JANUARY 2002

- CRO Security enhancements

Index

W

Y