License Administration Guide

FLEXnet Publisher Licensing Toolkit 11.6

FNP-116-LA02

Legal and Contact Information

Part Number: FNP-116-LA02
Product Release Date: May 2008

Contacting Acresso Software

You may contact us from anywhere in the world by visiting the Acresso Web site at http://www.acresso.com/company.htm.

Copyright Notice

Trademarks

Restricted Rights Legend

Contents

What's New in this Document

This section describes the areas of this document that have been updated for this release.

New License Server Manager, lmadmin

A GUI-based license server manager is included in this release. It is available on a limited number of platforms: Windows 32-bit, Mac OS X, and Red Hat Enterprise Linux 32-bit. A brief description of `lmadmin` is included in this document in lmadmin - GUI-based License Server Manager. For full details of how to install and use `lmadmin` see *License Server Manager (lmadmin) Installation Guide* and the on-line help provided in `lmadmin`.

Delaying the Start of the License Server Manager

Configuring the License Server Manager Service for a Delayed Start describes how to configure your `lmgrd` license server system so that the license server manager starts up after all other required components are loaded. A typical scenario where a delay is needed is when a FLEXnet ID dongle is used to lock the license server to a machine.

1

Selecting Systems to Run the License Server

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

Resources Used by the License Server

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

Sockets

When using TCP/IP ports, each FLEXenabled application connected to a license server uses one or more sockets. Depending on how the software publisher implemented FLEXnet Publisher Licensing Toolkit, the FLEXenabled application may need one or more sockets. Consult with the software publisher for this information. The per-process system limit for file descriptors determines the number of sockets available to the license server. The total number of sockets that the license server uses is slightly larger than the total number needed by the FLEXenabled applications that connect to it.

If the number of sockets required by the license server on a single system becomes excessive, then it's probably good to split the license file into more than one file, onto different servers, to lighten the networking traffic (which requires the software publisher to agree to issue new licenses). FLEXenabled applications then check out licenses from multiple servers using a license search path via the LM_LICENSE_FILE environment variable.

CPU Time

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

For a large number of clients (are each exchanging heartbeat messages with the license 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 system you select has enough CPU cycles to spare.

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 FLEXnet Manager. If you have a lot of license activity, these log files grow very large. You need to consider where to put these files and how often to rotate and archive them. You have the option to suppress log file output if disk space is at a premium.

It is recommended that the log files are local files on the server systems to avoid networking dependencies.

See Also
Setting the License Search Path using an Environment Variable
Starting the License Server Manager on UNIX Platforms
Report Log File
Debug Log File

Memory

The license server uses little memory.The vendor daemons use approximately 2 MB each, although memory usage increases in the vendor daemon with the size of the license file, size of the options file, and the number of concurrent users. The GUI-based license server manager, `lmadmin`, uses between 7 and 10 MB of memory during typical usage. On Solaris the command line license server manager, `lmgrd`, uses approximately 2 MB.

Network Bandwidth

FLEXnet Publisher Licensing Toolkit sends relatively small amounts of data across the network. Each transaction, such as a checkout or checkin, generally transfers less than 1 KB of data. This means that FLEXnet Publisher Licensing Toolkit can be effectively run over slow networks (such as dial-up SLIP lines) for small numbers of clients.

For a large number of FLEXenabled applications (hundreds), each of which exchange heartbeat messages with the vendor daemon, the network bandwidth used may start to become significant. In this case, run the FLEXenabled application and server on the same local area network, which may require splitting licenses between two files for two servers. Users can use a license search path in the LM_LICENSE_FILE environment variable to have effective access to both servers. Enterprises can experience a performance issue when there is slow network communication or if FLEXenabled clients are using a dialup link to connect to the network.

When you are using the GUI-based license server manager, `lmadmin`, which uses HTTP, you need to consider the clients that connect to the `lmadmin` user interface. Depending on the number of clients and the frequency of the page refresh, they can impose a significant burden on network traffic.

See Also
Specifying the Location of the License File

Remote Mounted Disks

It is recommended that you do not use remote mounted disks when you run the license server. In other words, it is recommended that `lmadmin` or `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 are on a remote mounted disk, you double the points of failure. This could lead to a temporary loss of all of your licenses. When all files are mounted locally, the licenses are available as long as the server is running. When the files are on a different system, licenses may become unavailable if the license server or file server fails.

2

lmadmin - GUI-based License Server Manager

The *license server manager* is one of the components that make up a license server (the other being the vendor daemon). It handles the initial contact with FLEXenabled applications, passing the connection on to the appropriate vendor daemon. The purpose of the license server manager is to:

- Start and maintain vendor daemons as required for serving license rights from different software publishers.
- Refer application checkout (or other) requests to the correct vendor daemon.

There are two versions of the license server manager:

- **lmadmin** - the latest license server manager with a graphical user interface.
- **lmgrd** - the original license server manager with a command line interface.

This section describes `lmadmin`, for information on `lmgrd` see lmgrd - License Server Manager.

`lmadmin` provides improved methods of managing the license server and vendor daemons. A brief description of the improved capabilities follows, for more detailed information on `lmadmin` see *License Server Manager (lmadmin) Installation Guide* and the on-line help available in `lmadmin`. `lmadmin` and supporting documentation is freely available from the Acresso download site.

lmadmin Capabilities

- Direct configuration of the vendor daemons and license server manager - license server port number; vendor daemon path and port; and three-server redundant port can be configured without any edits to the license files.
- Configurable alerts - you can set up `lmadmin` to issue alerts to warn you of potential problems, for example: license expiry, no available licenses, or vendor daemon status.
- License rights status display - configurable display of all available and in use license rights. This display can include all concurrent (floating) licenses both from license files or from trusted storage. It can also include activatable licenses (held in trusted storage) when these are available on the license server.

- GUI buttons replace command line utilities - for example 'Stop Server' and 'Reread License Files'. For a list of license administration functions that are available directly from `lmadmin` see lmadmin License Administration Functions.
- Minimal editing of license files - option file specification requires editing.

This release of `lmadmin` is available for a limited number of platforms. For full details contact your software publisher or see the Acresso download site. `lmadmin` is compatible with licensing components from version 9.2 or later. See Version Compatibility between Components for detailed information on how to determine what versions of the licensing components are provided in your licensed applications.

lmadmin License Administration Functions

`lmadmin` provides some of the license administration functions previously provided by the command-line based license administration utilities or LMTOOLS on the Windows platform. The following table lists functions provided within `lmadmin` that replace those provided by the license administration utilities.

Table 2-1: lmadmin License Administration Functions

lmadmin Function	Description	Replaces Utility
Dashboard - Licenses	Displays details of licenses rights available and in use.	lmstat
Vendor Daemon Configuration - Administer - Stop	Stops the vendor daemon.	lmdown - some usage cases
Vendor Daemon Configuration - Administer - Reread License Files	Rereads license rights from license files included in the lmadmin configuration. Only required when the content of a license file is updated.	lmreread
Vendor Daemon Configuration - Administer - Rotate Report Logs	Switches the report log to a new file name.	lmswitchr
Server Configuration - Stop Server	Stop the license server. Note that lmadmin's default setting disables this button, to enable it start lmadmin with the -allowStopServer argument.	lmdown - some usage cases

The following table details which command-line utilities may no longer be required and which utilities are required when using `lmadmin`.

Table 2-2: lmadmin Use of License Administration Utilities

Utility	Required when using lmadmin
lmborrow	Yes, if using license rights in license files and borrow capability.
lmdiag	Yes, to diagnose license checkout problems.
lmdown	Not normally required. Note that `lmadmin's` default setting disables the operation of lmdown. To enable it, start `lmadmin` with the `-allowStopServer` argument.

Table 2-2: lmadmin Use of License Administration Utilities (cont.)

Utility	Required when using lmadmin
lmhostid	Not normally required for `lmadmin` as it displays information about the system it is running on that includes the various identities normally used as hostids. Required for determining the hostids of client systems.
lminstall	Yes, converts license files between different formats.
lmnewlog	Yes, if you use this function instead of lmswitchr to change to a new report log because you do not want to edit the report log file name in the Options file.
lmpath	Yes, allows users direct control over license file path settings.
lmremove	Yes, releases a hung license to the pool of free licenses. Note that lmadmin's default setting disables the operation of lmremove, to enable it start lmadmin with the -allowLicenseReclaim argument.
lmreread	Not normally required.
lmstat	Only required to show additional information (such as borrow or reservations).
lmswitch	Yes, controls debug log location and size.
lmswitchr	Not required.
lmver	Yes, reports the version of a library or binary file. Note that you can determine the version of `lmadmin` by starting it with the `-version` argument.

Manually Starting the License Server Manager

You can start the license server using one of the following methods:

- On Windows platforms, open the installation directory in Windows Explorer and then double-click the `lmadmin.exe` file. This mechanism does not allow you to specify non-default command-line options.
- Execute the `lmadmin` command from the root installation directory. To see a list of available command-line options, execute the command:

  ```
  lmadmin -help
  ```

  The help display identifies the default options and which options are *persistent*, options that will remain in effect for later instances of `lmadmin`.
- Create a shell script file (Unix) or a batch file (Windows) that will run the `lmadmin` command with your desired command-line options and then execute that file.

Note: *If either the default license server port or the HTTP port for the user interface is in use, the server will not start. Use the -licPort and/or -webPort options to override the defaults if you have a conflict.*

Manually Stopping the License Server Manager

The `allowStopServer` command-line option toggles the presense of the **Stop Server** button in the `lmadmin` user interface. The default is the **Stop Server** button is not present. This allows non-administrator use of the interface for monitoring purposes.

The `allowStopServer` command-line option is persistent. If `lmadmin` is started, or had been previously started, with the option `-allowStopServer yes`, then in the Administration section of the license server management interface, the Server Configuration tab includes the **Stop Server** button. Click the **Stop Server** button to shut down the license server manager (`lmadmin`) and all vendor daemons.

If `lmadmin` is started with the command-line option `-allowStopServer no`, if the persistent command-line option `-allowStopServer yes` has never been used, or if `-allowStopServer no` was the most recent use of the `-allowStopServer` option , you cannot stop the license server using the license server management interface. In this situation, to stop the license server you must stop the `lmadmin` process.

On Unix systems you can use the *ps* utility to identify the process and the `kill` command to terminate it. (Please note that you should not use `kill -9`, only `kill` with its default signal; otherwise the license server will not shut down cleanly.) On Windows systems you can use the Task Manager to identify the `lmadmin.exe` process and stop it.

You cannot restart the license server from the management interface. You must restart the license server as described in the previous section.

Installing the License Server Manager as an Operating System Service

While it is possible to manually start and stop the `lmadmin` license server manager, it is recommended that you install it as an operating system service so that it will automatically start whenever the operating system restarts.

Windows Systems

On Windows systems, you can install the `lmadmin` license server manager as a service. Only users in the Windows Administrators group can perform this action. The Startup Type is set to Automatic so that the service starts automatically when the system is restarted. Use the following license server manager (`lmadmin`) command-line options to install and uninstall the service (see the following table).

Important: *After executing the command to install the license server manager as a Windows service, it is not started automatically. You must start the service for the first time using the Windows Services Console.*

Table 2-3: Command-line options to lmadmin used to configure lmadmin as a Windows service.

lmadmin Command-line Option	Description
–installService	Creates a Windows service, with the name you defined, to run the license server manager.
–removeService	Uninstalls the Windows service with the name you defined. Make sure you stop the service before removing it.
–delay `nn`	This is used with the `-installService` option. Allows you to set the number of seconds (*nn*) to delay between the time you start the service and when it actually begins running. A typical scenario where a delay is needed is when a FLEXnet ID dongle is used to lock the license server to a machine (the FLEXid is used on the SERVER line). In this scenario the license server will sometimes fail to start upon reboot of the system because the license server is loaded before the dongle device driver has loaded properly.

Red Hat Linux and Sun Solaris Systems

On Red Hat Linux and Sun Solaris Systems, the installed `/examples` directory contains a sub-directory, `/service`. In the `/service` directory is a shell-script file, `lmadmin`. Do not confuse this `lmadmin` file with the license server manager. In the script file are installation instructions including details on where this file should be installed for either Red Hat Linux or Sun Solaris Systems. This script has been tested and will work in the majority of installations. It may need to be modified for your specific requirements.

Mac OS Systems

On Mac OS Systems, adminstrators will have to create their own startup script in a directory such as `/Library/StartupItems/LMadmin`. The installed `/examples/service/lmadmin` script is the same script as installed for Linux and Solaris systems and is provided *for reference only*; it will not work properly on Mac OS systems. For more information on installing an executable file as a system service on Mac OS systems, see any of the many publicly available references such as *`http://www.oreilly.com/pub/a/mac/2003/10/21/startup.html`*. (Please note that Acresso Software cannot be responsible for the accuracy of information obtained from such reference sources or for the startup script that you write.)

3

lmgrd - License Server Manager

The *license server manager*, is one of the components that make up a license server (the other being the vendor daemon). It handles the initial contact with FLEXenabled applications, passing the connection on to the appropriate vendor daemon. The purposes of the license server manager are to:

- Start and maintain all the vendor daemons listed in the VENDOR lines of the license file used to start `lmgrd`.
- Refer application checkout (or other) requests to the correct vendor daemon.

`lmgrd` is an application-based version of the license server manager. On most platforms it is controlled from a command-line. On Windows there is a GUI tool, `LMTOOLS`, that can be used to manage `lmgrd`.

A newer `lmgrd` can be used with an older vendor daemon or FLEXenabled application, but a newer vendor daemon or FLEXenabled application might not work properly with an older `lmgrd`. Always use the newest version of `lmgrd` as possible, which is available from the download site. See Version Compatibility between Components for detailed information.

lmgrd Command-Line Syntax

When you invoke `lmgrd`, it looks for a license file that contains information about vendors and features and starts those vendor daemons.

Usage

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

where:

Table 3-1: lmgrd Command-Line Syntax Usage

Term	Description
-c *license_file_list*	Use the specified license files.

Table 3-1: lmgrd Command-Line Syntax Usage

Term	Description
-l [+]*debug_log_path*	Write debugging information to file `debug_log_path`. This option uses the letter `l`, not the numeral 1. Prepending `debug_log_path` with the + character appends logging entries. See Debug Log File for more information on this file.
-2 -p	Restricts usage of `lmdown`, `lmreread`, and `lmremove` to a license 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` and `lmreread` commands to be run only from the same system where `lmgrd` is running.
-x lmdown	Disable the `lmdown` command (no user can run `lmdown`). If `lmdown` is disabled, 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.
-x lmremove	Disable the `lmremove` command (no user can run `lmremove`).
-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	Displays `lmgrd` version number and copyright and exits.
-help	Displays usage information and exits.

Starting the License Server Manager on UNIX Platforms

If any licenses in the license file are counted (license count > 0), the license server manager, and hence the license server, must be started before the FLEXenabled application can be used.

The license server manager, `lmgrd`, is started either manually on the command line or automatically at system startup. Both methods are discussed in the following sections.

Note: *Start lmgrd only on the system specified on the SERVER line in the license file.*

If you are running license servers configured for three-server redundancy, maintain an identical copy of the license file (as well as the lmgrd and the vendor daemons binaries) locally on each system rather than on a file server. 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.

Manual Start

Start `lmgrd` from the UNIX command line using the following syntax:

```
lmgrd -c license_file_list -L [+]debug_log_path
```

where

- `license_file_list` is one or more of the following:
  - the full path to a single license file
  - a directory, where all files named `*.lic` in that directory are used
  - `debug_log_path` is the full path to the debug log file

Prepending `debug_log_path` with the + character appends logging entries.

Start `lmgrd` by a user other than root since processes started by root can introduce security risks. If `lmgrd` must be started by the root user, use the `su` command to run `lmgrd` as a non-privileged user:

`su` *`username`* `-c "lmgrd -c` *`license_file_list`* `-l` *`debug_log_path`*"

where *`username`* is a non-privileged user. You must 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 VENDOR line.

Automatic Start

On UNIX, edit the appropriate boot script, which may be `/etc/rc.boot`, `/etc/rc.local`, `/etc/rc2.d/Sxxx`, `/sbin/rc2.d/Sxxxx`. Include commands similar to the following. See the following notes for a full explanation.

```
/bin/su daniel -c 'echo starting lmgrd > \
/home/flexlm/v11/hp700_u9/boot.log'

/bin/nohup /bin/su daniel -c 'umask 022; \
/home/flexlm/v11/hp700_u9/lmgrd -c \
/home/flexlm/v11/hp700_u9/license.dat >> \
/home/flexlm/v11/hp700_u9/boot.log'

/bin/su daniel -c 'echo sleep 5 >> \
/home/flexlm/v11/hp700_u9/boot.log'

/bin/sleep 5

/bin/su daniel -c 'echo lmdiag >>\
/home/flexlm/v11/hp700_u9/boot.log'

/bin/su daniel -c '/home/flexlm/v11/hp700_u9/lmdiag -n -c\
/home/flexlm/v11/hp700_u9/license.dat >> \
/home/flexlm/v11/hp700_u9/boot.log'

/bin/su daniel -c 'echo exiting >>\
/home/flexlm/v11/hp700_u9/boot.log'
```

Please note the following about how this script was written:

- All paths are specified in full because no paths are assumed at boot time.

- Because no paths are assumed, the vendor daemon must be in the same directory as `lmgrd`, or the VENDOR lines in the license file 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 is a security risk to run any program as root that does not require root permissions. `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. 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: *This does not start the vendor daemon until you reboot the system.*

Starting the License Server Manager on Windows

This section provides procedural information on manual starts from the command line and how to configure the License Server Manager (`lmgrd`) as a service.

Manual Start from the Command Line

To start `lmgrd` from the command line:

1. `Start lmgrd` as an application from a Windows command shell using the following syntax:

   `C:\fnp> lmgrd -c` *`license_file_list`* `-L [+]`*`debug_log_path`*

   where

   - *`license_file_list`* is one or more of the following:
     - the full path to a single license file
     - a directory, where all files named `*.lic` in that directory are used
   - *`debug_log_path`* is the full path to the debug log file

     Prepending *`debug_log_path`* with the + character appends logging entries.

Spaces in pathnames require double quotes around the path.

On Windows, `lmgrd` can be installed as a service to allow it to be started and stopped through a user interface and run in the background.

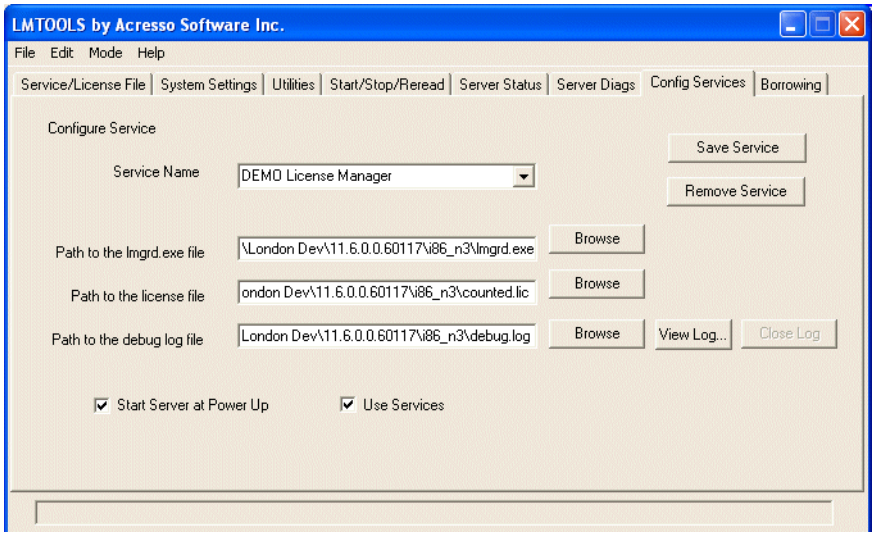
Configuring the License Server Manager as a Windows Service

To configure a license server manager (`lmgrd`) as a service, you must have Administrator privileges. The service will run under the *LocalSystem* account. This account is required to run this utility as a service.

Task ***To configure a license server as a service:***

1. Run the `lmtools` utility.
2. Click the **Configuration using Services** button, and then click the **Config Services** tab.
3. In the **Service Name**, type the name of the service that you want to define, for example, **DEMO License Manager**. If you leave this field blank, the service will be named FLEXnet Licensing Service.
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. Prepending the debug log file name with the + character appends logging entries. The default location for the debug log file is the `c:\winnt\System32` folder. To specify a different location, make sure you specify a fully qualified path.
7. To save the new **DEMO License Manager** service, click **Save Service**.

Figure 3-1: Completed Config Services Tab



Configuring the License Server Manager Service for a Delayed Start

In situations where the license server needs to wait for other drivers or services to start before it starts, you can configure a delay before the license server service starts. A typical scenario where a delay is needed is when a FLEXnet ID dongle is used to lock the license server to a machine (the FLEXid is used on the SERVER line). In this scenario the license server will sometimes fail to start upon reboot of the system because the license server is loaded before the dongle device driver has loaded properly.

Task ***To Configure a delayed start for the license server manager service:***

1. Configure the license server manager as a service (Configuring the License Server Manager as a Windows Service).
2. Locate the registry entry for your license server manager service at:

   `HKEY_LOCAL_MACHINE\SOFTWARE\FLEXlm License Manager\`*`service_name`*

   where *`service_name`* is the name of the license servermanager service.
3. Optionally, to configure a delay longer than 20 seconds, add a string value to the registry entry and set the fields in this entry as follows:

   Name - unlimitedServiceDelay

   Type - REG_SZ (set automatically when a string value is created)

   Data - no value set
4. Add a string value to the registry entry and set the fields in this entry as follows:

   Name - serviceDelay

   Type - REG_SZ (set automatically when a string value is created)

   Data - the service delay in seconds. This value is limited to the range 1-20 seconds unless unlimitedServiceDelay has previously been defined (see Step 3).

Manually Start the License Server Using the lmtools Utility

A graphical user interface to the license server manager tools is provided called `lmtools`. Some of the functions `lmtools` performs include:

- starting, stopping, and configuring license servers.
- getting system information, including hostids.
- getting server status.

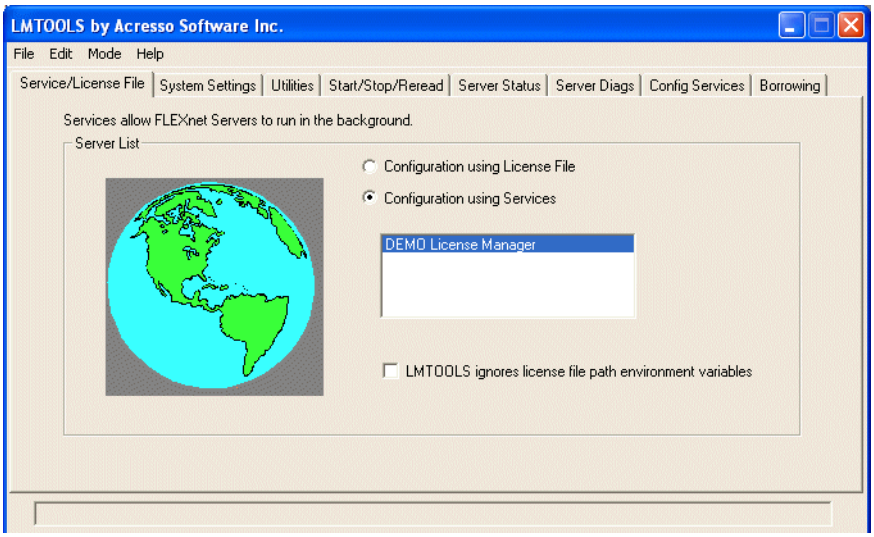
In order to control the operation of `lmgrd` from the `lmtools` user interface, you first must configure it as a license server manager service. Follow the procedure in Configuring the License Server Manager as a Windows Service before proceeding.

Once the license server manager service is configured, `lmgrd` is started by starting the service from the `lmtools` interface.

To start the service from the lmtools interface:

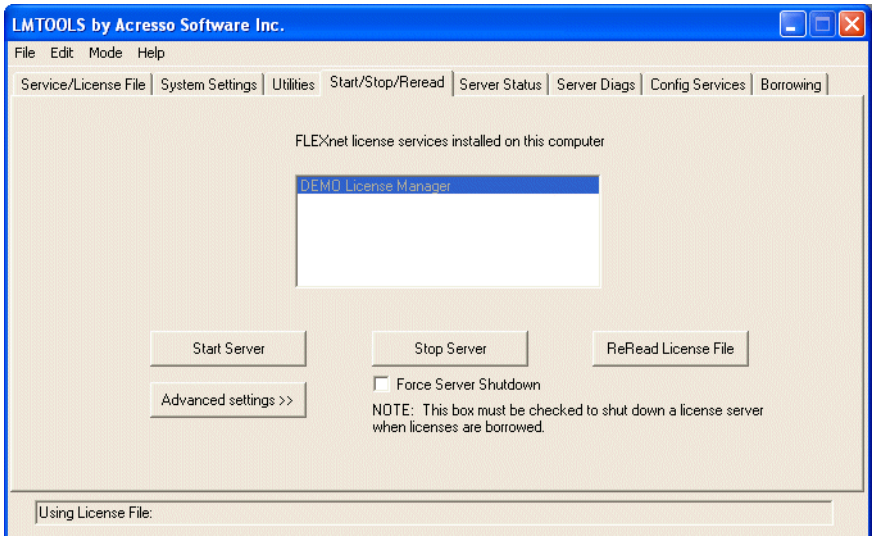
1. Start `lmtools` and display the **Service/License File** tab.
2. Click **Configuration using Services** button.
3. Select the service name from the list presented in the selection box. In this example, the service name is **DEMO License Manager**.

Figure 3-2: Service/License File Tab



4. Click the **Start/Stop/Reread** tab.
5. Start DEMO License Manager by clicking the **Start Server** button. DEMO License Manager license server starts and writes its debug log output to `c:\prods\i86_n3\debuglog`.

Figure 3-3: Start/Stop/Reread Tab



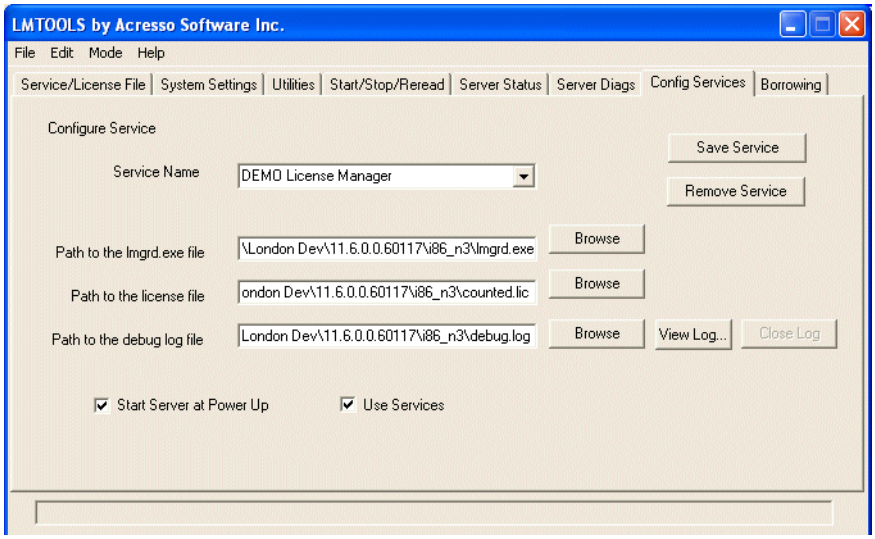
Automatically Start the License Server when System Starts

In order for `lmgrd` to start up automatically at system start-up time, you first must configure it as a service. Follow the procedure in Configuring the License Server Manager as a Windows Service before proceeding, and then continue with the steps below.

To configure lmgrd as a service:

1. With `lmtools` started and the desired service name selected, click the **Config Services** tab.

Figure 3-4: Config Services Tab



2. Make this license server manager a Windows service by selecting the **Use Services** check box.
3. Configure it to start at system startup time by selecting the **Start Server at Power Up** check box.

From now on, when the system is rebooted, this license server manager starts automatically as a Windows service.

4

Using License Administration Tools

License administration tools are available from the Acresso download site to help license administrators manage licenses and license servers. Always use the newest version of the utilities as possible. The table, License Administration Utilities, lists these utilities.

Command Line Utilities

All license server utilities are packaged as a single executable called `lmutil`. The `lmutil` is either installed as individual commands (either by creating links to the individual command names, or making copies of `lmutil` as the individual command names), or as a wrapper that runs the individual command as `lmutil command`. For example, `lmutil lmstat` or `lmutil lmdown`.

On Windows systems, the `lmutil command` form of the commands are available. There is also a graphical user interface available for these commands—see lmtools (Windows only).

Table 4-1: License Administration Utilities

Utility	Description
lmborrow	Supports license borrowing.
lmdiag	Diagnoses license checkout problems.
lmdown	Gracefully shuts down selected vendor daemons (both `lmgrd` and all vendor daemons) on the license server (or on all three systems in the case of three-server redundancy).
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.
lmpath	Allows users direct control over license file path settings.

Table 4-1: License Administration Utilities (cont.)

Utility	Description
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.
lmstat	Displays the status of a license server.
lmswitch	Controls debug log location and size.
lmswitchr	Switches the report log to a new file name.
lmver	Reports the version of a library or binary file.

- The `lmpath` utility introduced in the version 7.0 utilities.
- The `lmborrow` utility introduced in the version 8.0 utilities.
- The `lmswitch` utility introduced in version 8.0 vendor daemon.
- The `lmswitchr` utility introduced in version 5.0 vendor daemon.

Common Arguments for lmutil

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

Table 4-2: lmutil Valid Arguments

Argument	Description
-c *license_file_path*	Most `lmutil` utilities need to know the path to the license file. This is specified with a `-c license_file_path` argument, or by setting the LM_LICENSE_FILE environment variable. Otherwise, the default location is used. The utilities also honor all `VENDOR_LICENSE_FILE` environment variables. Some utilities take more than one license file path in a license search path separated by colons on UNIX and semicolons on Windows. Pathnames that include spaces must be enclosed in double quotes.
-help	Displays usage information and exits.
-v	Displays the version of the utility and exits. <DL: hidden because of
-verbose	Displays longer description for all errors found.

- `VENDOR_LICENSE_FILE` environment variable honored in utilities starting with version 7.0 utilities.
- `-verbose` option introduced in version 6.0 of the utilities.

lmborrow

`lmborrow` supports borrowing of licenses that contain the BORROW attribute. It must be run on the system where licenses are borrowed. It is used to perform the following:

- Initiating borrowing by setting the borrow period
- Clearing the borrow period
- Determining borrow status
- Returning a borrowed license early

Initiating Borrowing

To initiate borrowing, the user sets the borrow period by running `lmborrow` from the command line or through lmtools:

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

where:

Table 4-3: lmborrow Arguments for Initiating Borrowing

Argument	Description
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 FLEXenabled application's local time. If `time` is unspecified, the checkout lasts until the end of the given end date.

For example:

```
lmborrow sampled 20-aug-2007 13:00
```

This has the effect of setting `LM_BORROW` with the borrow period in either the registry (Windows) or in `$HOME/.flexlmborrow` (UNIX).

To borrow licenses for the desired vendor name, *on the same day and the same system* that the user runs `lmborrow`, run the applications to check out the licenses. If you run the applications 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 is initiated.

In addition to the `lmborrow` utility, there are other ways to initiate borrowing:

- Using the borrowing interface in application, if provided in the application.
- Setting the `LM_BORROW` environment variable directly.

See Initiating License Borrowing for more information on these other ways.

Clearing the Borrowed License Setting

Task ***To clear the LM_BORROW setting in the registry or $HOME/.flexlmborrow:***

- Issue the command `lmborrow -clear`.

Clearing the `LM_BORROW` setting stops licenses from being borrowed until borrowing is initiated again. A user might run `lmborrow -clear` after she has borrowed licenses for features that are used offline if—before disconnecting from the network—she wants to run an application that checks out additional features, served by that *`vendor name`*, that are not meant to be borrowed. Clearing `LM_BORROW` does *not* change the status for already borrowed licenses.

Determining Borrowed License Status

Task ***To print information about borrowed features:***

- Issue the following command on the system from which they are borrowed:

```
lmborrow -status
```

The borrowing system does not have to be connected to the network to determine the status.

Returning a Borrowed License Early

Task ***To return a borrowed license early:***

1. Reconnect the borrowing system back to the network.
2. From the same system that initiated the borrowing, issue the command:

```
lmborrow -return [-fqdn][-c license_file_list] [-c display] feature
```

where:

Table 4-4: lmborrow Arguments for Returning a Borrowed License Early

Argument	Description
-fqdn	Directs `lmborrow` to access the borrowing system using its fully qualified host name. Use this option if the license was borrowed based on the fully qualified host name, rather than the relative distinguished name. Use `lmstat` to determine the format of the host name used when the license was borrowed.
-c *license_file_list*	Use the specified license files. In some configurations, the license file needs to be specified in order to return the license file early.

Table 4-4: lmborrow Arguments for Returning a Borrowed License Early

Argument	Description
-d *display*	Used to specify the display from which the borrow was initiated. Required if your current display is different than what was used to initiate the borrow. On Windows, it is the system name or, in the case of a terminal server environment, the terminal server client name. On UNIX, it is in the form `/dev/ttyxx` or the X-Display name.
feature	The name of the borrowed feature to be returned early. Use `lmborrow -status` to get a list of borrowed feature names.

Returning the license early has the effect of clearing the `LM_BORROW` setting for the vendor daemon that serves the returned license.

If the borrowing system is not placed back on the network before attempting the early return, the license is not returned and `LM_BORROW` is kept intact. Additionally, an error message is issued to the user with notification that the system needs to be connected to the network.

- Early borrowed license return was introduced in version 8.3 utilities.

lmdiag

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

Usage

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

where:

Table 4-5: lmdiag Argument Usage

Argument	Description
-c *license_file_list*	Diagnose the specified files.
-n	Run in non-interactive mode; `lmdiag` does 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, select a particular line for `lmdiag` to report on. For example: lmdiag f1:HOSTID=12345678 attempts a checkout on the line with the hostid "12345678." *keyword* is one of the following: VERSION, HOSTID, EXPDATE, KEY, VENDOR_STRING, ISSUER

If no `feature` is specified, `lmdiag` operates on all features in the license files in your list. `lmdiag` first prints information about the license, then attempts to check out each license. If the checkout succeeds, `lmdiag` indicates this. If the checkout fails, `lmdiag` gives 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 TCP/IP port on the license server, and detects if the port number in the license file is incorrect. `lmdiag` indicates each TCP/IP port number that is listening, and if it is an `lmgrd` or `lmadmin` process, `lmdiag` indicates this as well. If `lmdiag` finds the vendor daemon for the feature being tested, then it indicates the correct port number for the license file to correct the problem.

See Also
FLEXLM_DIAGNOSTICS

lmdown

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

Usage

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

where:

Table 4-6: lmdown Argument Usage

Argument	Description
-c *license_file_list*	Use the specified license files. Note that specifying `-c license_file_list` is always recommended with `lmdown`.
-vendor *vendor_daemon*	Shut down only this vendor daemon. `lmgrd` continues running. Requires version 6.0 `lmdown` and `lmgrd`.
-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. `-q` is implied with `-all`.
-force	If licenses are borrowed, `lmdown` runs only from the system where the license server is running, and then only if the user adds `-force`.

If `lmdown` encounters more than one server (for example if `-c` specifies a directory with many `*.lic` files) and `-all` is not specified, a choice of license servers to shut down is presented.

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

When using the lmdown utility to shut down license servers configured for three-server redundancy, there is a one-minute delay. The `lmdown` utility shuts down all three license servers. If you need to shut down only one of these license servers (this is not recommended because you are left with two points of failure), you must shut down both the `lmgrd` and vendor daemon processes on that license server.

You can protect the unauthorized execution of `lmdown` when you start up the license server manager, `lmadmin` or `lmgrd`. Shutting down the servers causes users to lose their licenses.

See Also
lmadmin License Administration Functions
lmgrd Command-Line Syntax for details about securing access to `lmdown`
lmreread

lmhostid

The `lmhostid` utility returns the hostid of the current platform. Invoked without any arguments, `lmhostid` displays the default hostid type for current platform. Otherwise, the hostid corresponding to the requested *`type`* is displayed, if supported on the current platform.

Usage

```
lmhostid [-n] [-type] [-utf8]
```

Where:

Table 4-7: lmhostid Argument Usage

Argument	Description
-n	Only the hostid, itself, is returned as a string, which is appropriate to use with HOSTID= in the license file. Header text is suppressed.
-*type*	One of the following hostid types. If not specified, the default hostid for the current platform is displayed. See Hostids for Supported Platforms for a list of the default types. **PLATFORM-DEPENDENT HOSTIDS** • `-ether`—Ethernet address • `-string`—String id • `-vsn`—Volume serial number. (Windows platforms only) • `-flexid`—Parallel or USB FLEXnet ID dongle identification. This is applicable only for those platforms that support FLEXnet ID dongles. See Obtaining System Hostids for a complete list. • -long—32-bit hostid **PLATFORM-INDEPENDENT HOSTIDS** • -user—Current user name • -display—Current display name. On Windows, it is the system name or, in the case of a terminal server environment, the terminal server client name. On UNIX, it is in the form `/dev/ttyxx` or the X-Display name. • -hostname—Current host name • -internet—IP address of current platform in the form ###.###.###.###.

Table 4-7: lmhostid Argument Usage

Argument	Description
-utf8	The hostid is output as a UTF-8 encoded string rather than an ASCII string. If your hostid contains characters other than ASCII A through Z, a through z, or 0 through 9, use this option with lmhostid. To view a correct representation of the resulting hostid, use a utility, such as Notepad, that can display UTF-8 encoded strings.

The output of this command looks as follows:

```
lmhostid - Copyright (c) 1989-2008 Acresso Software Inc. All Rights Reserved.

The FLEXnet host ID of this machine is ""00ff5018c189 0019d244e9fc 0016cfdaf65d 001558809422
005056c00001 005056c00008""

Only use ONE from the list of hostids.
```

See Also
Hostids for Supported Platforms

lminstall

The `lminstall` utility is designed primarily for typing in decimal format licenses to generate a readable format license file.

Usage

```
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. Move this file to the application's default license file directory, if specified by the software publisher. Otherwise, use the LM_LICENSE_FILE or `VENDOR_LICENSE_FILE` environment variables to specify the directory where the `*.lic` files are located.

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 comes from stdin and prompts the user for an output file name.

`lminstall` is also used to convert licenses from readable to decimal format and between different license versions.

Task ***To convert from readable to decimal:***

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

Task ***To convert to v5.1 format:***

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

Task ***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 1,000 lines of input.

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 do not have to change the file name in the REPORTLOG line of the vendor daemon's option file. Requires a version 7.1 or later vendor daemon.

Usage

`lmnewlog [-c license_file_list] feature renamed_report_log`

or:

`lmnewlog [-c license_file_list] vendor renamed_report_log`

where:

Table 4-8: lmnewlog Argument Usage

Argument	Description
-c *license_file_list*	Use the specified license files.
feature	Any feature in this license file.
vendor	name of the vendor daemon in this license file.
renamed_report_log	New file path where existing report log information is to be moved.

lmpath

The `lmpath` utility allows direct control over license path settings. It is used to add to, override, or get the current license path settings.

Usage

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

where:

Table 4-9: lmpath Argument Usage

Argument	Description
-add	Prepends `license_file_list` to the current license search path or creates the license search path, if it doesn't exist, initializing it to `license_file_list`. Duplicates are discarded.
-override	Overrides the existing license search path with `license_file_list`. If `license_file_list` is the null string, `""`, the specified list is deleted. • `lmpath -override all ""`—Deletes the value of `LM_LICENSE_FILE`. • `lmpath -override vendor ""`—Deletes the value of `VENDOR_LICENSE_FILE`.
vendor	A vendor daemon name. Affects the value of `VENDOR_LICENSE_FILE`.
all	Refers to all vendor daemons. Affects the value of `LM_LICENSE_FILE`.
license_file_list	On UNIX, separate values with a colon. On Windows, separate values with a semicolon. If `license_file_list` is the null string, `""`, then the specified entry is deleted.

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

Task ***To display the current license path settings:***

```
lmpath -status
```

The following is displayed:

```
lmpath - Copyright (C) 1989-2008 Acresso 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.

lmremove

The `lmremove` utility allows you to remove a single user's license for a specified feature. If the application is active, it rechecks out the license shortly after it is freed by `lmremove`.

Usage

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

or

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

where:

Table 4-10: lmremove Argument Usage

Argument	Description
-c *license_file_list*	Specify license files.
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	TCP/IP 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 files are used as the license file.

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 TCP/IP port is **7654**, and the license handle is **102**.

Task ***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 are also removed. If license lingering is set and `lmremove` is used to reclaim the license, `lmremove` starts, but does not override, the license's linger time.

You can protect the unauthorized execution of `lmremove` when you start up the license server manager, `lmadmin` or `lmgrd`, because removing a user's license is disruptive.

See Also
Imadmin License Administration Functions
Imgrd Command-Line Syntax for details about securing access to `lmremove`

lmreread

The `lmreread` utility causes the license server manager to reread the license file and start any new vendor daemons that have been added. In addition, all currently running vendor daemons are signaled to reread the license file and their options files for changes. If report logging is enabled, any report log data still in the vendor daemon's internal data buffer is flushed. `lmreread` recognizes changes to system host names, but cannot be used to change server TCP/IP port numbers.

If the optional vendor daemon name is specified, only the named daemon rereads the license file and its options file (in this case, `lmadmin` or `lmgrd` does not reread the license file).

Usage

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

where:

Table 4-11: lmreread Argument Usage

Argument	Description
-c *license_file_list*	Use the specified license files.
-vendor *vendor*	Only the vendor daemon, specified by the *vendor* option, rereads the license file and the options file. Additionally, `lmgrd` restarts *vendor* if necessary.
-all	If more than one `lmgrd` is specified, instructs all `lmgrds` to reread.

Note: *If you use the* `-c license_file_list` *option, the license files specified are read by* `lmreread`*, not by* `lmgrd`*;* `lmgrd` *rereads the file it read originally.*

You can protect the unauthorized execution of `lmreread` when you start up the license server manager, `lmgrd`.

See Also
Imadmin License Administration Functions for GUI-based functions available in lmadmin
Imgrd Command-Line Syntax for details about securing access to `lmreread`
Ability for vendor daemon to participate in rereading of its option file introduced in version 8.0 vendor daemon

lmstat

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

- Daemons that are running
- License files

- Users of individual features
- Users of features served by a specific vendor daemon
- BORROW licenses borrowed

The `lmstat` utility prints information that it receives from the license server; therefore, it does not report on unserved licenses such as uncounted licenses. To report on an uncounted license, the license must be added to a served license file and the application must be directed to use the license server for that 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

```
lmstat [-a] [-c license_file_list] [-f [feature]] [-i [feature]] [-s[server]]
              [-S [vendor]] [-t timeout_value]
```

where:

Table 4-12: lmstat Argument Usage

Argument	Description
-a	Displays all information. This option is a potentially expensive command. With many active users, this command option generates a lot of network activity.
-c *license_file_list*	Uses the specified license files.
-f [*feature*]	Displays users of *feature*. If *feature* is not specified, usage information for all features is displayed.
-i [*feature*]	Displays information from the feature definition line for the specified *feature*, or all features if *feature* is not specified.
-s [*server*]	Displays status of all license files listed in `$VENDOR_LICENSE_FILE` or `$LM_LICENSE_FILE` on *server*, or on all servers if *server* is not specified.
-S [*vendor*]	Lists all users of *vendor*'s features.
-t *timeout_value*	Sets connection timeout to *timeout_value*. This limits the amount of time `lmstat` spends attempting to connect to *server*.

The output of `lmstat -a` looks similar to:

```
lmstat - Copyright (c) 1989-2008 Acresso Software Inc. All Rights Reserved.

Flexible License Manager status on Wed 11/28/2007 14:49

[Detecting lmgrd processes...]

License server status: 27000@prod

    License file(s) on prod: C:\prod\i86_n3\counted.lic:

prod: license server UP v11.5
Feature usage info:
Users of f1:  (Total of 4 licenses issued;  Total of 1 license in use)
  "f1" v1.0, vendor: demo
  floating license
```

```
daniel myhost2 19.36.18.26 (v1.0) (myhost1/27000 102), start Fri
     5/3 7:29
```

where:

Table 4-13: lmstat Output

Output	Argument	Description
daniel	*user*	User name.
myhost2	*user_host*	Host where user is running.
19.36.18.26	*display*	Display where user is running.
v1.0	*version*	Version of feature.
myhost1	*server_host*	Host where license server is running.
27000	*port*	TCP/IP 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.

The *user*, *user_host*, *display*, *server_host*, *port*, and *handle* information is used when removing licenses with lmremove.

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 also starts a new debug log file written by that vendor daemon if one does not already exist.

Usage

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

where:

Table 4-14: lmswitch Argument Usage

Argument	Description
-c *license_file_list*	Use the specified license files.
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 vendor daemons 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`*.

Note: *The effect of `lmswitch` continues only until the vendor daemon is shut down or its options file is reread via `lmreread`. When the vendor daemon is restarted or its options file is reread, it looks 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:
lmadmin License Administration Functions for information on lmadmin display
DEBUGLOG
lmreread
Debug Log File

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 also starts a new report log file if one does not already exist.

Usage

`lmswitchr [-c license_file_list] feature new_report_log`

or with version 5.0 or later vendor daemon:

`lmswitchr [-c license_file_list] vendor new_report_log`

where:

Table 4-15: lmswitchr Argument Usage

Argument	Description
-c *license_file_list*	Use the specified license files.
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`*.

Note: *The effect of `lmswitchr` continues only until the vendor daemon is shut down or its options file is reread via `lmreread`. When the vendor daemon is restarted or its options file is reread, it looks 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:
REPORTLOG
lmnewlog
lmreread
Report Log File

lmver

The `lmver` utility reports the version of a FLEXnet Publisher Licensing Toolkit library or binary file.

Usage

`lmver` *`filename`*

where *`filename`* is one of the following:

- the name of an executable file built with FLEXnet Publisher Licensing Toolkit
- lmgrd
- a license administration tool
- a vendor daemon

For example, if you have an application called **spell**, type `lmver spell`.

lmtools (Windows only)

The lmtools utility is a graphical user interface that allows you to administer the license server. This executable is available in the 32-bit and 64-bit Windows packages. Always use the newest version possible. You can get it from the software download site.

Some of the functions this utility performs include:

- starting, stopping, and configuring license servers
- getting system information, including hostids
- getting server status

The lmtools utility has two modes in which to configure a license server:

- Configuration using a license file
- Configuration using services

On Windows Vista, you must run the lmtools utility as an administrator. If you do not run this executable as an administrator, the User Account Control (UAC) dialog will display as soon as it is started (as long as the UAC prompt is not disabled on the system).

Configuration Using License File

Operations are performed on a particular license file. The file can be either local or remote. In this mode, you cannot start the `lmgrd` process, but you can do everything else.

To configure this mode:

1. Run the lmtools utility.
2. Click the **Configuration using License File** button.
3. Enter one or more the license file names or *`port@host`* specifications.

Configuration Using Services

Operations are performed on a service, which allows starting `lmgrd` processes local to the system on which lmtools is running. For details on configuring services, see Configuring the License Server Manager as a Windows Service.

5

IPv6 Support

Internet Protocol version 6 (IPv6) is the next generation IP protocol. This section contains information for license administrators who support have networks that support IPv6 addresses. The information in this section assumes the reader has a familiarity with the IPv6 networking protocols. The following sections of this chapter describe the FLEXnet Publisher Licensing Toolkit support for IPv6.

Capabilities that Support IPv6

This section describes the capabilities in the FLEXnet Publisher Licensing toolkit configurable by license administrators that support IPv6. This section describes components used with both license file-based licensing and trusted storage-based licensing.

When working with a software publisher to obtain a software package that supports IPv6, you should collect and provide the IP addresses of systems (FLEXenabled clients and license servers) that will be used in the license file.

License File

In a license file, the SERVER line can define an IPv6 address as the host value.

Options File

An options file can contain an IPv6 address to specify host restrictions when using the:

- INTERNET type in these keywords - EXCLUDE, EXCLUDEALL, EXCLUDE_BORROW, INCLUDE, INCLUDEALL, INCLUDE_BORROW, MAX, and RESERVE.
- HOST type in these keywords - EXCLUDE, EXCLUDE_ENTITLEMENT, EXCLUDEALL, EXCLUDE_BORROW, INCLUDE, INCLUDE_ENTITLEMENT, INCLUDEALL, INCLUDE_BORROW, MAX, and RESERVE
- HOST_GROUP keyword (it takes IP addresses).

License Search Path

Entries in the license search path that use the 'port@host' convention to identify the license server, can specify an IPv6 address as the 'host' value.

Deploying License Servers in Mixed Protocol Environments

For FLEXnet Publisher Licensing Toolkit components to work properly using IPv6 addresses, all systems in an enterprise (including the network hardware and software) must be configure properly to support communication using IPv6 addresses. Before testing or deploying a FLEXenabled application that supports IPv6 or IPv4/IPv6 dual communication, make sure that all systems on the network can communicate successfully.

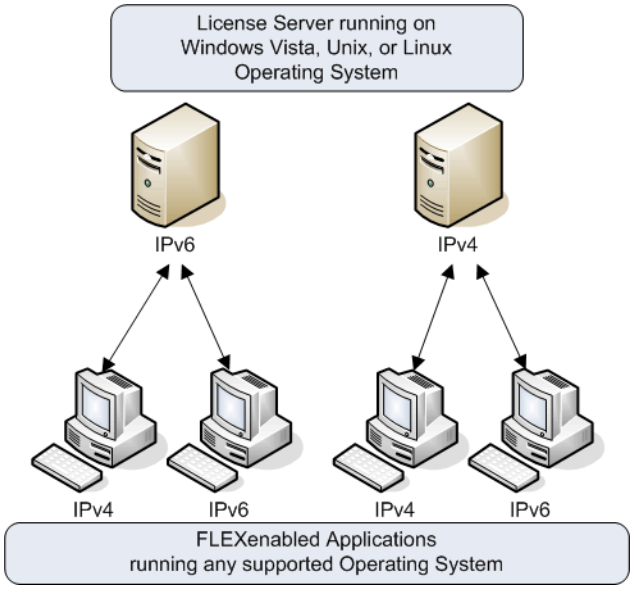
If the license server will run under any of the following operating systems, it can communicate with FLEXenabled clients using either IPv4 or IPv6 (as long as the network is configured properly).

- Any supported edition of Windows Vista
- Any supported Linux platform
- Any supported Unix platform

Because these operating systems support dual-layer communication, both IPv4 and IPv6 FLEXenabled clients can communicate with an IPv6 license server. In addition, IPv6 clients can communicate with an IPv4 license server using the IPv4 address. Figure 5-1 illustrates this behavior.

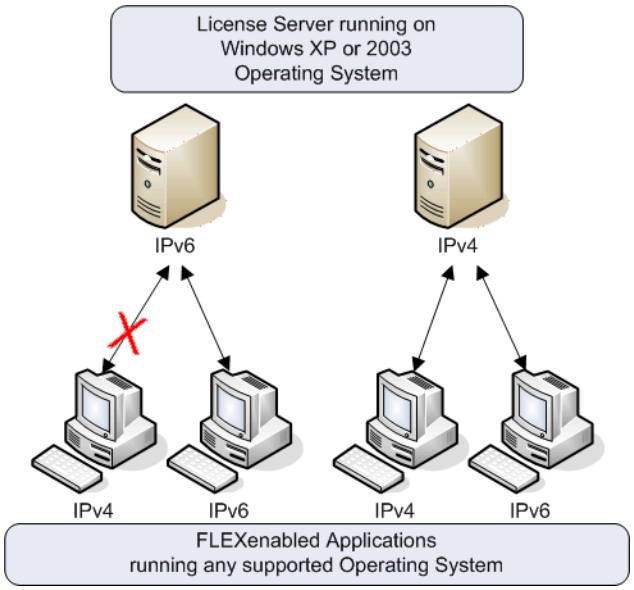
If you are using `lmadmin` as your license server, it supports both IPv4 and IPv6 clients. You must rename one of your vendor daemon executable files, because separate IPv4 and IPv6 vendor daemons are required.

Figure 5-1: License Server Running on Windows Vista, Unix, or Linux



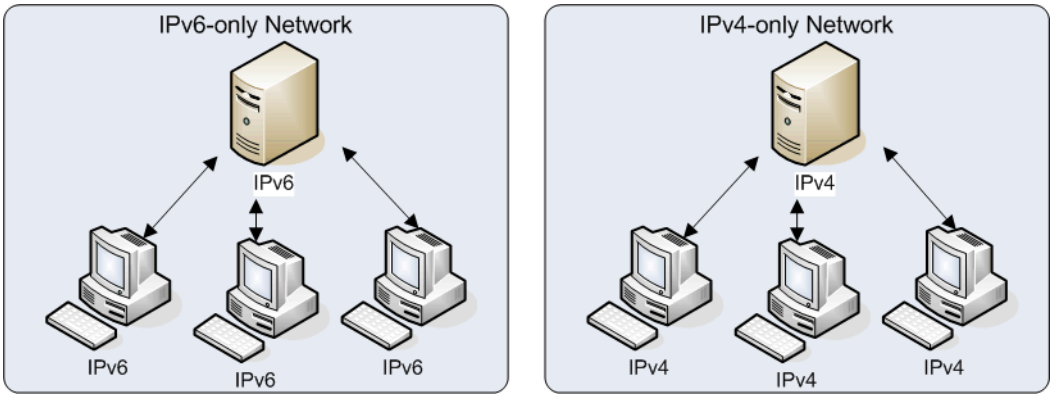
If the license server runs on Windows XP or Windows Server 2003, there are certain limitations because of the limited dual-layer support on these operating systems. IPv4 FLEXenabled clients **cannot** communicate with a IPv6 license server running on these operating systems. However, IPv6 FLEXenabled clients **can** communicate with an IPv4 license server running on these operating systems. Figure 5-2 illustrates this behavior.

Figure 5-2: License Server running on Windows 2003 or XP



If an enterprise runs license servers on Windows 2003 or Windows XP the license administrators should create and maintain two separate networks - one for IPv6 FLEXenabled clients (that will use the IPv6 license server) and the other for IPv4 FLEXenabled clients (that uses the IPv4 license server). The following figure illustrates this configuration.

Figure 5-3: Separate IPv4 and IPv6 Environments



Using Wildcards in an IPv6 Address

The wildcard character, "*," may be used in place of an entire field or on a byte-by-byte basis to specify a range of addresses without having to list them all. For example, this example feature definition line is locked to four specific addresses:

```
FEATURE f1 myvendor 1.0 1-jan-2008 uncounted \
     HOSTID="INTERNET=127.17.0.1,\
        INTERNET=2001:0db8:0000:0000:ff8f:effa:13da:0001,\
        INTERNET=127.17.0.4,\
        INTERNET=2001:0db8:0000:0000:ff8f:effa:13da:0004" \
     SIGN="<...>"
```

The following example feature definition line specifies an entire range of addresses, including the four specific ones from the line above:

```
FEATURE f1 myvendor 1.0 1-jan-2008 uncounted \
     HOSTID="INTERNET=127.17.0.*,\
        INTERNET=2001:0db8:0000:0000:*:*:*:000*"\
     SIGN="<...>"
```

6

Using Three-Server Redundancy

This section describes how to configure license servers in a three-server redundancy configuration.

License administrators can implement failover protection for license servers using either of the following methods:

- Three-server redundancy: configure and maintain a set of three license server systems configured specifically for three-server redundancy. This provides failover protection only. License administrators manage only one version of the license file and vendor daemon on all three license servers.
- Redundancy using the license search path: configure and maintain multiple independent license servers, each with a subset of the total licenses available to the enterprise. Configure the FLEXenabled client with the license servers in the license search path. This provides load balancing capabilities and limited failover protection. License administrators must manage different versions of the license rights on each license server.

License administrators should work with their software publishers to enable and configure three-server redundancy for failover protection. Three-server redundancy is a specific capability available in FLEXnet Publisher Licensing Toolkit that provides failover protection while preventing license rights from being improperly replicated. In this configuration, only one license server supplies licenses to FLEXenabled applications.

When at least two of the three license servers are running and communicating, the system serves licenses to FLEXenabled applications. Three-server redundancy is supported with license file-based licensing only. It is not supported with trusted storage-based licensing.

Overview of Three-Server Redundancy

Using the three-server redundancy capability in FLEXnet Publisher Licensing Toolkit, all three license servers operate to form a triad. The license servers send periodic messages to each other to make sure that at least two servers are running and communicating. A quorum is formed when at least two of the three license servers are running and communicating with each other.

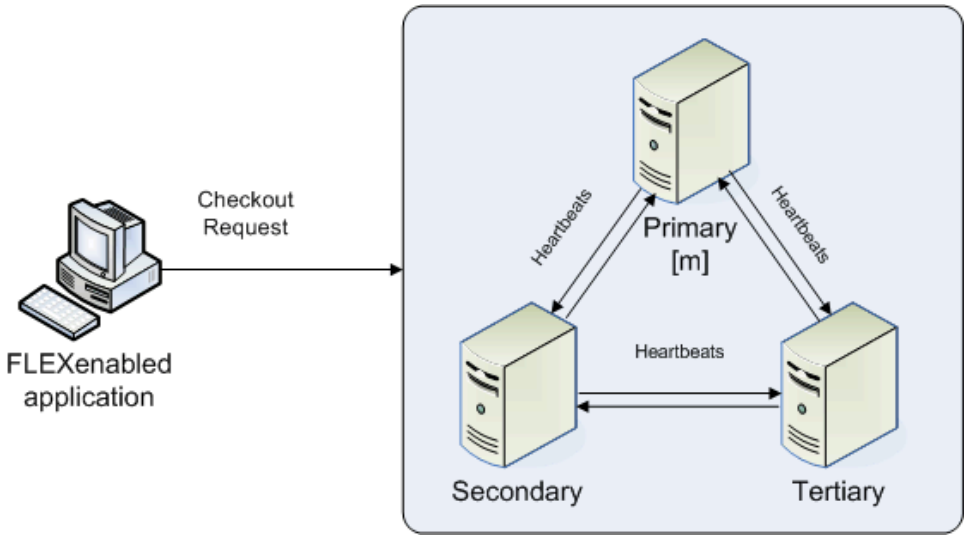
The license servers are identified as either primary, secondary, or tertiary. One license server is also designated as the master [m] and is responsible for:

- serving licenses to FLEXenabled applications
- recording information into the debug log.
- recording information into the report log.

If the master fails, then another license server becomes the master.

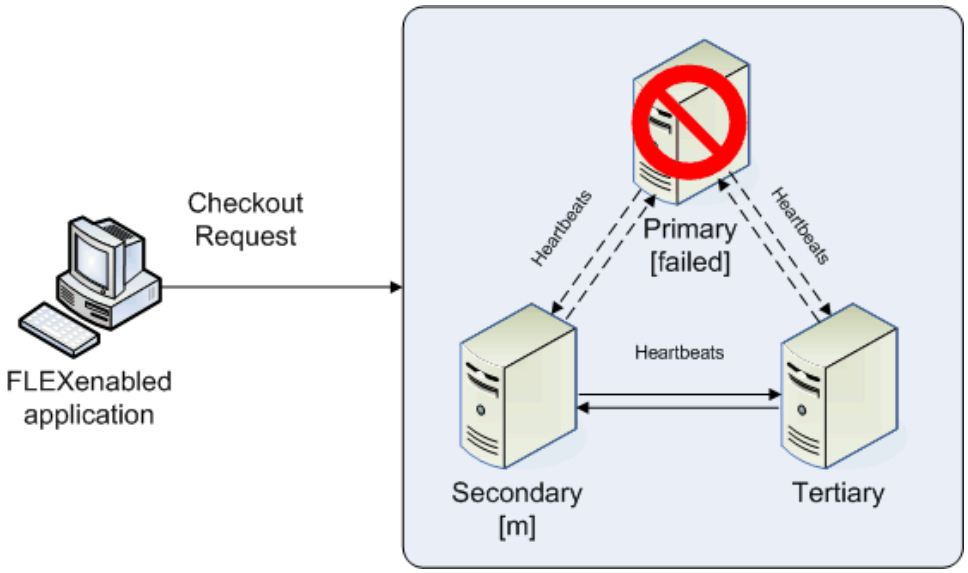
In the following figure, the primary license server is the master [m]. When a FLEXenabled application sends a checkout request for a license, the master responds and then serves the license to the FLEXenabled application.

Figure 6-1: Three-Server Redundancy Overview



If the master fails, then the secondary license server becomes the master (see the following figure) and will server licenses to FLEXenabled applications. The tertiary license server can never be the master. If both the primary and secondary license servers go down, licenses are no longer served to FLEXenabled applications. The master will not serve licenses unless there are at least two license servers in the triad running and communicating.

Figure 6-2: Three-Server Redundancy Backup Failover



Understanding How License Servers Communicate

When started, each license server reads the license file and checks that it can communicate with the other license servers. Until each license server establishes this first connection with the others, it will continue to send messages periodically.

Once the initial communication has been established, each license server periodically sends a heartbeat to the others. Heartbeats are messages sent over TCP/IP. Each license server sends a heartbeat and waits for a response from the other license servers. If a license server does not receive a response, it shuts down the vendor daemon so that it cannot serve licenses. A publisher or license administrator can configure the amount of time a license server waits to receive a heartbeat using the HEARTBEAT_INTERVAL property.

Poor network communication causes system performance to slow. Slow network communication can also cause a delay in the transmission of heartbeats between license servers.

Managing License Servers in this Configuration

Using the lmstat Utility

The output message generated by the lmstat utility identifies which license server is the master. In the following example lmstat output, the secondary license server is the master.

```
[Detecting lmgrd processes...]
License server status: 30000@RMD-PRIMARY,30000@RMD-SECONDARY,
30000@RMD-TERTIARY
```

```
License file(s) on RMD-PRIMARY: C:\server\3.lic:
RMD-PRIMARY: license server UP v11.4
RMD-SECONDARY: license server UP (MASTER) v11.4
RMD-TERTIARY: license server UP v11.4
```

Starting and Stopping License Servers

To start the entire system, you must start each license server manager (`lmadmin` or `lmgrd`). Generally, it is good practice to start the primary license server before the secondary or tertiary license server. This allows the primary license server to become the master before the others start. If you start the secondary and tertiary before the primary, then the secondary will establish itself as master.

If you do not set the PRIMARY_IS_MASTER keyword for the primary license server, then the order in which you start the license servers is important. If you do not set this property, when you start the primary license server after the secondary license server control will not transfer to the primary license server. By setting the PRIMARY_IS_MASTER keyword, you ensure that when the primary license server is running, it is always the master.

The lmdown utility will shut down all three license servers using a single command. You do not have to shut down each license server separately.

Running the License Server Manager (lmgrd) as a Service on Windows

There are no dependencies or known issues related to running the license server manager (`lmgrd` executable) as a service in this configuration.

Logging and the Debug Log

When using three-server redundancy, the master records information to its local debug log and report log (and the Windows event log if this is configured). If this system fails, another license server becomes the master and records information to its local debug log and report log. Subsequently, there may be different versions of the debug log and report log on the primary and secondary license server which each contains different information.

Configuring License Servers for Three-Server Redundancy

Both the software publisher and the license administrator must perform certain configuration steps. This section describes the steps that each must perform.

Configuration for License Administrators

The license administrators should perform the following steps:

1. Before the license administrator gets the license server software package, they should identify and set up the three systems. When selecting systems, make sure they are stable. Do not use systems that are frequently rebooted or shut down.
2. Send the publisher the hostname and hostid values for these systems. Ask the publisher what system identifier they need for the hostid. This could be an Ethernet address, disk serial number, etc. The publisher will create license server components specifically for these systems.

3. After receiving the license server package from the publisher, change the following SERVER line properties in the license file if necessary:
   - **port number** the license servers uses to listen for communication
   - **PRIMARY_IS_MASTER** keyword
   - **HEARTBEAT_INTERVAL** property

Do not change the hostid values. If the hostid changes at any time, the license administrator must work with the software publisher to obtain a new license file.

4. Perform any additional configuration as required by the software publisher.
5. Copy or install the license server software package to each of the three systems.
6. Start the license servers in the following order: primary, secondary, and then tertiary.

An Example License File

The following is an example of a license file that is configured for three-server redundancy.

```
SERVER pat 17003456 2837 PRIMARY_IS_MASTER
SERVER lee 17004355 2837
SERVER terry 17007ea8 2837
VENDOR demo
FEATURE f1 demo 1.0 1-jan-2018 10 SIGN="<...>"
FEATURE f2 demo 1.0 1-jan-2018 10 SIGN="<...>"
```

The following portions of the license file directly affect the three-server redundant configuration:

- SERVER lines: These three lines define each of the systems involved.
  - The **host** values: they are: pat, lee, and terry.
  - The **hostid** values: they are: 17003456, 17004355, and 17007ea8. This example uses the value returned by the lmhostid utility default hostid type. The default hostid type is different for every platform.
- The TCP/IP **ports**: All servers use the same port (2837, in the example) to listen for communication.

The following properties of the license file do not affect the three-server redundant configuration directly, but are used to define license rights or configure the license server.

- VENDOR line: this is required and references the publisher's vendor daemon.
- FEATURE lines: The two features, f1 and f2, define the license rights. The SIGN value for each FEATURE line encodes the license server hostid values.

Using Other Capabilities with Three-Server Redundancy

The following section describe other capabilities available in FLEXnet Publisher Licensing Toolkit and how they interact with three-server redundancy.

Configuring the License Search Path

This configuration can be performed by either the software publisher or the license administrator. Before a FLEXenabled application can check out a license, it must know where to locate the license rights. The license search path identifies the location of license rights.

When connecting to a license server configured for three-server redundancy, the FLEXenabled application must use the port@host convention (and not a license file location) in the license search path.

The license search path should list the license servers in the same order that they appear in the license file. This helps shorten the amount of time it takes to identify the master server and respond to the checkout request. Although the configuration will work if you include only one of the license servers in the license search path, this may lengthen the amount of time it takes for the license server to respond to the checkout request. This is because the license server must identify all other license servers and designate a master.

You must also separate each port@host entry with a comma and not a semicolon (Windows), colon (Unix/Mac), or ampersand (Java). The comma indicates that the license servers are configured for three-server redundancy.

Using the previous license file as an example, the license search path should be:

2837@pat,2837@lee,2837@terry

The FLEXenabled application will try to connect to each of the license servers in the list, in the order listed, until it either successfully connects to a license server or reaches the end of the list. This helps ensure that the FLEXenabled application can connect to the quorum.

Using License File Keywords

The following keywords and properties for the **SERVER** line allow you to modify the configuration**.**

- **Host:** this is the hostname of the system. The publisher should know this information when generating the license file. This value can be changed after the license file has been signed.
- **Port**: the port number that the license server uses to listen for communication. Unlike single license servers, each SERVER line must include a port number. This can be any number between 1024 and 64000 that is not used by another process running on the system. This value can be changed after the license file has been signed. If you are using `lmadmin`, you do not need to edit the license file: you can configure the port number using the GUI interface. See on-line help for more details.

  To make it easier to administer the license server, we strongly recommended that you define the same port number for each SERVER line. This value can be changed after the license file has been signed.
- **PRIMARY_IS_MASTER**: this keyword ensures that the primary server is the master whenever it is running and communicating with one of the other license servers.
  - If this is set and the primary server goes down, when the primary server comes back up again, it will always become the master.
  - If this is not set and the primary server goes down, the secondary server becomes the master and remains the master even when the primary server comes back up. The primary can only become the master again when the secondary license server fails.

This parameter is optional and should be placed on the first SERVER line. This value can be changed after the license file has been signed. The license server must be running a version 10.8 or later vendor daemon to use this keyword.

- **HEARTBEAT_INTERVAL=seconds:** this indicates how long the license servers wait to receive a heartbeat from another license server before shutting down the vendor daemon. This value is used in the following equation to calculate the actual timeout value:

  ```
  timeout = (3 * seconds) + (seconds – 1)
  ```

  The default value is 20, which equates to an actual timeout of 79 seconds. Valid values are 0 through 120. This parameter is optional and should be placed on the first SERVER line in the license file. This value can be changed after the license file has been signed. The license server must be running a version 10.8 or later vendor daemon to use this keyword.

Using Options File Keywords

None of the keywords in the options file affect three-server redundancy.

Troubleshooting Tips and Limitations

Separating the Contents of a License File

Because the hostid values in the SERVER lines are computed into the signature of each feature definition line, make sure you keep SERVER lines together with any feature definition lines as they were generated. This means that if you move a feature definition line to another file, you must also move the respective SERVER lines and VENDOR line.

Putting the License File on a Network File Server

Do not put the license file on a network file server. If you do this, you lose the advantages of having failover protection because the file server becomes a possible single point of failure.

Using License Servers in Heavy Network Traffic

On a network with excessive traffic, the license servers may miss heartbeats which causes them shut down the vendor daemon. The master may then stop serving licenses. If you find that heavy network traffic causes this to occur, you should set the HEARTBEAT_INTERVAL to a larger value. Enterprises can experience a performance issue when there is slow network communication or if FLEXenabled clients are using a dialup link to connect to the network.

Using Multiple Vendor Daemons

The license server manager (`lmadmin` or `lmgrd`) can not start vendor daemons from other software publishers when configured for three-server redundancy. The license server manager can only manage one vendor daemon. If one of the systems runs more that one vendor daemon, then the license administrator must run separate instances of the license server on that system to support the other vendor daemons. Make sure the port numbers do not clash.

7

Reading a License File

The license file contains information required to manage licenses for a FLEXenabled application. This information includes:

- License server names and hostids
- VENDOR names and paths to vendor daemon executables
- Feature information

The license file must be accessible to systems that run the FLEXenabled application or a license server.

Specifying the Location of the License File

Software publishers often recommend a specific location for your license file. You have the following options for making your licenses available to all systems:

- Place the license file in a partition which is available to all systems in the network that need it.
- Copy the license file to each of the individual systems.
- Set the LM_LICENSE_FILE or *VENDOR*_LICENSE_FILE (where *VENDOR* is the vendor daemon name) environment variable to *port*@*host*, where *host* and *port* come from the SERVER line in the license file. Alternatively, if the license file SERVER line specifies a TCP/IP port in the default port range (27000–27009) or does not specify a port (thereby allowing the license server manager to choose one from the default range), use the shortcut specification, @*host*.

  For license servers configured for three-server redundancy, use a comma separated list of three *port*@*host* specifiers to identify the three license servers. For example,

  ```
  port1@host1,port2@host2,port3@host3
  ```

Table 7-1 shows some examples of `LM_LICENSE_FILE` and *`VENDOR`*`_LICENSE_FILE` environment variable settings.

Table 7-1: Environment Variable Specification Examples

SERVER Line	LM_LICENSE_FILE or *VENDOR*_LICENSE_FILE Setting
SERVER myserver 17007ea8 40000 where: • host = myserver • port = 40000	40000@myserver
SERVER myserver 17007ea8 27001 where: • host = myserver • port = 27001, within the default range	@myserver
SERVER myserver 17007ea8 where: • host = myserver • port = none specified, uses a default TCP/IP port number in the range of 27000-27009	@myserver

- On Windows, if the application cannot find the license file, the user is presented with a dialog that asks the user to specify the license file location, the license server, or license fulfillment from the internet.

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 copy `lmadmin` or `lmgrd` and the vendor daemon to a location that the FLEXenabled application can access on the network.

Setting the License Search Path using an Environment Variable

Most applications specify a location where they expect to find the license file and install it automatically. However, you can change the license file location by setting the LM_LICENSE_FILE environment variable to a `license_file_list`. Wherever `license_file_list` is specified, it can consist of the following components:

- the full path to the license file
- a directory containing one or more license files with a **.lic** extension
- a *`port@host`* setting, where *`port`* and *`host`* are the TCP/IP port number and host name from the SERVER line in the license file. Alternatively, use the shortcut specification, *`@host`*, if the license file SERVER line uses a default TCP/IP port or specifies a port in the default port range (27000–27009).
- A comma separated list of three *`port@host`* specifiers identifying the license servers configured for three-server redundancy. For example,

```
port1@host1,port2@host2,port3@host3
```

Applications accept an environment variable (or Windows Registry) named VENDOR_LICENSE_FILE, where VENDOR is the vendor daemon name, for example, DEMO_LICENSE_FILE. This environment variable's scope is limited to just those applications from software publisher using the VENDOR name. On UNIX, the license search path entries are separated by colons; on Windows, the entries are separated by semicolons.

With lmgrd and lmutil (lmstat, lmdown, and so on), the -c option overrides the setting of the LM_LICENSE_FILE environment variable.

Note: *Some applications do not recognize the LM_LICENSE_FILE environment variable. FLEXenabled Java applications, in particular, do not recognize it.*

See Also
Managing Multiple License Files for more information about LM_LICENSE_FILE.
Environment Variables

License File Format Overview

License files begins with either a single SERVER line or three SERVER lines (when configured for three-server redundancy) followed by one or more VENDOR lines, followed by one or more FEATURE or INCREMENT lines. In some cases, the license file requires no SERVER line and no VENDOR line.

Please note that eight-bit Latin-based characters are fully supported in license files, options files, log files, and FLEXenabled application environments.

See Counted vs. Uncounted Licenses for more information on SERVER and VENDOR line requirements.

You can modify these elements in the license file:

- On the SERVER line

  Host names on the SERVER lines

  TCP/IP port numbers

  HEARTBEAT_INTERVAL and PRIMARY_IS_MASTER properties

- On the VENDOR line
  - Paths to the vendor daemon.
  - Options file paths
  - TCP/IP port numbers (for firewall support only)
- The USE_SERVER line.
- On the feature definition lines,
  - The values in *keyword=value* pairs on FEATURE lines, if *keyword* is specified in lowercase
  - You can use the \ line-continuation character to break up long lines.

See Also
Using Three-Server Redundancy
Counted vs. Uncounted Licenses

License File Syntax

This section describes the contents of the license file, including SERVER lines and VENDOR lines. This is an example of a license file for a single VENDOR name with two features.

```
SERVER my_server 17007ea8 1700
VENDOR sampled
FEATURE f1 sampled 1.000 01-jan-2008 10 SIGN="<...>"
FEATURE f2 sampled 1.000 01-jan-2008 10 SIGN="<...>"
```

This example allows the license server, called **my_server** with the hostid **17007ea8**, to serve ten floating licenses for each feature, **f1** and **f2** to any user on the network.

SERVER Lines

The SERVER line specifies the host name and hostid of the license server and the TCP/IP port number of the license server manager (`lmadmin` or `lmgrd`). Normally a license file has one SERVER line. Three SERVER lines mean that you are using license servers configured for three-server redundancy. The absence of a SERVER line means that every feature definition line in the license file is uncounted.

The hostids from the SERVER lines are computed into the license key or signature on every feature definition line. For this reason, make sure you keep SERVER lines together with any feature definition lines as they were sent from the software publisher.

The format of the SERVER line is:

`SERVER` *`host hostid`* `[`*`port`*`] [PRIMARY_IS_MASTER] [HEARTBEAT_INTERVAL=`*`seconds`*`]`

For example:

```
SERVER my_server 17007ea8 21987
```

The following table describes the attributes on this line.

Table 7-2: SERVER Line Format

Field	Description
host	The system host name or IP address. String returned by the UNIX `hostname` or `uname -n` command. On NT/2000/XP, `ipconfig /all`; on Windows 95/98/ME, `winipcfg /all` return the host name.
hostid	Usually the string returned by the `lmhostid` command. This is changed only by your software supplier.

Table 7-2: SERVER Line Format

Field	Description
port	TCP/IP 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. If no TCP/IP port number is specified, one of the default ports in the range of 27000–27009 is used. You must specify a port number when the SERVER line define license servers configured for three-server redundancy.
PRIMARY_IS_MASTER	Used with license servers configured for three-server redundancy to indicate how master control is transferred between the primary and secondary servers. • If this is set and the primary server goes down, when the primary server comes back up again, it will always become the master. • If this is not set and the primary server goes down, the secondary server becomes the master and remains the master even when the primary server comes back up. The primary can only become the master again when the secondary license server fails. If both primary and secondary go down, licenses are no longer served. The tertiary server never becomes the master. This parameter is optional and is placed on the first SERVER line in the license file. You must be running a version 10.8 or later vendor daemon to use this parameter.
HEARTBEAT_INTERVAL = *seconds*	Used with license servers configured for three-server redundancy to indicate how long a license server waits to receive a heartbeat from another license server in the triad before shutting itself down. The `seconds` value is used in the following equation to calculate the timeout: • timeout = (3 x *seconds*) + (*seconds* - 1) If not specified, the default value for `seconds` is 20, equating to an actual timeout value of 79 seconds. Valid values for the `seconds` value are 0–120. This parameter is optional and is placed on the first SERVER line in the license file. You must be running a version 10.8 or later vendor daemon to use this parameter.

See Also
Using Three-Server Redundancy

VENDOR Lines

The VENDOR 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 VENDOR line is shown below.

```
VENDOR vendor [vendor_daemon_path]\
              [[OPTIONS=]options_file_path] [[PORT=]port]
```

where:

Table 7-3: VENDOR Line Format

Field	Description
vendor	Name of the vendor daemon used to serve some features in the file. This name cannot be changed.
vendor_daemon_path	Optional path to the executable for this daemon. Generally, the license administrator is free to install the vendor daemon in any directory. It is recommended, however, that it be installed in a local directory on the license server. If omitted, `lmgrd` looks for the vendor daemon binary in: • the current directory • the path specified in `lmgrd`'s `$PATH` environment variable • in the directory where `lmgrd` is located If *`vendor_daemon_path`* is blank, then any options or TCP/IP port number specifications require the `OPTIONS=` and `PORT=` strings.
options_file_path	Full path to the options file for this daemon. An options file is not required. If omitted, the vendor daemon, by default, looks for a file called *`vendor`*`.opt` (where *`vendor`* is the vendor daemon name) located in the same directory as the license file.
port	Vendor daemon TCP/IP port number. The default, if *`port`* is not specified, is chosen by the operating system at run-time. Sites with Internet firewalls need to specify the TCP/IP port number the daemon uses. If a TCP/IP port number is specified on the VENDOR line, there may be a delay restarting the vendor daemon.

See Also
Managing the Options File for further information regarding options file contents.

Version 6.0 or Later

```
VENDOR sampled
```

USE_SERVER Line

The `USE_SERVER` line takes no arguments and has no impact on the license server. When the application sees the `USE_SERVER` line, it ignores everything in the license file except the preceding SERVER lines and transfers checkout validation to the vendor daemon.

`USE_SERVER` is recommended since it improves performance when a license server is used. For uncounted features, `USE_SERVER` is used to force logging of usage by the daemons.

FEATURE and INCREMENT Lines

A FEATURE and INCREMENT lines describe the license model for a product. Only the first FEATURE line for a given feature name is 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, or *pools*, based on the following fields:

- feature name
- version
- DUP_GROUP
- FLOAT_OK
- HOST_BASED
- HOSTID
- PLATFORM
- USER_BASED
- VENDOR_STRING (if configured by the publisher as a pooling component)

If two lines differ by any of these fields, a new group of licenses, called a *license pool*, is created in the vendor daemon, and this group is counted independently from other license pools with the same feature name. A FEATURE line does not give an additional number of licenses, whereas an INCREMENT line always gives an additional number of licenses.

The basic feature definition line format is:

```
{FEATURE|INCREMENT} feature vendor feat_version exp_date \
            num_lic [optional_attributes] SIGN="<...>"
```

The six fields after the feature definition line keyword are required and have a fixed order. They are defined by the software publisher and cannot be changed. Table 7-4 presents these fields in the order they must appear.

Table 7-4: Feature Definition Line Required Fields

Field	Description
feature	Name given to the feature by the software publisher.
vendor	Name of the vendor daemon; also found in the VENDOR line. The specified daemon serves this feature.
feat_version	Version of this feature that is supported by this license. When this field contains a date with the format yyyy.mmdd, this defines a date-based version that you can set as an Alert in the license server manager, `lmadmin`.
exp_date	Expiration date of license in the format `dd-mmm-yyyy`, for example, 07-may-2008. **Note:** If *`exp_date`* is the string "permanent" or the year is 0 (or 00, 000, 0000) then the license never expires.

Table 7-4: Feature Definition Line Required Fields (cont.)

Field	Description
num_lic	Number of concurrent licenses for this feature. If the `num_lic` is set to the string "uncounted" or 0, the licenses for this feature are uncounted and no license server is required but a hostid on the FEATURE line is required. See Counted vs. Uncounted Licenses.
SIGN=*sign* **or** **AUTH=...**	SIGN= signature to authenticate this FEATURE line. If your publisher has deployed his vendor daemon using the common vendor daemon technology, signatures are embedded within the AUTH= keyword. Contact your publisher for further details.

Table 7-5 lists attributes that may appear in a FEATURE or INCREMENT line. They are supplied at the discretion of the software publisher to define the license model. If present in the FEATURE or INCREMENT line, they must remain there and cannot be altered by the end user. These attributes have a `keyword=value` syntax where `keyword` is in uppercase.

In places where `value` is a string surrounded with double quotes ("..."), the string can contain any characters except a quote.

Table 7-5: Attributes Set by the Software Publisher

Attribute	Description
BORROW[=n]	Enables license borrowing for a particular feature definition line. `n` is the number of hours that the license is borrowed. The default borrow period is 168 hours, or one week.
DUP_GROUP=...	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 FLEXnet ID dongle with `FLOAT_OK` for a particular feature definition line. This feature definition line must also be node-locked to a FLEXnet ID dongle. 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 runs only on the system with the hostid that `lmhostid` returns equal to the `server_hostid` specified with `FLOAT_OK`.

Table 7-5: Attributes Set by the Software Publisher (cont.)

Attribute	Description
HOSTID= *"hostid1 [hostid2 ... hostidn]"*	Id of the host to which the feature line is bound. *hostid* is determined with the `lmhostid` utility. This field is required for uncounted licenses; but can be used for counted licenses as well. See Hostids for Supported Platforms for more information.
HOST_BASED[=*n*]	Host names must be specified in INCLUDE statements in the 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.
ONE_TS_OK	Detects when a node-locked uncounted license is used by an application running under remote desktop.
OVERDRAFT=n	The overdraft policy allows a software publisher to specify a number of additional licenses which users are 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 is reported by the FLEXnet Manager 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.
SUITE_DUP_GROUP=...	Similar to DUP_GROUP, but affects only the enabling FEATURE line for a package suite. It limits the total number of users of the package to the number of licenses, and allows the package to be shared among the users that have the SUITE checked out.
SUPERSEDE= *"f1 f2 ..."*	If this appears, all licenses issued before the date specified in `ISSUED=` are *superseded* by this line and become ineffective.
SUPERSEDE_SIGN= {f1:xxxx, f2:xxxx) **SUPERSEDE_SIGN= {p1:xxxx, p2:xxxx)**	Overrides the license models of all feature definition lines or package lines defined as the value.
TS_OK	FLEXnet Publisher Licensing Toolkit detects when a node-locked uncounted license is running under Windows Terminal Server. To run the application via a Terminal Server client window, TS_OK must be added to the FEATURE line. Without TS_OK, a user running on a Terminal Server client is denied a license.
USER_BASED[=*n*]	Users must be specified in INCLUDE statements in the options file, and the number of users are limited to *num_lic*, or the number specified in =*n*.
VENDOR_STRING="..."	This is a custom value defined by the software publisher and enclosed in double quotes.

The following attributes listed in Table 7-6 are optional and are under control of the license administrator. These attributes have a *keyword=value* syntax where *keyword* is in lowercase.

Table 7-6: Optional Feature Definition Line Attributes

Attribute	Description
asset_info="..."	Additional information provided by the license administrator for asset management.
dist_info="..."	Additional information provided by the software distributor.
sort=*nnn*	Specifies sort order of license file lines. See Sort Rules.
user_info="..."	Additional information provided by the license administrator.
vendor_info="..."	Additional information provided by the software publisher.

Examples

```
FEATURE sample_app sampled 2.300 31-dec-2008 20 \
        SIGN=”<...>”
INCREMENT f1 sampled 1.000 permanent 5 \
        HOSTID=INTERNET=195.186.*.* NOTICE="Licensed to \
        Sample corp" SIGN=”<...>”
```

Sort Rules

Feature definition lines are automatically sorted when they are read from the license file. The default sorting rules are is as follows:

1. License file. Automatic sorting does not occur across files in a license search path.
2. Feature name.
3. FEATURE before INCREMENT.
4. Uncounted before counted.
5. Version, higher versions before lower versions.
6. Issued date, in reverse order, newest first. The date is taken from ISSUED= or START=.
7. Original order is otherwise maintained.

To turn off automatic ordering add `sort=nnn` to the feature definition line, where *nnn* is the same on all lines; *nnn* specifies the relative sort order. The default sort order value is `100`. Lines with a sort order value of less than `100` are sorted before all lines without this attribute, and lines with a sort order value greater than `100` appear after all unmarked lines. All lines with the same number are sorted as they appear in the file.

Changes in FEATURE and INCREMENT Line Format

The following lists the significant changes in the format of feature definition lines and when additional keywords were introduced.

- Version 7.1 and earlier feature definition line format uses *license_key*:

  ```
  {FEATURE|INCREMENT} feature vendor feat_version exp_date \
              num_lic [optional_attributes] SIGN="<...>"
  ```

  The version 7.1 and earlier format is understood by the current release.

- The SIGN= keyword introduced in version 7.1.
- For version 7.1 through version 8.0 client libraries and vendor daemons, the feature definition line must have a SIGN= signature and, for backward compatibility with version 8.1 and earlier, can contain a *license_key*:

  ```
  {FEATURE|INCREMENT} feature vendor feat_version exp_date \
              num_lic [license_key] [optional_attributes] SIGN="<...>"
  ```

- *license_key* obsoleted in version 8.1 client library and vendor daemon
- The keyword "permanent" for *exp_date* introduced in version 6 client library.
- The keyword "uncounted' for *num_lic* introduced in version 6 client library.
- BORROW keyword introduced in version 8.0 client library and vendor daemon.
- FLOAT_OK keyword introduced in version 8.0 client library and vendor daemon.
- TS_OK keyword introduced in version 8.0 client library and vendor daemon.
- AUTH keyword introduced in version 10.8 client library and vendor daemon.

PACKAGE Lines

The purpose of the PACKAGE line is to support two different 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 definition line to license the whole package. A PACKAGE line is shipped by your software publisher with a product, independent of any licenses. Later, when you purchase a license for that package, one or more corresponding feature definition lines enable the PACKAGE line.

Example

```
PACKAGE package vendor [pkg_version] COMPONENTS=pkg_list \
        [OPTIONS=SUITE] [SUPERSEDE["p1 p2 ..."] ISSUED=date]
        SIGN="<...>"
```

Table 7-7 lists the PACKAGE line fields. They must appear in the order listed.

Table 7-7: PACKAGE Line Fields

Field	Description
package	Name of the package. The corresponding feature definition line must have the same name.
vendor	Name of the vendor daemon that supports this package.
pkg_version	Optional field specifying the package version. If specified, the enabling feature definition line must have the same version.

COMPONENTS=*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 definition line. *num_lic* is only legal if OPTIONS=SUITE is not set—in this case the resulting number of licenses is *num_lic* on the COMPONENTS line multiplied by the number of licenses in the feature definition line. Examples:
	`COMPONENTS="comp1 comp2 comp3 comp4"`
	`COMPONENTS="comp1:1.5 comp2 comp3:2.0:4"`

OPTIONS=SUITE	Optional field. Used to denote a package suite. If set, the corresponding feature of the same name as the package is checked out in addition to the component feature being checked out. If not set, then the corresponding feature of the same name as the package is removed once the package is enabled; it is not checked out when a component feature is checked out.
OPTIONS= SUITE_RESERVED	Optional field. If set, reserves a set of package components. Once one package component is checked out, all the other components are reserved for that same user.
SUPERSEDE [="*p1 p2 ...***"]**	Optional field. Used in conjunction with ISSUED date. Replaces all PACKAGE lines for the same package name with ISSUED dates previous to *dd-mmm-yyyy*.
ISSUED= ***dd-mmm-yyyy***	Optional field. Used in conjunction with SUPERSEDE. Replaces all PACKAGE lines for the same package name with ISSUED dates previous to *dd-mmm-yyyy*.
SIGN=*sign* **or** **AUTH=...**	SIGN= signature to authenticate this FEATURE line. If your publisher has deployed his vendor daemon using the common vendor daemon technology, signatures are embedded within the AUTH= keyword. Contact your publisher for further details.

Examples

```
PACKAGE suite sampled 1.0 SIGN="<...>" \
        COMPONENTS="comp1 comp2" OPTIONS=SUITE
FEATURE suite sampled 1.0 1-jan-2008 5 SIGN="<...>"
```

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" is also checked out.

```
PACKAGE suite sampled 1.0 SIGN="<...>"\
        COMPONENTS="apple:1.5:2 orange:3.0:4"
FEATURE suite sampled 1.0 1-jan-2008 3 SN=123 SIGN="<...>"
```

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 apple sampled 1.5 1-jan-2008 6  SN=123 SIGN="<...>"
FEATURE orange sampled 3.0 1-jan-2008 12 SN=123 SIGN="<...>"
```

- Ability to store PACKAGE lines in separate files introduced in version 6 client library.
- *pkg_version* field required in version 7.1 and earlier client library.
- AUTH keyword introduced in version 10.8 client library and vendor daemon.

UPGRADE Lines

```
UPGRADE feature vendor from_feat_version to_feat_version \
exp_date num_lic [options ... ] 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 provide three version 1.0 licenses of **f1** and two version 2.0 licenses of **f1**.

```
INCREMENT f1 sampled 1.000 1-jan-2008 5 SIGN="<...>"
UPGRADE f1 sampled 1.000 2.000 1-jan-2008 2 SIGN="<...>"
```

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

Note: *UPGRADE lines do not work for node-locked, uncounted licenses.*

Feature Lines in Decimal Format

Licenses can be represented in decimal format. Decimal has the advantage that it is simpler to type in, and often the licenses are much shorter. A simple demo license in readable format:

```
FEATURE f1 sampled 1.00 1-jan-2008 0 HOSTID=DEMO SIGN="<...>"
```

and its decimal equivalent:

```
sampled-f1-00737-55296-1825
```

If needed, decimal lines can be mixed with readable format lines in a license file. Use the `lminstall` command to convert decimal licenses to readable format.

See Also
lminstall for additional information on the `lminstall` command.

Order of Lines in the License File

The order of the lines in a license file is not critical. They are sorted when they are processed so that in most cases the optimal result is achieved. However, version 7.0 and earlier versions of FLEXenabled applications and license servers implicitly impose an ordering to license file lines. Note the following suggestions for ordering lines 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.

- 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, it is possible the floating license is consumed instead of the node-locked license, 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, place any uncounted licenses not served by SERVER before the USE_SERVER line. Make sure 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.

See Also
Sort Rules

8

License Models

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

Floating (Concurrent) Licenses

A *floating license* means anyone on the network can use the FLEXenabled application, 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 a license server manager 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 17007ea8
VENDOR sampled
FEATURE f1 sampled 1.00 1-jan-2008 SIGN="<...>"
FEATURE f2 sampled 1.00 1-jan-2008 6 SIGN="<...>"
FEATURE f3 sampled 1.00 1-jan-2008 1 SIGN="<...>"
```

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, called **lulu**. The license server manager, `lmadmin` or `lmgrd`, uses one of the default TCP/IP ports.

Node-Locked Licenses

Node-locking means the FLEXenabled application can be used on one system or a set of systems only. A node-locked license has a hostid on the FEATURE line that identifies a specific host. There are two types of node-locked licenses: uncounted and counted.

If the number of licenses value is set to either zero (0) or `uncounted`, then the license will not be counted which allows the license to be used an unlimited number of times. This configuration does not require a license server because it is not necessary to count the concurrent usage of the features.

The following license file allows unlimited usage of feature **f1** on the systems with hostids of **17007ea8** and **1700ab12**:

```
FEATURE f1 sampled 1.000 1-jan-2008 uncounted HOSTID=17007ea8 SIGN="<...>"
FEATURE f1 sampled 1.000 1-jan-2008 uncounted HOSTID=1700ab12 SIGN="<...>"
```

Alternately, these two FEATURE lines could have been issued by your software publisher with a *hostid list*:

```
FEATURE f1 sampled 1.000 1-jan-2008 uncounted HOSTID="17007ea8 1700ab12" SIGN="<...>"
```

If these were the only FEATURE lines in this license file, neither the license server manager or vendor daemon are necessary and you do not need to start one.

The following license file provides three licenses for feature **f1**, locked to the system with hostid **1300ab43**. Since the license server and licenses are locked to the same system, the daemons run on the same system that runs the FLEXenabled application.

```
SERVER lulu 1300ab43 1700
VENDOR sampled /etc/sampled
FEATURE f1 sampled 1.00 1-jan-2008 3 HOSTID=1300ab43 SIGN="<...>"
```

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 systems **17007ea8** and **1700ab12**, while allowing two other licenses for feature **f1** to be used anywhere else on the network:

```
SERVER lulu 17001234 1700
VENDOR sampled C:\flexlm\sampled.exe
FEATURE f1 sampled 1.00 1-jan-2005 uncounted HOSTID=17007ea8 SIGN="<...>"
FEATURE f1 sampled 1.00 1-jan-2005 uncounted HOSTID=1700ab12 SIGN="<...>"
FEATURE f1 sampled 1.00 1-jan-2005 2 SIGN="<...>"
```

This configuration requires a license server manager and vendor daemon because the licenses on the third FEATURE line are counted.

Counted vs. Uncounted Licenses

The license model (as defined in the license file) determines whether a license server is needed. If all feature definition lines have a license count set to either zero (0) or uncounted, then the customer does not need a license server. This type of license is called uncounted. Alternatively, if any features have a non-zero license count, then the customer needs a license server to count those licenses. If a software publisher wants to use FLEXnet Publisher Licensing Toolkit without a license server, they must issue uncounted licenses.

The license server can serve uncounted licenses also. This is often done so that:

- transactions can be logged into the report log for all license requests, which can then be reported on by FLEXnet Manager
- options file constraints can be applied to the licenses

To have uncounted licenses served, include a SERVER line in the license file, and put the USE_SERVER line immediately after the SERVER line. The vendor daemon serves the uncounted licenses, and the USE_SERVER line indicates to applications that requests must go to the license server for authorization.

Mobile Licensing

End users often want to use applications on computers that do not have a continuous connection to a 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

FLEXnet Publisher Licensing Toolkit supports licenses that allow one of several kinds of mobile licensing:

- Node-locked to a laptop
- Node-locked to a FLEXnet ID dongle
- Node-locked to a FLEXnet ID dongle with FLOAT_OK keyword
- License borrowing with BORROW keyword
- Node-locked to a user name
- Fulfilled from a prepaid license pool

You should use license rehosting if an enterprise wants to move a license without using one of these methods. The software publisher must generate a new node-locked license file for each new client computer. Rehosting requires administrative overhead because the software publisher must be involved involved with each move.

Node-Locked to a Laptop Computer

To use a license exclusively on one laptop computer, the license should be node-locked to an address associated with that computer. The license file resides on the laptop computer.

Node-locked to a FLEXnet ID dongle

To move a license between different Windows systems, you can lock it to a FLEXnet ID dongle (a dongle that connects to a parallel or USB port). You can move this license between systems by installing a copy of the license file on each system and moving the FLEXnet ID dongle from one system to another. Since the license is tied to the FLEXnet ID dongle, only the system with the FLEXnet ID dongle can use the license.

Node-Locked to a FLEXnet ID dongle with FLOAT_OK

Because the FLEXnet ID dongle defines the license server and the license floats on the network, this method has an advantage over simply using a license locked to a FLEXnet ID dongle. Licenses with a FLOAT_OK keyword, and that are node-locked to a FLEXnet ID dongle, are supported only where both the FLEXenabled application and the license server are running on Windows.

A software publisher issues a license file with a FEATURE line node-locked to a FLEXnet ID dongle and containing the FLOAT_OK keyword and a FLEXnet ID dongle for that FEATURE line. One FEATURE line containing the FLOAT_OK keyword and one FLEXnet ID dongle is needed for each instance of a license that is mobile. When the FLEXnet ID dongle is attached to a license server, the license floats on the network. When the FLEXnet ID dongle is removed from the license server, the license is available only on the standalone computer.

This method supports parallel or USB FLEXnet ID dongles. Because it is simpler to attach multiple USB dongles to a computer, USB FLEXnet ID dongles may be preferable.

Initiating FLEXnet ID dongle with FLOAT_OK

A software publisher issues the customer a FLEXnet ID dongle, a dongle driver installer, and a license file that contains a FEATURE line node-locked to that FLEXnet ID dongle containing the FLOAT_OK keyword. A license administrator then:

1. Installs the license file on the license server
2. Attaches all of the FLEXnet ID dongles to the license server
3. Installs the FLEXnet ID dongle driver on the license server
4. Starts the license server or rereads the license file

While the FLEXnet ID dongles are attached to the license server, 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:

1. Copies the license file containing the FLOAT_OK node-locked FEATURE line from the license file on the license server to a license file on the client in the location where the FLEXenabled application expects to find its license file.
2. Moves the FLEXnet ID dongle matching the node-locked FEATURE line from the license server to the client. When the FLEXnet ID dongle is removed from the license server, this license is unavailable on the network.
3. Installs the FLEXnet ID dongle 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 FLEXnet ID dongle, even though that computer is disconnected from the network.

Returning a FLEXnet ID dongle with FLOAT_OK License

Task ***To return the license to the license server so it floats on the network again, the end user:***

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

FLEXnet ID dongle with FLOAT_OK Example

The following is a sample license file. It is shipped with two FLEXnet ID dongles: `FLEXID=7-b28520b9` and `FLEXID=7-b2857678`.

```
SERVER myhost ANY
VENDOR sampled
FEATURE f1 sampled 1.0 permanent uncounted FLOAT_OK \
        HOSTID=FLEXID=7-b28520b9 SIGN="<...>"
FEATURE f1 sampled 1.0 permanent uncounted FLOAT_OK \
        HOSTID=FLEXID=7-b2857678 SIGN="<...>"
```

The user installs the license file and the two FLEXnet ID dongles on the license server. When attached to the license server, each uncounted FLOAT_OK license floats on the network and allows a single use. Therefore, up to two users can use **f1** on the end user's network, except on the license server itself, where the license use is disallowed.

If an user wants to work at home, the 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 FLEXnet ID dongle `FLEXID=7-b28520b9` from the license server to the client, and installs the FLEXnet ID dongle driver on the client computer (this also only needs to be done once). The user disconnects the client computer from the network and uses 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 dongle to the license server, the license administrator runs `lmreread` so the returned license can float again.

- FLOAT_OK keyword introduced in version 8.0 client library, license server manager, and vendor daemon. All components must be version 8.0 or later in order to use FLOAT_OK.

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 the 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 software publisher before an user can borrow licenses.

With license borrowing, a software publisher issues a floating license with a FEATURE line that contains the BORROW keyword. A user specifies the expiration date a borrowed license is to 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 FLEXenabled application automatically uses the local borrowing data to do checkouts during the borrow period. If enabled by the software publisher, borrowed licenses can be returned early, that is, before the borrow period expires. Upon the earlier of either the expiration of the borrow period or the early return of a borrowed license, 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 and the system running the FLEXenabled application.

Initiating License Borrowing

If a software publisher has enabled license borrowing by issuing a license file that contains a FEATURE line with the BORROW keyword, an user initiates 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 initiates license borrowing this way only if the application provides a borrowing interface. Information about this is supplied by the software publisher.

Running the lmborrow Utility

`lmborrow` is one of the `lmutil/lmtools` utilities. To initiate borrowing, the user runs `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 FLEXenabled application's local time. If *`time`* is unspecified, the checkout lasts until the end of the given end date.

For example:

```
lmborrow sampled 20-aug-2007 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/.flexlmborrow` (UNIX). `LM_BORROW` can also be set directly as an environment variable:

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

where:

Table 8-1: LM_BORROW Environment Variable Arguments

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

Table 8-1: LM_BORROW Environment Variable Arguments

Argument	Description
time	Optional. *time* is specified in 24-hour format (*hh:mm*) in the FLEXenabled application's local time. If *time* is unspecified, the checkout lasts until the end of the given end date.

For example:

```
LM_BORROW=15-aug-2006:sampled:20-aug-2006:13:00
```

In this example, one or more licenses served by the `sampled` vendor daemon are borrowed on August 15, 2006, and are scheduled to be returned at 1 P.M. on August 20, 2006.

Borrowing a License

To borrow a license for a desired feature, *on the same day and the same system* that the user runs `lmborrow` or sets `LM_BORROW` (and while still connected to the network), the user runs the application to check out and borrow the license. If the 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 `sampled` vendor daemon. Today, while you are connected to the network, run `lmborrow` or set `LM_BORROW` directly. For example:

```
lmborrow sampled 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, close the application and disconnect your system from the network. The license that you just checked out stays checked out from the license server until the borrow period expires—that license now is used on your disconnected system until the borrow period expires. Once checked out, it remains checked out for the full borrow period. The borrow period cannot be renewed until the period has expired.

Clearing the Borrow Period

Once you have borrowed all the licenses that you need for the current borrow period (defined by the `LM_BORROW` environment variable), prevent licenses for any additional features from being borrowed by running `lmborrow -clear`. This clears the `LM_BORROW` setting in the registry (Windows) or `$HOME/.flexlmborrow` (UNIX). `lmborrow -clear` does *not* clear the local information about licenses you have already borrowed.

Checking Borrow Status

Task ***To print information about borrowed features:***

1. Issue the following command on the system from which they are borrowed:

   ```
   lmborrow -status
   ```

The system that borrowed the features does not have to be connected to the network to determine the status.

Returning a Borrowed License Early

Task ***To return a borrowed license before the borrow period expires:***

1. Reconnect the borrowing system back to the network.
2. From the same system that initiated the borrowing, issue the command:

   ```
   lmborrow -return [-c license_file_list] feature
   ```

This option may or may not be allowed by your software publisher. Check directly with your software publisher to determine if they support borrowed licenses being returned early.

Returning the license early has the effect of clearing the `LM_BORROW` setting for the vendor daemon that serves the returned license.

Support for License Borrowing

See the following sections for more information about the utilities and keywords in the options file that support license borrowing:

- lmborrow utility
- lmdown utility
- lmstat utility
- BORROW_LOWWATER keyword
- EXCLUDE_BORROW keyword
- INCLUDE_BORROW keyword

Note: *BORROW keyword introduced in version 8.0 client library, license server manager, and vendor daemon. All components must be version 8.0 or later in order to use BORROW.*

Node-locked to a User Name

If a license is to be used exclusively by one user on different systems, that license can be node-locked to the user's user name. The license file is copied to the different systems on which the user might work; the user's user name must be identical on each system. For this method to be useful, individual user names in an organization need to be unique.

Fulfilled from a Prepaid License Pool

In this method, the user buys a prepaid number of license-days from the software publisher. The user can then fulfill a license using a partial amount of the total license-days for the given borrow period, node-locked to a particular system. For example, in preparation for a business trip (or even during a business trip), the user fulfills a license that expires in five days that is node-locked to their laptop. Each fulfillment can be node-locked to a different system (or even multiple times to the same system), thus allowing mobility of license usage within the pre-paid number of license-days.

This model is like pay-per-use because each fulfillment is made from a decreasing number license-days. It is different than other pay-per-use models because, once node-locked to a system, that system is allowed unlimited use of the application until the license expires. This short-term license cannot be returned early; once fulfilled, those license-days cannot be refunded. Other pay-per-use models charge based on the number of times the application is used.

9

Managing Licenses from Multiple Software Publishers

You may need to administer licenses from more than one software publisher.

Overview of Multiple License Management Strategies

When you are running FLEXenabled applications from multiple software publishers, you may need to take steps to prevent conflicts during installation. There are several strategies to accomplish this, three of which are presented here:

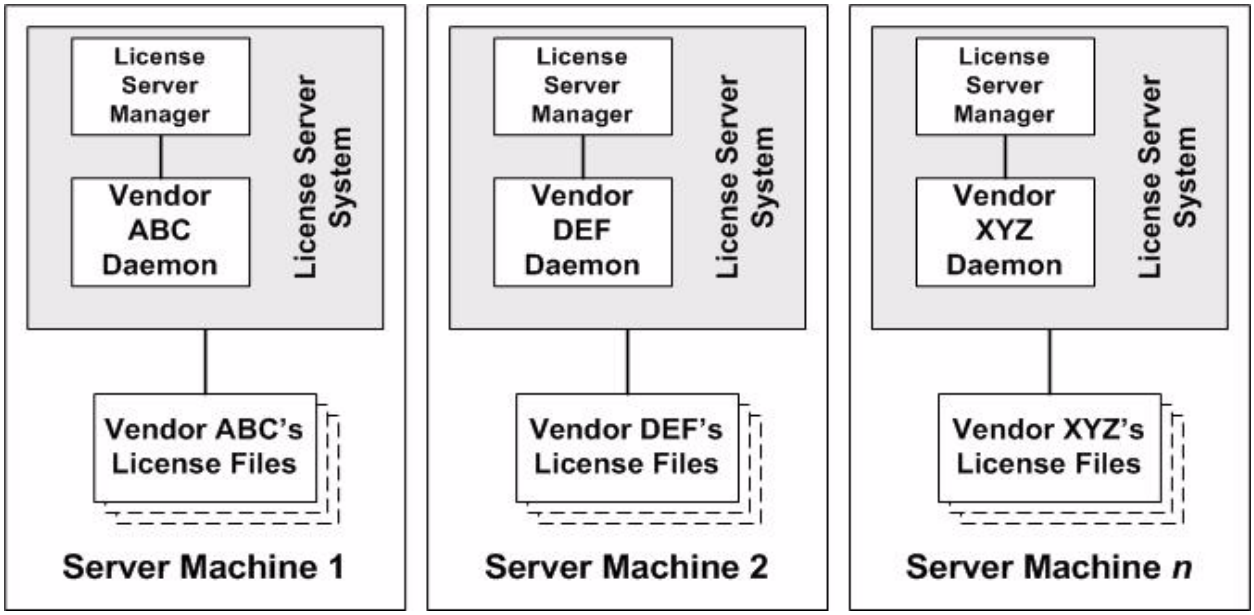
- Multiple systems, each running one license server manager, one vendor daemon, and using one license file.
- One system running multiple license server managers, each managing one vendor daemon and one license file.
- One system running one license server manager, that manages multiple vendor daemons each using its own license file. License files share a common directory.

Each of these three strategies is described in detail in the following sections. Variations are mentioned in Additional Considerations.

Multiple Systems

In this scenario, each license server instance (`lmadmin` or `lmgrd`, vendor daemon, license file, and other files) is located on a separate system. Each system serves licenses just for its vendor daemon and runs its own local copy of the license server manager. Figure 9-1 shows this arrangement.

Figure 9-1: Multiple License Server Systems



Advantages

- The license files for each software publisher are independent from one another.
- Systems are maintained separately. If one system goes down, the other systems continue to serve licenses for their software publishers.
- Each server has its own debug log.
- The license requests are distributed.

Disadvantages

- Administrative overhead is the highest.

Starting the License Server

The following example uses `lmgrd` as the license server manager.

Task ***To start the license server:***

- Invoke the license server manager on each system:

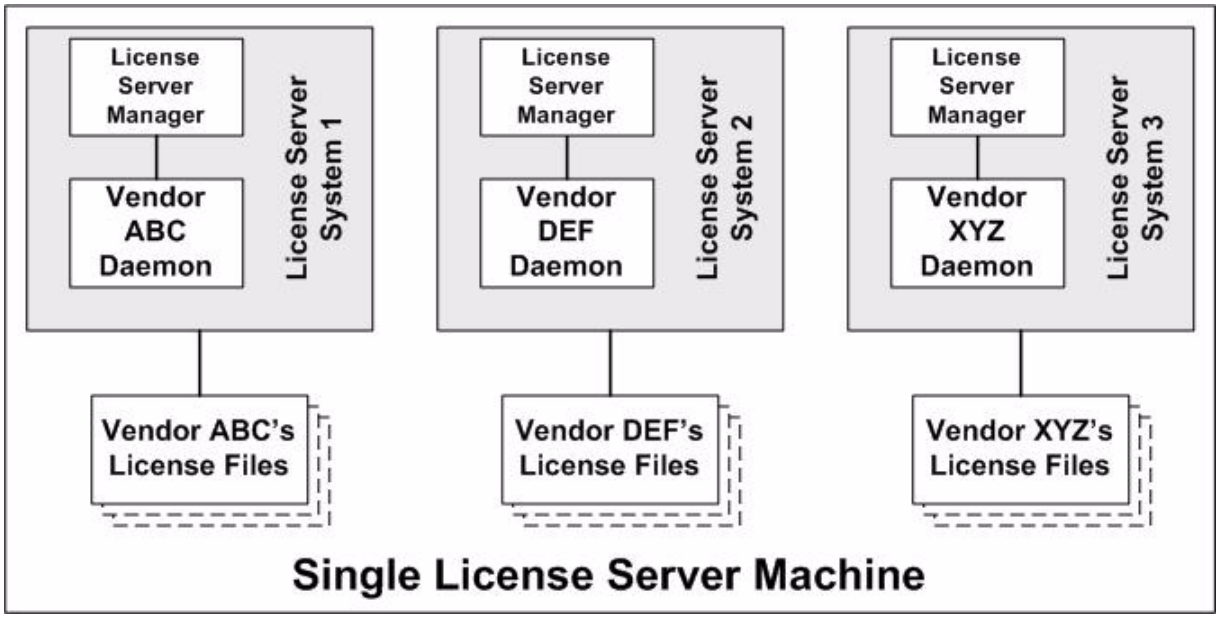
  `lmgrd -c` *`server_system_n_license_list`*

  where `server_system_n_license_list` is a list of license files as described in Managing Multiple License Files. Each `lmgrd` starts the vendor daemon referred to in its license files.

One System with Multiple License Server Instances

In this model, each vendor daemon and its associated license file or files is served by its own license server manager, and everything is contained in one system. Figure 9-2 depicts this scheme.

Figure 9-2: Multiple license server managers, Multiple License Files



When maintaining separate license servers on the same system, keep in mind:

- If the TCP/IP port number is specified on the SERVER line, it must be different for each license server instance. Use a standard text editor to change the TCP/IP port number in each license file so that they are all different. If you are running 10 instances or less, you can omit all port numbers and `lmadmin` or `lmgrd` will choose unique ones for you within the default range of 27000–27009.
- You must make sure that you are using a compatible version of `lmadmin` or `lmgrd` for each particular license file. This is done by using an explicit path. See Version Component Compatibility.

- The number of license server instances is limited only by the CPU, available memory, and networking of the system.

Advantages

- The license files for each software publisher are independent from one another.
- License servers are maintained separately. If one server goes down, the other servers continue to serve licenses.
- Each server has its own debug log.

Disadvantages

- Administrative overhead is high.
- If the system goes down, all licenses are disabled.
- License request load is concentrated to one system.

Starting the License Server

The following example uses `lmgrd` as the license server manager.

Task ***To start the license server:***

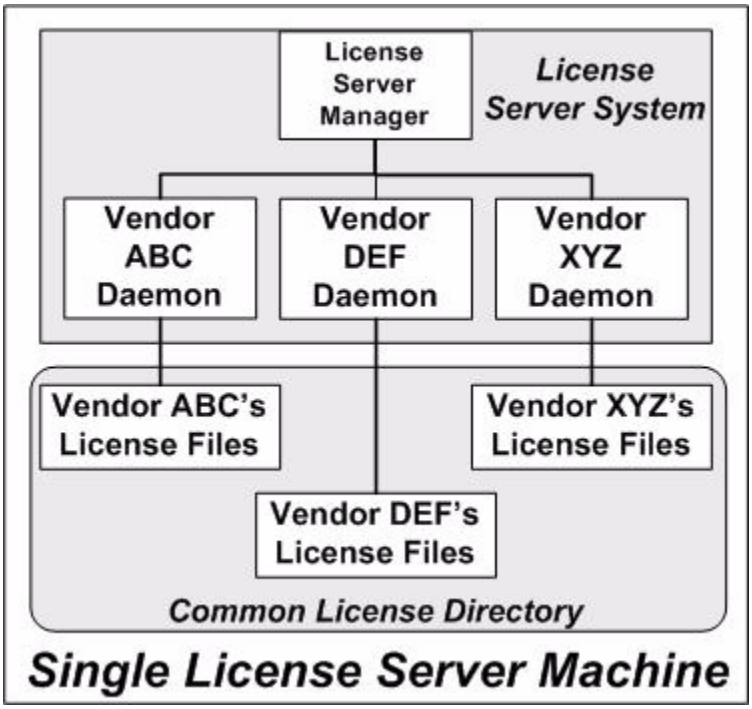
- Invoke each license server:
  - **a.** For Server 1: `lmgrd -c vendor_ABC_license_dir_list`
  - **b.** For Server 2: `lmgrd -c vendor_DEF_license_dir_list`
  - **c.** For Server 3: `lmgrd -c vendor_XYZ_license_dir_list`

  where `vendor_nnn_license_list` is a list of license files as described in Managing Multiple License Files. Each `lmgrd` starts the vendor daemon referred to in its license files.

One System with One License Server and Multiple License Files

In this scenario, one license server manager runs on the system and serves one or more vendor daemons, each with one or more license files. If you are using `lmadmin`, you can maintain license files from different vendors in separate directories. If you are using `lmgrd`, all the license files are usually held in the same directory. The standard filename extension for license files is `.lic`. The number of vendor daemons is not limited by FLEXnet Publisher Licensing Toolkit. Figure 9-3 illustrates this scenario.

Figure 9-3: One license server manager, Multiple License Files



Advantages

- The license files can be maintained separately.
- Reduced administrative overhead.

Disadvantages

- One license server manager serves all vendor daemons. If the license server manager goes down, all licenses are unavailable.
- If the system goes down, all licenses are unavailable.
- Output from all vendor daemons goes into one common debug log unless separate debug logs are specified with DEBUGLOG in each vendor daemon's options file. Having one common debug log makes it harder to debug a single vendor daemon's problem.

- Maximizes licensing load to one system and one license server manager.

Starting the License Server

The following example uses `lmgrd` as the license server manager.

Task ***To start the license server:***

- Invoke the license server manager once on the system.

  `lmgrd -c` *`common_license_directory`*

  `lmgrd` processes all files with the `.lic` extension in `common_license_directory` and starts all vendor daemons referred to in those files, so there is no need to enumerate each license file name on the `lmgrd` command line.

See Also
Managing Multiple License Files
Capturing Debug Log Output for a Particular Vendor Daemon

Managing Multiple License Files

When using `lmgrd` as the license server manager, you can manage multiple license files that are on the same system via a license search path. A license search path is specified two ways:

- By using the `-c` option to `lmgrd`

  `lmgrd -c` *`license_file_list`* `[other lmgrd options]`

- By defining the `LM_LICENSE_FILE` environment variable within the scope of the `lmgrd` process's environment.

Install the license files in convenient locations on the system and then define the `license_file_list`.

Wherever `license_file_list` is specified it consists of a list of one or more of the following components:

- the full path to the license file
- a directory containing one or more license files with a `.lic` extension
- a *`port@host`* setting, where *`port`* and *`host`* are the TCP/IP port number and host name from the SERVER line in the license file. Alternatively, use the shortcut specification, *`@host`*, if the license file SERVER line uses a default TCP/IP port or specifies a port in the default port range (27000–27009).
- A comma separated list of three *`port@host`* specifiers denoting a license servers configured for three-server redundancy. For example,

  `port1@host1,port2@host2,port3@host3`

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

`lmgrd` builds up an internal license search path when it starts up by parsing each entry in the order listed.

Some scenarios where a license search path is used include those described in Multiple Systems, One System with Multiple License Server Instances, or One System with One License Server and Multiple License Files.

When using `lmadmin` as your license server manager, you specify the list of license files for each vendor daemon in the GUI. Use the Import License File screen (accessed from within the Vendor Daemon Configuration screen) to specify a license file. Refer to `lmadmin` on-line help for more information.

See Also
Setting the License Search Path using an Environment Variable
Using Three-Server Redundancy
Environment Variables

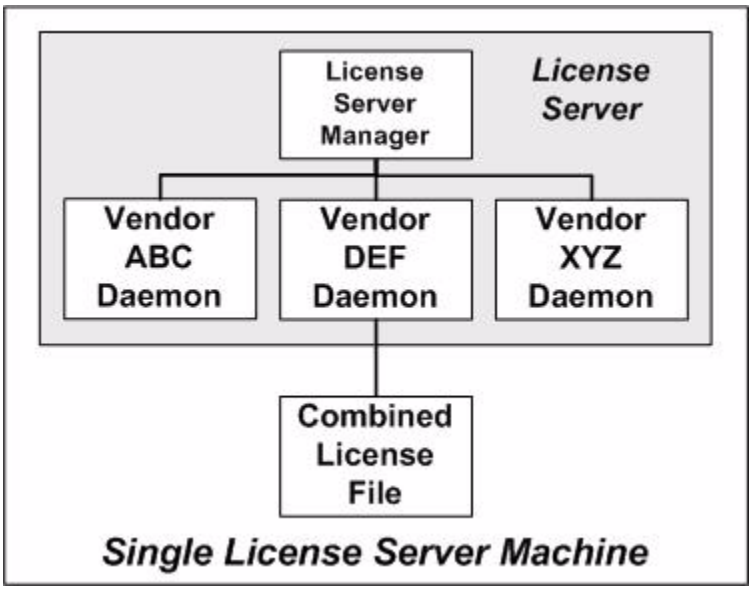
Additional Considerations

Combining license files

If you have two or more products whose licenses are intended for the same system, as specified by their SERVER lines, you may be able to combine the license files into a single license file. This has advantages if you are using `lmgrd` as your license server manager. If you are using `lmadmin` as your license server manager, you do not need to combine license files. When using multiple license files with `lmadmin` import each license file and launch `lmadmin`, which launches each of the vendor daemons defined in the imported license files.

The license files for the models described in One System with Multiple License Server Instances and One System with One License Server and Multiple License Files could be combined if they met certain criteria. See Criteria for Combining License Files. Figure 9-4 shows one possible scenario using a combined license file.

Figure 9-4: One lmgrd, One License File



Advantages

- A single license file to administer.
- Once the files are combined, there is low administrative overhead.

Disadvantage

- Careful planning must be given in combining license lines from multiple software publishers into one file, initially and over time.

Starting the License Server

Task ***To start the license server:***

- Invoke the license server manager once on the system.

  ```
  lmgrd -c combined_license_file
  ```

Criteria for Combining License Files

Your product's license files define the license server systems by host name and hostid in the SERVER lines in the license file. License files are candidates for combining 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:

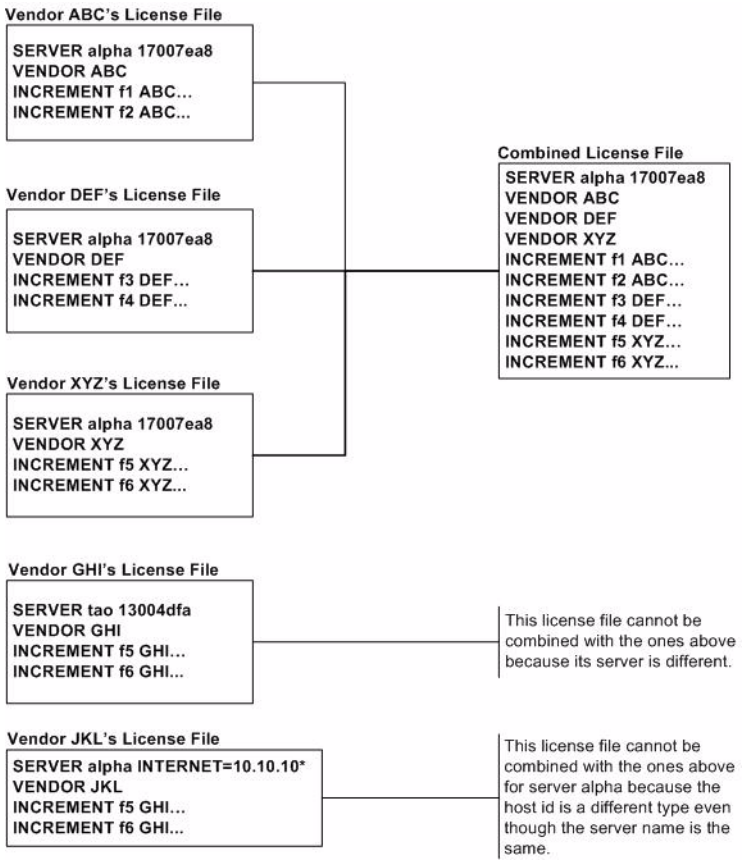
- License files are set up to run on different server systems, so hostids are different.
- One file is set up for a single license server (has only one SERVER line), the other is set up for a three-server redundancy (has three SERVER lines).
- Hostids for the same system use different hostid types. For example, the SERVER line in one license file uses INTERNET= for its hostid type and the other file uses the ethernet MAC address for its hostid type.

If your license files are compatible as described above, then you have the option of combining license files as summarized in Figure 9-4 and below in How to Combine License Files. Note that you are not required to combine compatible license files. There is no performance or system-load penalty for not combining the files.

How to Combine License Files

If your license files are compatible, use any text editor to combine them. 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. Save the resulting data, and you have your combined license file. Figure 9-5 shows an example of combining license files.

Figure 9-5: Combining License Files



Version Component Compatibility

When one license server manager manages multiple vendor daemons, it may be the case that those vendor daemons do not use the same version of FLEXnet Publisher Licensing Toolkit. By observing the FLEXnet Publisher Licensing Toolkit version compatibility rules described in Version Compatibility between Components you are assured that all of your FLEXnet Publisher Licensing Toolkit components are compatible.

You can maintain multiple versions of FLEXenabled applications in the enterprise. The vendor daemon for an application must be at least the same version as the FLEXnet Licensing version used in the FLEXenabled application.

10

Hostids for Supported Platforms

FLEXnet Publisher Licensing Toolkit uses different system identifiers, called *hostids*, for different system architectures. For example, all Sun Microsystems systems have a unique hostid. For this reason, the Ethernet address is used on some system 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," specify "0800200005ac" as the hostid.

Hostid Formats

Numeric, 32-bit hostids are normally used in hexadecimal format. On some systems, the system command returns the ID in decimal format. Use a `#` character before the hostid to indicate a decimal number. For example, if the system command returns **2005771344**, FLEXnet Publisher Licensing Toolkit accepts **#2005771344**. Alternatively, convert the decimal value to hexadecimal.

Obtaining System Hostids

The `lmhostid` utility prints the exact hostid that FLEXnet Licensing requires on any given system. If your hostid contains characters other than the ASCII A–Z, a–z, or 0–9, use the `-utf8` option with lmhostid. To view a correct representation of the resulting hostid, use a utility, such as Notepad, that can display UTF-8 encoded strings.

The following table lists alternate methods to obtain the required hostid for each system architecture. FLEXnet Publisher Licensing Toolkit also supports a group of special hostids and vendor-defined hostids.

Table 10-1: Alternate Hostid Procurement Methods

Hardware Platform	Hostid	Type this command:	Example
AIX (RS/6000, PPC)	32-bit hostid	`uname -m` (returns 000276513100), then remove last two digits and use remaining last eight digits	02765131
HP (32-bit and 64-bit non-Itanium platforms)	32-bit hostid	`uname -i` and convert to hex, or prepend with #	778DA450 or #2005771344
HP (64-bit Itanium)	machine identification	getconf \ CS_PARTITION_ IDENT then prefix with "ID_STRING="	ID_STRING=9c766319-db72-d411-af62-0060b05e4c05
Mac OS X	ethernet address	`/sbin/ifconfig eth0` and remove colons from `ether` value	000A277EA17E
	FLEXnet ID dongle USB port dongle	lmhostid -flexid	FLEXID=9-b28520b9
Linux	ethernet address	`/sbin/ifconfig eth0` and remove colons from HWaddr	00400516E525
	FLEXnet ID dongle USB port dongle	lmhostid -flexid	FLEXID=9-b28520b9
SGI	32-bit hostid	`/etc/sysinfo -s`, convert to hex, or prefix #	69064C3C or #1762020412
Sun	32-bit hostid	hostid	170a3472
	ethernet address	lmhostid -ether	00400516E525
Windows	ethernet address	lmhostid	00B0A9DF9A32
	Disk serial number	DIR C: (look for **Volume Serial Number is** and remove **-**)	DISK_SERIAL_NUM= 3e2e17fd
	FLEXnet ID dongle parallel or USB port dongle	`lmhostid -flexid`	FLEXID=8-b28520b9
		FLEXnet ID dongles are made available by your software publisher. Your software publisher can also provide you with an installer that installs drivers for all FLEXnet ID dongles. For parallel FLEXnet ID dongles, the parallel port must be configured in bi-directional mode.	

Special Hostids

FLEXnet Publisher Licensing Toolkit contains a number of special hostid types that apply to all platforms. These hostid types are valid to use in both SERVER lines and FEATURE lines, wherever a hostid is required. These are:

Table 10-2: Special Hostid Types

Hostid	Description
ANY	Locks the software to any system (meaning that it does not lock anything).
DEMO	Similar to ANY, but only for use with uncounted FEATURE lines.
COMPOSITE= *composite_hostid*	Locks the software to a composite hostid. A composite hostid is a hashed 12-character hexidecimal value formed by combining the values of one or more simple hostids types, as defined by the software publisher. Note that composite hostids are not returned by lmhostid, LMTOOLS, or lmadmin: when composite hostids are used, the software publisher will provide a utility that determines the publisher's composite hostid. On some systems multiple composite hostids may be provided, any of which may be used to identify the system that the software is locked to.
DISPLAY= *display*	Locks the software to display. On UNIX, `display` is `/dev/ttyxx` (which is always `/dev/tty` when an application is run in the background) or the X-Display name. On Windows, it is the system name or, in the case of a terminal server environment, the terminal server client name. (version 8 or later FLEXenabled applications only)
HOSTNAME= host	Locks the software to computer host name *host*.
ID=n	Functionally equivalent to the "ANY" hostid—it runs on any system. The difference is that the license is unique and is used to identify the end user. This hostid is used to lock the license server (on the SERVER line) or the FLEXenabled application (on the feature definition 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
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 is used on the feature definition line as a hostid lock.
USER=*user*	Locks the software to user name *user*.

Examples

```
FEATURE f1 demo 1.0 1-jan-2008 uncounted \
        HOSTID=FLEXID=6-a6300015f SIGN="<...>"
```

or

```
FEATURE f1 demo 1.0 1-jan-2008 uncounted \
        HOSTID=INTERNET=10.10.10.* SIGN="<...>"
```

11

Troubleshooting

This section documents areas of the license server that have given customers difficulty in the past.

General Troubleshooting Hints

This list provides some general debugging information:

- When you start the license server be sure that you direct the output into a local log file where you can examine it. The log file often contains useful information. 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 can 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 version 4.1 or later (version 5 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). See FLEXLM_DIAGNOSTICS for more information.
- When you talk to a support person, be prepared with answers to the following questions:
  - What kind of system is your license server running on?
  - What version of the operating system?
  - What system and operating system is the application running on?
  - What version of FLEXnet Publisher Licensing Toolkit does the FLEXenabled application use?

    Use the `lmver` script, or, on UNIX, execute the following command on your license server manager, vendor daemon, and application:

    ```
    strings binary_name | grep Copy
    ```

    Alternatives are: for `lmadmin`, use the command `lmadmin -version`; for `lmgrd` and the vendor daemon use the -v argument, for example `lmgrd -v`.

- 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 FLEXenabled products?
- Are you using a combined license file or separate license files?
- Are you using a three-server redundancy (i.e. there are multiple SERVER lines in your license file)?

FLEXLM_DIAGNOSTICS

Note: *The ability for FLEXnet Publisher Licensing Toolkit to produce diagnostic output is controlled by your software publisher.*

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

On UNIX, the diagnostic output goes to stderr.

On Windows, the output is a file in the current directory called `flex`*`pid`*`.log`, where `pid` is the application's process ID.

Level 1 Content

If `FLEXLM_DIAGNOSTICS` is set to 1, then the standard FLEXnet Publisher Licensing Toolkit error message is be presented, plus a complete list of license files that the application tried to use. For example:

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

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
FLEXnet 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 is 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`.

This is a description of the arguments to **lm_checkout**:

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

where:

Table 11-1: lm_checkout Arguments

Argument	Description
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`.

Level 3 Content (Version 6.0 or Later 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 Machine: @localhost
app3
Checkout succeeded: f1/BC64A7B120AE
License file: servtest.lic
License Server Machine: @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.

12

Managing the Options File

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

For concurrent (floating) licenses, the license administrator can:

- Allow the use of features
- Deny the use of features
- Reserve licenses

The concurrent licenses can be held either in license files or in fulfillment records within trusted storage.

For activatable licenses, the license administrator can:

- Allow activation of licenses in a specific fulfillment record
- Deny activation of licenses in a specific fulfillment record

For all licenses, the license administrator can:

- Restrict the number of licenses available
- 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.

- PROJECT identification (set by `LM_PROJECT`) in options file was introduced in version 7.0 vendor daemon.
- Option file control for licenses held in fulfillment records in trusted storage has been introduced in 11.3 vendor daemon.

Creating an Options File

Task ***To create an options file:***

1. Use the appropriate options listed in Options File Syntax to create the options file for a vendor daemon using any text editor.
2. Locate the options file anywhere; however, it is recommended that the options file be placed in the same directory as the license file.
3. Add the path to the options file in the license file as the fourth field on the VENDOR line for the application's vendor daemon. For example:

   ```
   VENDOR sampled /etc/sampled \
         [options=]/sample_app/sampled/licenses/sampled.opt
   ```

   enables the `sampled` vendor daemon to look at the specified options file.

If the path is omitted, the vendor daemon automatically looks for a file according to the following criteria:

- The name of the file is *`vendor`*`.opt`, where *`vendor`* is the vendor daemon name.
- The directory that contains the license file used by the license server manager.

Note: *The default options file name, `vendor.opt`, introduced in version 6 vendor daemon.*

Options File Syntax

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

Each line of the file controls one option. Table 12-1 lists the option keywords.

Table 12-1: Option Keywords

Option Keyword	Description
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 (version 8.0 or later vendor daemon).
EXCLUDE	Deny a user access to a feature.
EXCLUDE_BORROW	Deny a user the ability to borrow BORROW licenses.
EXCLUDE_ENTITLEMENT	Deny a user the ability to activate licenses held in a fulfillment record in trusted storage.

Table 12-1: Option Keywords (cont.)

Option Keyword	Description
EXCLUDEALL	Deny a user access to *all* features served by this vendor daemon.
FQDN_MATCHING	Sets the level of host name matching.
GROUP	Define a group of users for use with any options.
GROUPCASEINSENSITIVE	Sets case sensitivity for user and host lists specified in GROUP and HOST_GROUP keywords.
HOST_GROUP	Define a group of hosts for use with any options (version 4.0 or later).
INCLUDE	Allow a user to use a feature.
INCLUDE_BORROW	Allow a user to borrow BORROW licenses.
INCLUDE_ENTITLEMENT	Allow a user to activate licenses held in a fulfillment record in trusted storage.
INCLUDEALL	Allow a user to use *all* features served by this vendor daemon.
LINGER	Allow a user to extend the linger time for a feature beyond its checkin.
MAX	Limit usage for a particular feature/group—prioritizes usage among users.
MAX_BORROW_HOURS	Changes the maximum borrow period for the specified feature.
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 FLEXnet Manager 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.

Comments

Include comments in your options file by starting each comment line with a pound sign, `#`.

Specifying Features

When used within an options file entry, the feature name can be modified with an optional keyword-value pair to fully qualify it. This notation is used for distinguishing a particular group of licenses when there are multiple FEATURE lines for a single feature. The following syntax is used:

```
feature:keyword=value
```

For example:

```
f1:VERSION=2.0
```

specifies the version 2.0 pool of licenses for feature f1.

Note: *A colon (:) is a valid feature name character. If colons are in your feature names, specify a group of licenses with the following alternative syntax using quotation marks and spaces:*

*"**feature keyword=value**"*

The following option keywords are used as feature name modifiers to denote a specific group of licenses:

- VERSION=
- HOSTID=
- EXPDATE=
- KEY=
- SIGN=
- ISSUER=
- NOTICE=
- VENDOR_STRING= (if configured by the publisher as a pooling component)
- dist_info=
- user_info=
- asset_info=

If the USER_BASED or HOST_BASED keywords appear in a feature line, this feature specification syntax must be used to qualify the feature.

Using a package name in place of a feature name applies the option to all of the components in the package.

Specifying License Restrictions Using Type

Some option keywords restrict who may use licenses or where licenses may be used. These options take a `type` argument that specifies what the restriction is based on.

When using the option keywords EXCLUDE, EXCLUDE_ENTITLEMENT, EXCLUDEALL, EXCLUDE_BORROW, INCLUDE, INCLUDE_ENTITLEMENT, INCLUDEALL, INCLUDE_BORROW, MAX, and RESERVE, the following values can be used for type:

- **USER**—user name of the user executing the FLEXenabled application. User names are case sensitive.
- **HOST**—system host name or IP address where the application is executing. Host names are case sensitive. The IP address can contain wildcard characters.

The IP-address can contain wildcard characters.

When using the option keywords EXCLUDE, EXCLUDEALL, EXCLUDE_BORROW, INCLUDE, INCLUDEALL, INCLUDE_BORROW, MAX, and RESERVE, the following values can be used for type:

- **DISPLAY**—display where the application is displayed. On UNIX, DISPLAY is **`/dev/ttyxx`** (which is always **`/dev/tty`** when an application is run in the background) or the X-Display name. On Windows, it is the system name or, in the case of a terminal server environment, the terminal server client name. Display names are case sensitive.
- **INTERNET**—IP address of the system where the application is executing (wildcard characters can be used in the IP address)
- **PROJECT**—`LM_PROJECT` environment variable set by the user who is executing the FLEXenabled application. Project names are case sensitive.

On Windows (without terminal server), the HOST and DISPLAY names are both set to the system name. For licenses that allow checkouts from a terminal server (TS_OK keyword in the feature line), the USER, HOST, and DISPLAY names can be different from one another.

The types listed above take a single member. For example:

```
EXCLUDE coolsoft USER joe
```

To specify a list of users or hosts, first define the list using the GROUP or HOST_GROUP option lines, then use the GROUP or HOST_GROUP type to specify the group name. For example:

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

- IP address as a HOST specification introduced in version 8 vendor daemon.
- Colons in feature names introduced in version 8 vendor daemon.

BORROW_LOWWATER

This option is used for licenses held in license files. When licenses are available in trusted storage, activation is normally provided instead of BORROW.

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

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

Table 12-2: BORROW_LOWWATER Terms

Term	Description
feature	Name of feature being affected.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
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 allows only 7 licenses to be borrowed.

```
BORROW_LOWWATER f1 3
```

DEBUGLOG

`DEBUGLOG [+]`*`debug_log_path`*

Specifies a location for the debug log output from the vendor daemon associated with this options file. Preceding the *`debug_log_path`* with a + character appends logging entries; otherwise, the file is overwritten each time the daemon is started. Note that this affects output from only the vendor daemon associated with this options file. The debug log output of `lmadmin` or `lmgrd` and any other vendor daemons in the same license file is not captured in this file.

On Windows, pathnames which include spaces have to be enclosed in double quotes. If `lmgrd` is started as a service, the default location for the report log file is the `c:\winnt\System32` folder unless a fully qualified path is specified.

See Also:
Configuring the License Server Manager as a Windows Service
lmswitch
Debug Log File—Debug log output restricted to that of just the vendor daemon introduced in version 8 vendor daemon.

EXCLUDE

This option applies to concurrent licenses held in license files and trusted storage.

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

Excludes a user or predefined group of users from the list of who is allowed to use the feature. EXCLUDE supersedes INCLUDE; conflicts between the EXCLUDE list and the INCLUDE list are resolved by the EXCLUDE taking precedence.

Table 12-3: EXCLUDE Terms

Term	Description
feature	Name of the feature or package being affected.
keyword=value	Feature name modifier to denote a group of licenses. See "Specifying Features" for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See "Specifying License Restrictions Using Type" for details.
name	Name of an item of type *`type`* for which license usage is excluded.
group_name	Name of the group to exclude.

Task ***To exclude the user hank from the list of users able to use feature f1:***

```
EXCLUDE f1 USER hank
```

EXCLUDE_BORROW

This option is used for licenses held in license files. When licenses are available in trusted storage, activation is normally provided instead of BORROW.

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

Excludes a user or predefined group of users from the list of who is allowed to borrow licenses for this BORROW feature. EXCLUDE_BORROW supersedes INCLUDE_BORROW; conflicts between the EXCLUDE_BORROW list and the INCLUDE_BORROW list are resolved by the EXCLUDE_BORROW taking precedence.

Table 12-4: EXCLUDE_BORROW Terms

Term	Description
feature	Name of the feature being affected.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type `type` for which license borrowing is excluded.
group_name	Name of the group to exclude from borrowing.

Task ***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
```

EXCLUDE_ENTITLEMENT

This option only applies to licenses held in trusted storage and supplied using activation.

```
EXCLUDE_ENTITLEMENT entitlementId type {name | group_name}
```

Excludes a user or pre-defined group of users, etc., from the list of who is allowed to activate the licenses contained in a fulfillment record held in trusted storage. EXCLUDE_ENTITLEMENT supersedes INCLUDE_ENTITLEMENT; conflicts between the EXCLUDE_ENTITLEMENT list and the INCLUDE_ENTITLEMENT list are resolved by the EXCLUDE_ENTITLEMENT taking precedence.

Table 12-5: EXCLUDE_ENTITLEMENT Terms

Term	Description
entitlementId	The entitlement Id used when requesting a license activation.
type	One of USER, HOST, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type `type` for which license usage is excluded.
group_name	Name of the group to exclude.

Important: *To exclude the user "pete" from the list of users able to activate licenses provided in the fulfillment record specified by the entitlement ID "AB456":*

```
EXCLUDE_ENTITLEMENT AB456 USER pete
```

EXCLUDEALL

This option applies to concurrent licenses held in license files and trusted storage.

```
EXCLUDEALL type {name | group_name}
```

Excludes a user or predefined group of users from the list of who is allowed to use all features served by this vendor daemon.

Table 12-6: EXCLUDEALL Terms

Term	Description
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type *type* for which license usage is excluded.
group_name	Name of the group to exclude.

To exclude any user on the system called **chaos** using all features served by this vendor daemon:

```
EXCLUDEALL HOST chaos
```

FQDN_MATCHING

This option applies to all licenses held in license files or trusted storage.

```
FQDN_MATCHING exact | lenient
```

Sets the level to which host names used in HOST type-specifiers must match the host name sent by the FLEXenabled application. The application is configured to send either its host name or its fully qualified domain name (FQDN) to the vendor daemon for validation with HOST type-specifiers. Check with your software publisher to determine fully qualified domain name support.

Table 12-7: FQDN_MATCHING Terms

Term	Description
exact	The host name in the HOST type specifier must match in content and format to that sent by the application. This is the default setting.
lenient	The host name sent by the application needs match to the extent supplied in the HOST type specifier or by the application, which ever is less restrictive.

Only the last FQDN_MATCHING keyword in the options file has effect; all others are ignored.

Table 12-8 shows the outcome of matching attempts between HOST type-specifiers in the options file and host names sent by the application.

Table 12-8: Host Name Matching Matrix

		Application configured for FQDN—sends.myhost.abc.com	Application not configured for FQDN—sends.myhost.abc.com
FQDN_MATCHING exact	HOST myhost	no	yes
	HOST myhost.abc.com	yes	no
FQDN_MATCHING lenient • Options File	HOST myhost	yes	yes
	HOST myhost.abc.com	yes	yes

Examples

Consider the following example that demonstrates restrictive host name matching:

```
INCLUDE f1 HOST myhost.abc.com
FQDN_MATCHING exact
```

This includes `myhost.abc.com` on the list of hosts able to use feature f1. Furthermore, the host name sent by the application must be a fully qualified domain name that matches `myhost.abc.com` exactly.

In contrast, consider this example, which is less restrictive:

```
INCLUDE f2 HOST myhost.abc.com
FQDN_MATCHING lenient
```

This includes `myhost.abc.com` on the list of hosts able to use feature f2. Host names sent such as `myhost.abc.com` or simply, `myhost` match; but `myhost.xyz.com`, `yourhost,` or `yourhost.abc.com` do not match.

The example below is even more lenient:

```
INCLUDE f2 HOST myhost
FQDN_MATCHING lenient
```

This includes the host name, `myhost,` on the list of hosts for feature f3. Since lenient matching is specified, host names such as `myhost`, `myhost.abc.com`, and `myhost.xyz.com` match, whereas `yourhost` or `yourhost.abc.com` do not match.

See Also
"Specifying License Restrictions Using Type"
FQDN_MATCHING introduced in version 9.3 client library and vendor daemon.

GROUP

`GROUP` *`group_name user_list`*

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

Table 12-9: GROUP Terms

Term	Description
group_name	Name of the group being defined. Group names are case sensitive.
user_list	List of user names in that group. Names are case sensitive. Set the `GROUPCASEINSENSITIVE` options file keyword to turn on case insensitivity. See GROUPCASEINSENSITIVE.

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

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

```
GROUP Hackers bob howard james
```

Note: *USER_GROUP is an alias for GROUP.*

GROUPCASEINSENSITIVE

```
GROUPCASEINSENSITIVE OFF|ON
```

If set to **ON**, user names and host names specified with the options file GROUP and HOST_GROUP keywords, respectively, are treated as case insensitive.

By default, **GROUPCASEINSENSITIVE** is **OFF**, and user names and host names are treated as case sensitive.

HOST_GROUP

HOST_GROUP *group_name host_list*

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

Table 12-10: HOST_GROUP Terms

Term	Definition
group_name	Name of the group being defined. Host group names are case sensitive.
host_list	List of host names in that group. Names are case sensitive. Set the GROUPCASEINSENSITIVE options file keyword to turn on case insensitivity. See GROUPCASEINSENSITIVE.

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

```
HOST_GROUP Pacific tokyo seattle auckland
```

Anywhere a host name can be used in an options file, an IP address can be used instead.

INCLUDE

This option applies to concurrent licenses held in license files and trusted storage.

INCLUDE *feature*[:*keyword*=*value*] *type* {*name* | *group_name*}

Includes a user or predefined group of users in the list of who is allowed to use licenses for this feature. Anyone not in an INCLUDE statement is not allowed to use that feature. EXCLUDE supersedes INCLUDE; conflicts between the EXCLUDE list and the INCLUDE list are resolved by the EXCLUDE taking precedence.

Table 12-11: INCLUDE Terms

Term	Definition
feature	Name of the feature or package being affected.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type ***type*** for which license usage is included.
group_name	Name of the group for which license usage is included.

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 license administrator specifies which users are allowed to use the product, via INCLUDE, and the license limits the number of users that are INCLUDEd.*

INCLUDE_BORROW

This option is used for licenses held in license files. When licenses are available in trusted storage, normally activation is provided instead of BORROW.

INCLUDE_BORROW *feature*[:*keyword*=*value*] *type* {*name* | *group_name*}

Includes a user or predefined group of users in the list of who is allowed to borrow the BORROW feature. Anyone not in an INCLUDE_BORROW statement is not allowed to borrow licenses. EXCLUDE_BORROW supersedes INCLUDE_BORROW; conflicts between the EXCLUDE_BORROW list and the INCLUDE_BORROW list are resolved by the EXCLUDE_BORROW taking precedence.

Table 12-12: INCLUDE_BORROW Terms

Term	Definition
feature	Name of the feature being affected.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type *type* for which license borrowing is included.
group_name	Name of the group for which license borrowing is included.

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

```
INCLUDE_BORROW f1 USER tom
```

INCLUDE_ENTITLEMENT

This option only applies to licenses held in trusted storage.

```
INCLUDE_ENTITLEMENT entitlementId type {name | group_name}
```

Includes a user or predefined group of users in the list of who is allowed to activate the licenses contained in a fulfillment record held in trusted storage. EXCLUDE_ENTITLEMENT supersedes INCLUDE_ENTITLEMENT; conflicts between the EXCLUDE_ENTITLEMENT list and the INCLUDE_ENTITLEMENT list are resolved by the EXCLUDE_ENTITLEMENT taking precedence.

Table 12-13: INCLUDE_ENTITLEMENT Terms

Term	Definition
entitlementId	The entitlement Id originally used when requesting a license activation.
type	One of USER, HOST, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type *type* for which license usage is included.
group_name	Name of the group to include.

To include the user **claire** in the list of users able to activate licenses provided in the fulfillment record specified by the entitlement Id AB456:

```
INCLUDE_ENTITLEMENT AB456 USER claire
```

INCLUDEALL

This option applies to concurrent licenses held in license files and trusted storage.

```
INCLUDEALL type {name | group_name}
```

Includes a user or predefined group of users in the list of who is allowed to use all features served by this vendor daemon. Anyone not in an INCLUDEALL statement is not allowed to use these features.

Table 12-14: INCLUDEALL Terms

Term	Definition
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type *type* for which license usage is included.
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
```

LINGER

This option applies to concurrent licenses held in license files and trusted storage.

`LINGER feature[:keyword=value] seconds`

A lingering license stays checked out for a specified period of time beyond its checkin or FLEXenabled application exit, whichever comes first. This option extends the default linger time configured by the software publisher in the FLEXenabled application.

Note: *The software publisher must have enabled this feature in the FLEXenabled application for it to work. Contact your software publisher to find out if this feature is implemented.*

Table 12-15: LINGER Terms

Term	Definition
feature	Name of the feature.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
seconds	Number of seconds the license lingers. The software publisher sets a minimum value. If you specify a value for `seconds` that is smaller than the minimum, the minimum is used.

To set the linger value for feature `f1` to one hour (3600 seconds):

```
LINGER f1 3600
```

The actual linger time varies somewhat since the vendor daemon checks all lingering licenses just once per minute. If, however, a new license request is made that would otherwise be denied, a check of the lingering licenses is made immediately to attempt to satisfy the new request.

MAX

This option applies to concurrent licenses held in license files and trusted storage.

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

Limits usage for a group or user.

Table 12-16: MAX Terms

Term	Description
num_lic	Usage limit for this user or group.
feature	Feature or package this limit applies to.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type `type` for which usage is limited.

Table 12-16: MAX Terms

Term	Description
group_name	Name of the group to limit.

For example, to limit the user **jan** to five licenses for feature `f1`, include the following line in the option file:

```
MAX 5 f1 USER jan
```

MAX_BORROW_HOURS

This option is used for licenses held in license files. When licenses are available in trusted storage, normally activation is provided instead of BORROW.

`MAX_BORROW_HOURS` *`feature`*`[:`*`keyword=value`*`]` *`num_hours`*

Changes the maximum period a license can be borrowed from that specified in the license file for *`feature`*. The new period must be less than that in the license file. If multiple MAX_BORROW_HOURS keywords appear in the options file, only the last one is applied to *`feature`*.

Table 12-17: MAX_BORROW_HOURS Terms

Term	Description
feature	Feature this borrow period applies to. The *`feature`* must have BORROW enabled.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
num_hours	Number of hours in the new borrow period. This value must be less than that specified in the license file for feature (the default, if not specified, is 168 hours).

MAX_OVERDRAFT

This option applies to concurrent licenses held in license files and trusted storage.

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

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

Table 12-18: MAX_OVERDRAFT Terms

Term	Description
feature	Feature this limit applies to.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
num_lic	Usage limit for this user or group.

NOLOG

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

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

Table 12-19: NOLOG Terms

Entry	Description
NOLOG IN	Turns off logging of checkins. Two separate NOLOG lines are required to turn off logging of checkouts and queued requests.
NOLOG DENIED NOLOG QUEUED	Turns off logging of checkouts and queued requests. License administrators use this option to reduce the size of the debug log file. However, it can reduce the usefulness of the debug log when debugging license server problems.
NOLOG UNSUPPORTED	Suppresses "UNSUPPORTED" messages in the debug log. This suppresses error messages in the debug log that report a failure due to the feature being unsupported.

See Also
lmswitch

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 is overwritten each time the daemon is started.

On Windows, pathnames that include spaces have to be enclosed in double quotes. If `lmgrd` is started as a service, the default location for the report log file is the `c:\winnt\System32` folder unless a fully qualified path is specified.

Note: *FLEXnet Manager is a separate product available from Acresso Software, is used to process report log files. FLEXnet Manager processes only report log files, not debug log files.*

Reporting on Projects with LM_PROJECT

The FLEXnet Manager report writer reports on projects. A project is set up by having all users working on the same project set their `LM_PROJECT` environment variable (or registry on Windows) to a string that describes the project. FLEXnet Manager groups usage by project, as defined by what `LM_PROJECT` was set to when the application was run.

See Also
Configuring the License Server Manager as a Windows Service
Environment Variables
Report Log File

RESERVE

This option applies to concurrent licenses held in license files and trusted storage.

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

Reserves licenses for a specific user.

Table 12-20: RESERVE Terms

Term	Description
num_lic	Number of license to reserve for this user or group.
feature	Feature or package this reservation applies to.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
type	One of USER, HOST, DISPLAY, INTERNET, PROJECT, GROUP, or HOST_GROUP. See Specifying License Restrictions Using Type for details.
name	Name of an item of type *type* for which license usage is reserved.
group_name	Name of group for which license usage is reserved.

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. If a package name is specified, all components that comprise the package are reserved.

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

TIMEOUT

This option applies to concurrent licenses held in license files and trusted storage.

`TIMEOUT feature[:keyword=value] seconds`

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

Note: *The software publisher must have enabled this feature in the FLEXenabled application for it to work. Contact your software publisher to find out if this feature is implemented.*

Table 12-21: TIMEOUT Terms

Term	Description
feature	Name of the feature.
keyword=value	Feature name modifier to denote a group of licenses. See Specifying Features for details.
seconds	Number of seconds after which inactive license is reclaimed. The software publisher sets a minimum value.If you specify a value for `seconds` that is smaller than the minimum, the minimum is used.

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

```
TIMEOUT f1 3600
```

TIMEOUT checks in the licenses if the FLEXenabled application has been inactive for a period longer than the specified time period. The daemon declares a process inactive when it has not received heartbeats from it whereas an active FLEXenabled application sends heartbeats.

A TIMEOUT line must be present in the options file in order to take advantage of the this feature.

TIMEOUTALL

This option applies to concurrent licenses held in license files and trusted storage.

```
TIMEOUTALL seconds
```

Same as TIMEOUT, but applies to all features.

How the Vendor Daemon Uses the Options File

When the vendor daemon is started by `lmadmin` or `lmgrd`, the vendor daemon reads its options file. There is only 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. The `lmreread` utility causes the vendor daemon to reread its options file.

- The `lmreread` utility enhanced in version 8.0 vendor daemon so that it causes the vendor daemon to reread the options file. If you are using earlier versions, the vendor daemon must be stopped and restarted in order for the options file to be reread.

Rules of Precedence in Options Files

Rules of precedence take effect when INCLUDE and EXCLUDE statements are combined in the same options file and control access to the same feature (in license files) or fulfillment record (in trusted storage). The following define the precedence when both types of statements appear together:

- If there is only an EXCLUDE list, everyone who is not on the list is allowed to use the feature.
- If there is only an INCLUDE list, only those users on the list are 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 is not 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 a license 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 either:

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

Options File Examples

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

Simple Options File Example

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

This options file restricts the use of concurrent licenses as follows:

- Reserves one license for the feature **compile** for the user **robert**.
- Reserves three licenses for the feature **compile** for anyone on the system with the host name **mainline**.
- Prevents the user **lori** from using the **compile** feature on any system on the network.
- Causes 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/sampled/licenses/sampled.opt`, then modify the license file VENDOR line as follows:

```
VENDOR sampled /etc/sampled /sample_app/sampled/licenses/sampled.opt
```

Limiting Access for Multiple Users

Each INCLUDE, INCLUDEALL, INCLUDE_BORROW, INCLUDE_ENTITLEMENT, EXCLUDE, EXCLUDEALL, EXCLUDE_BORROW, EXCLUDE_ENTITLEMENT, MAX, 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 excludes **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.

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
```

See Also
HOST_GROUP for more information about defining groups

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:

- Prevents the users **picasso**, **mondrian**, and **klee** from using the feature **spell** on any system on the network.
- Prevents the user **bob** from using the feature **spell** on any system on the network.
- Prevents 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**.
- Allows 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.

EXCLUDE_ENTITLEMENT Example

```
#First Define the group "admin"
GROUP admin johns adrianp maryt
EXCLUDE_ENTITLEMENT qf573k GROUP admin
EXCLUDE_ENTITLEMENT qf573k USER bob
EXCLUDE_ENTITLEMENT qf573k HOST cordelia
```

This options file:

- Prevents the users johns, adrianp, and maryt from activating any licenses contained in the fulfillment record obtained using the entitlement Id qf573k on any system on the network.
- Prevents the user bob from activating any licenses contained in the fulfillment record obtained using the entitlement Id qf573k on any system on the network.
- Prevents any user on the system called **cordelia** from activating any licenses contained in the fulfillment record obtained using the entitlement Id qf573k.
- By implication allows any other users on any system other than **cordelia** to activate the licenses contained in the fulfillment record obtained using the entitlement Id qf573k.

INCLUDE Example

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

This options file:

- Allows the user **picasso** to use the feature **paint** on any system on the network.
- Allows the user **mondrian** to use the feature **paint** on any system on the network.
- Allows any user, as long as they are on the host **bigbrush**, to use feature **paint**.
- Denies access to the feature **paint** to anyone except **picasso**, **mondrian**, or anyone from the host **bigbrush** (by implication).

INCLUDE_ENTITLEMENT Example

```
INCLUDE_ENTITLEMENT gy7210 USER tom
INCLUDE paint USER anthony
INCLUDE paint HOST jupiter
```

This options file:

- Allows the user tom to activate any licenses contained in the fulfillment record obtained using the entitlement Id gy7210 on any system on the network.

- Allows the user anthony to activate any licenses contained in the fulfillment record obtained using the entitlement Id gy7210 on any system on the network.
- Allows any user, as long as they are on the host jupiter to activate any licenses contained in the fulfillment record obtained using the entitlement Id gy7210.
- By implication denies the activation of any licenses contained in the fulfillment record obtained using the entitlement Id gy7210 to anyone except tom, anthony, or anyone using the host jupiter.

13

Environment Variables

Environment variables are not required in order to use FLEXenabled applications. Environment variables are normally used for debugging or for changing license default location.

How to Set Environment Variables

FLEXnet Publisher Licensing Toolkit environment variables are set in two different ways:

- In the process's environment
- In the registry (Windows version 6.0 or earlier) or in `$HOME/.flexlmrc` (UNIX version 7.0 or earlier), which functions like the registry on UNIX.

Windows Registry

On Windows systems other than Windows Vista, the 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`*.

On Windows Vista, the location is HKEY_CURRENT_USER\Software\FLEXlm License Manager.

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.

Environment Variables

The table below provides various environment variables and their definitions:

Table 13-1: Environment Variables

Variable	Definition
FLEXLM_BATCH	Windows only: prevents interactive pop-ups from appearing. Set to 1 if a batch application. (Version 7.0 and later clients)
FLEXLM_DIAGNOSTICS	Used for debugging where applications do not print error message text. Set to 1, 2, or 3, depending on the amount of diagnostic information desired. See FLEXLM_DIAGNOSTICS (Version 5.0 and later clients)
FLEXLM_TIMEOUT	Windows only: Sets the timeout value a FLEXenabled application uses when attempting to connect to a license server port in the range 27000–27009. Values are in microseconds, within the range 0–2,147,483,647. The default setting is 100,000 microseconds.
LM_BORROW	Used for initiating license borrowing and setting the borrow period. See Initiating License Borrowing for more details. On UNIX platforms, `$HOME/.flexlmborrow` is used for the registry instead of `$HOME/.flexlmrc`.
LM_PROJECT	`LM_PROJECT`'s value is logged in the report log file and later reported on by FLEXnet Manager. Limited to 30 characters. (Version 5.0 or later client required.) This can also be used to RESERVE, INCLUDE, and so on licenses with PROJECT. For example: `RESERVE 1 f1 PROJECT airplane` Version 5.0 and later clients and version 7.0 and later vendor daemons are required for this feature.
LM_SERVER_HIGHEST_FD	Used to set the highest file descriptor value, above which the license server will not access.
LM_UTIL_CASE_SENSITIVE	Used by the FLEX*lm* utilities. If set to `1`, the utilities process license file lines as case sensitive. By default, this variable is set to `0`; license files are treated as case insensitive. This environment variable is applicable only when the license server, itself, has been configured by your software publisher to treat license files in a case sensitive manner.
TCP_NODELAY	Improves license server performance when processing license requests. Set to 1 to enable performance enhancements. Use with caution: when enabled it may cause an increase in network traffic.
LM_LICENSE_FILE or **VENDOR_LICENSE_FILE**	Reset path to license file. Can be a license search path, separated by " : " on UNIX and " ; " on Windows. If `VENDOR_LICENSE_FILE` used, *`VENDOR`* is the vendor daemon name used by this application. For example, Macrovision products use `MVSN_LICENSE_FILE`. Can be a file name, or *`port@host`*. See also Setting the License Search Path using an Environment Variable (`VENDOR_LICENSE_FILE` requires version 6.0 and later clients).

14

Error Codes

This section documents FLEXnet Publisher Licensing Toolkit error messages, including general format and error message descriptions.

Error Message Format

FLEXnet Publisher Licensing Toolkit error messages presented by applications have the following components:

- **Error Number**—a positive or negative number.
- **Error Text**—short sentence (< 80 characters) summarizing problem.
- **Error Explanation (optional)**—short paragraph (3–5 lines) explaining problem and possible solutions or work arounds.
- **Minor Error Number**—a positive number starting at 1. These numbers are unique error identifiers and are used by software publishers 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 version 6. Error Explanations, and supporting information, are only available in applications using version 6.0 and later.

These error messages may occur in two formats available with FLEXnet Publisher Licensing Toolkit or may appear in a format customized by the application.

Format 1 (short)

FLEXnet error text (-*lm_errno*, *minor_num*[:*sys_errno*]) [*sys_error_text*]

The error information may be missing.

Example

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

Format 2 (long—version 6.0 and later)

```
FLEXnet error text
 FLEXnet error explanation
[Optional Supporting information]
FLEXnet error: -lm_errno, minor_num. [System Error: sys_errno] ["system_error_text"]
```

Example

```
Cannot connect to license server system
     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"
```

Error Code Descriptions

The following table lists the most common errors produced by FLEXenabled applications.

Table 14-1: Error Codes

Error Code	Description
21	lc_flexinit failed because there was insufficient rights to start the FLEXnet Licensing Service. Resolve this by setting the service to start automatically.
20	FLEXnet Licensing Service is not installed.
13	Computed path to required file is too long for Mac OS X operating system.
12	Invalid bundle ID on Mac OS X operating system.
11	Framework specified by bundle ID was not loaded.
10	Error creating path from URL.
9	Error creating URL.
8	Path string not specified in UTF-8 format.
7	A call to lc_flexinit is not allowed after a call to lc_flexinit_cleanup.

Table 14-1: Error Codes (cont.)

Error Code	Description
6	The executable has not been prepped with the preptool. See “preptool” in *Trusted Storage-based Programming Reference* for instructions.
5	Unable to allocate resources.
4	Initialization failed.
3	Unsupported version of the operating system.
2	Unable to load activation library.
1	Unable to find activation library.
-1	Cannot find license file.
-2	Invalid license file syntax.
-3	No license server system for this feature.
-4	Licensed number of users already reached.
-5	No such feature exists.
-6	No TCP/IP port number in license file and FLEXnet Licensing Service does not exist. (pre-v6 only)
-7	No socket connection to license server 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 system.
-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. Work around: Use IP address (for example, 123.456.789.123) instead of host name.

Table 14-1: Error Codes (cont.)

Error Code	Description
-15	Cannot connect to license server system. The server (lmadmin or `lmgrd`) has not been started yet, or the wrong *`port@host`* or license file is being used, or the TCP/IP port or host name in the license file has been changed. Windows XP SP2 platforms have a limit on the number of TCP/IP connection attempts per second that can be made, which your application may have exceeded. Refer to the manufacturer's documentation on how to change this limit.
-16	Cannot read data from license server system.
-17	Cannot write data to license server system.
-18	License server system does not support this feature.
-19	Error in select system call.
-20	License server system busy (no majority).
-21	License file does not support this version.
-22	Feature checkin failure detected at license server system.
-23	License server system temporarily busy (new server connecting).
-24	Users are queued for this feature.
-25	License server system does not support this version of this feature.
-26	Request for more licenses than this feature supports.
-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 vendor daemon.
-34	Clock difference too large between client and license server system.
-35	In the queue for this feature.
-36	Feature database corrupted in vendor daemon.
-37	Duplicate selection mismatch for this feature. Obsolete with version 8.0 or later vendor daemon.
-38	User/host on EXCLUDE list for feature.
-39	User/host not on INCLUDE list for feature.
-40	Cannot allocate dynamic memory.

Table 14-1: Error Codes (cont.)

Error Code	Description
-41	Feature was never checked out.
-42	Invalid parameter.
-47	Clock setting check not available in vendor daemon.
-52	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.
-59	Message checksum failure.
-60	License server system message checksum failure.
-61	Cannot read license file data from license server system.
-62	Network software (TCP/IP) not available.
-63	You are not a license administrator.
-64	lmremove request before the minimum lmremove interval.
-67	No licenses available to borrow.
-68	License BORROW support not enabled.
-69	FLOAT_OK can't run standalone on license server system.
-71	Invalid TZ environment variable.
-73	Local checkout filter rejected request.
-74	Attempt to read beyond end of license file path.
-75	SYS$SETIMR call failed (VMS). Indicates and error due to an operating system failure.
-76	Internal FLEXnet Publisher error. Please report error to Acresso Software.
-77	Bad version number must be floating-point number with no letters.
-82	Invalid PACKAGE line in license file.
-83	FLEXnet Publisher version of client newer than server.
-84	USER_BASED license has no specified users; see license server system log.

Table 14-1: Error Codes (cont.)

Error Code	Description
-85	License server system doesn't support this request.
-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 FLEXnet Publisher 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. Change `this_host` on the SERVER line in the license file to the actual host name.
-96	License server machine 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 `lmadmin` or `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	A FLEXid borrow error occurred.
-103	Terminal Server remote client not allowed.
-104	Cannot borrow that long.
-105	Feature already returned to license server.

Table 14-1: Error Codes (cont.)

Error Code	Description
-106	License server system out of network connections. The vendor daemon can't handle any more users. See the debug log for further information.
-110	Cannot read dongle: check dongle or driver. Either the dongle is unattached, or the necessary software driver for this dongle type is not installed.
-112	Missing dongle driver. In order to read the FLEXnet ID dongle hostid, the correct driver must be installed. These drivers are available from your software publisher.
-114	SIGN= keyword required, but missing from license certificate. You need to obtain a SIGN= version of this license from your vendor.
-115	Error in Public Key package.
-116	TRL not supported for this platform.
-117	BORROW failed.
-118	BORROW period expired.
-119	lmdown and lmreread must be run on license server.
-120	Cannot lmdown the server when licenses are borrowed.
-121	FLOAT_OK requires exactly one FLEXid hostid.
-122	Unable to delete local borrow info.
-123	Returning a borrowed license early is not supported. Contact the vendor for further details.
-124	Error returning borrowed license.
-125	A PACKAGE component must be specified.
-126	Composite hostid not initialized.
-127	A item needed for the composite hostid is missing or invalid.
-128	Error, borrowed license doesn't match any known server license.
-135	Error enabling the event log.
-136	Event logging is disabled.
-137	Error writing to the event log.
-139	Communications timeout.
-140	Bad message command.

Table 14-1: Error Codes (cont.)

Error Code	Description
-141	Error writing to socket. Peer has closed socket.
-142	Error, cannot generate version specific license tied to a single hostid, which is composite.
-143	Version-specific signatures are not supported for uncounted licenses.
-144	License template contains redundant signature specifiers.
-145	Bad V71_LK signature.
-146	Bad V71_SIGN signature.
-147	Bad V80_LK signature.
-148	Bad V80_SIGN signature.
-149	Bad V81_LK signature.
-150	Bad V81_SIGN signature.
-151	Bad V81_SIGN2 signature.
-152	Bad V84_LK signature.
-153	Bad V84_SIGN signature.
-154	Bad V84_SIGN2 signature.
-155	License key required but missing from the license certificate. The application requires a license key in the license certificate. You need to obtain a license key version of this certificate from your vendor.
-156	Invalid signature specified with the AUTH= keyword.
-157	Trusted storage has been compromised; repair needed. Contact your vendor for repair instructions.
-158	Trusted storage open failure. Contact your vendor for further information.
-159	Invalid fulfillment record. Contact your vendor for further information.
-160	Invalid activation request received. Contact your vendor for further information.
-161	No fulfillment exists in trusted storage which matches the request. Contact your vendor for further information.
-162	Invalid activation response received. Contact your vendor for further information.
-163	Cannot return the specified activation. Contact your vendor for further information.
-164	Return count(s) would exceed the maximum for the fulfillment. Contact your vendor for further information.
-165	No repair count left. Contact your vendor for further repair authorization.
-166	Specified operation not allowed. Contact your vendor for further information.

Table 14-1: Error Codes (cont.)

Error Code	Description
-167	The requested activation has been denied because the user or host is excluded from activating this entitlement by a specification in the options file.
-168	The options file contains include specifications for the entitlement, and this user or host is not included in these specifications.

15

Report Log File

The license server produces both report log files and debug log files. The focus of this appendix is report log files. For information on debug log files see Debug Log File.

The report log file contains feature usage information and is generated by the vendor daemon. However, a vendor daemon does not write report logs by default; this action must be enabled. The data in report logs is compressed, authenticated, and organized into a repository.

Use Acresso Software's software license administration solution, FLEXnet Manager, to gain exceptional visibility into license usage data and to create insightful reports on critical information like license availability and usage. FLEXnet Manager can be fully automated to run these reports on schedule and can be used to track license servers and usage across a heterogeneous network of server including Windows NT, Linux and UNIX. Contact Acresso Software at www.acresso.com for more details on how to obtain an evaluation copy of FLEXnet Manager for your enterprise.

Managing Report Log Output

As a vendor daemon runs for a period of time, the volume of report log output increases. If you have a lot of license activity, these log files grow very large. You need to consider where to put these files and how often to rotate and archive them. Therefore, it may be necessary to rotate or switch report log output into different files over time, each file containing license activity over a particular period of time.

Report log data is collected by the vendor daemon into an internal data buffer area before being flushed to the output file. The daemon's internal buffer is flushed once a minute or whenever it gets full, whichever occurs first. To ensure the freshest data possible in the report log file, flush the buffer on demand with the `lmreread` command. Use standard file compression tools to 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 the vendor daemon write its report log to a file on a disk local to the system running the vendor daemon. Each vendor daemon must write to its own report log file.

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 REPORTLOG for more details.
- Invoke `lmswitchr` on the vendor daemon. See lmswitchr 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 different period of time. There are three ways in which to do this whether the vendor daemon is running or not:

- Change the REPORTLOG line in the vendor daemon's options file and reread its options file by invoking `lmreread` (version 8.0 or later vendor daemon) or restart.
- Invoke `lmswitchr` on the vendor daemon. See lmswitchr for more details.
- Invoke `lmnewlog` on the vendor daemon. Requires a version 7.1 or later vendor daemon. See lmnewlog for more details.

16

Debug Log File

The license server produces both debug log files and report log files. The focus of this section is debug log files. For information on report log files, see Report Log File.

A debug log file contains status and error messages useful for debugging the license server. A license server always generates debug log output. Some of the debug log output describes events specific to `lmadmin` or `lmgrd` and some of the debug log output describes events specific to each vendor daemon.

Managing Debug Log Output

As the license server manager and its vendor daemons run for a period of time, the volume of this output increases. 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; either archive or delete the old output.

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

See Debug Log Messages for a description of the debug log output format.

Capturing Debug Log Output for a License Server

If you are using `lmadmin` as your license server manager, separate log files are created for `lmadmin` and each vendor daemon that it manages. The log files are written to the `<root directory>/logs` directory.

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, 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 controlled by the same license server can be written to their own files (version 8.0 and later vendor daemon). There are three ways to do this:

- If you are using `lmadmin` as your license server manager, you configure the location and file name directly in the GUI from the Vendor Daemon Configuration screen. See on-line help for information on Vendor Daemon Log.
- Add the DEBUGLOG line to the options file for each vendor daemon. See DEBUGLOG for more details.
- Invoke `lmswitch` on the vendor daemon. See lmswitch for more details.

Note that `lmgrd` writes its own debug log output to standard out.

Redirecting Debug Log Output for a Running Vendor Daemon

It is possible to redirect the debug log output for a particular vendor daemon 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 options file by invoking `lmreread`. See DEBUGLOG for more details.
- Invoke `lmswitch` on the vendor daemon. See 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. One of the reasons you may want to limit the events that are logged is to reduce the size of the debug log output.

See Also
NOLOG

Debug Log Messages

FLEXnet Publisher Licensing Toolkit processes generate debug log files in the following format:

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

where:

Table 16-1: Debug Log Messages

Message	Description
hh:*mm*:*ss*	Time that the message was logged.

Table 16-1: Debug Log Messages

Message	Description
daemon	Either `lmadmin`, `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 FLEXnet Manager.*

Informational Messages

Table 16-2 lists the various informational messages used within FLEXnet Publisher Licensing Toolkit.

Table 16-2: Information Messages

Message	Description
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 kill all the vendor daemons; vendor daemons shut themselves down.
Lost quorum	The daemon lost quorum, so it processes only connection requests from other daemons.
MULTIPLE vendor servers running. Kill and restart license daemon.	The license server manager has detected that multiple vendor daemons with the same `vendor` name are running. Shutdown `lmadmin` or `lmgrd` and all `vendor` daemons and then restart `lmadmin` or `lmgrd`.

Table 16-2: Information Messages

Message	Description
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 TCP/IP 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 server manager 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 server manager logs this message whenever it starts a new vendor daemon.
Trying to connect to host	The daemon is attempting a connection to `host`.

Configuration Problem Messages

Table 16-3 lists configuration problem messages found in FLEXnet Publisher Licensing Toolkit.

Table 16-3: Configuration Problem Messages

Message	Description
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	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.

Table 16-3: Configuration Problem Messages

Message	Description
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`).

Daemon Software Error Messages

Table 16-4 lists various daemon software error messages:

Table 16-4: Daemon Software Error Messages

Message	Description
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.

17

Identifying FLEXnet Licensing Versions

Version Compatibility between Components

In general, always use the latest version of `lmadmin`, `lmgrd` and `lmutil/lmtools`, all of which are available from www.acresso.com, and you will automatically enjoy many of the enhancements available in the most recent versions of FLEXnet licensing. However, some enhancements require a vendor daemon built with a newer version of FLEXnet Publisher Licensing Toolkit, and yet others require a FLEXenabled application built with a newer version of FLEXnet Publisher Licensing Toolkit. Contact your software publisher for the latest version of their vendor daemon.

The rules about FLEXnet licensing component version compatibility are summarized as:

- Version of `lmutil/lmtools` must be >=
- Version of `lmadmin` (see note) or `lmgrd`, which must be >=
- Version of vendor daemon, which must be >=
- Version of the client library linked to the FLEXenabled application, which must be >=
- Version of license file format

Except for the license file, 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.

Note: *lmadmin can only be used with components with a version of 9.2 or higher.*

Determining the License File Version

The following rules apply to individual FEATURE, INCREMENT or UPGRADE lines. It is 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.

Table 17-1: Determining the License File Version

Version	Description
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 8.0	BORROW, FLOAT_OK, and TS_OK appear.
>= Version 8.1	SUITE_RESERVED appears.
>= Version 8.4	COMPOSITE appears.
>= Version 11.5	ONE_TS_OK and SUPERSEDE_SIGN appear.

Version Summary

Version 1.0—1988

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

Version 1.5—February 1990

First widely used version including DEMO

Version 2.1—March 1991

- Improved TIMEOUT support
- Improved ethernet hostid support

Version 2.21—November 1991

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

Version 2.26—March 1992 (Used only by Sun)

- Added license lingering

Version 2.4—December 1992

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

Version 2.61—March 1993 (Used only by Sun)

- Added INCREMENT and UPGRADE lines to license file

Version 3.0—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 search path
- New, optional license file format with *keyword=value* syntax for optional new features, including: asset_info, ISSUER, and NOTICE, " \ " license file continuation character, 2,048 character limit per feature

Version 4.0—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 diagnostics

Version 4.1—May 1995

- Performance improvements and new platform support

Version 4.1—Patch Release 6, October 1995

- Windows patch release for Windows 95 with various performance improvements

Version 5.0—March 1996

- Improved *port@host* behavior—FLEXenabled 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 expire within two weeks
- Improved options file functionality

Version 5.11—February 1997

- SUPERSEDE lists, PLATFORMS= license attribute,
- new options: MAX, TIMEOUTALL
- Windows control panel added
- Windows license generator `GENLIC` added

Version 5.12—April 1997

- Performance improvements and new platform support

Version 6.0—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 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 TCP/IP port numbers make SERVER line port number optional
- Default options file path
- `this_host` host name supported on SERVER line
- `VENDOR_LICENSE_FILE` supported (for example, `DEMO_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

Version 6.1—June 1998

- Performance improvements

Version 7.0—August 1999

- License Certificate Manager support for automatic license fulfillment
- Support for try-before-you-buy licensing
- License file handles inserted new lines 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 (version 7.0 vendor daemon and `lmgrd`)
- `lmreread` and `lmdown` take `-all` argument to shut down or reread all `lmgrds`
- 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 (version 7.0d or later, November 1999)

Version 7.1—August 2000

- Security enhancements
- SIGN= keyword in license
- `lmnewlog` utility (version 7.0d or later vendor daemon)

Version 7.2—December 2000

- Performance enhancements

Version 8.0—October 2001

- `lmborrow` (version 8.0 or later components), `lmpath` (version 8.0 or later vendor daemon), `lmswitch` (version 8.0 or later vendor daemon) utilities
- `lmreread` rereads options file and SERVER host name
- License borrowing with BORROW keyword

Version 8.1—January 2002

- CRO Security enhancements

Version 8.2—August 2002

- Support added for Windows XP compliancy

Version 8.3—October 2002

- Support added for returning borrowed licenses early

Version 8.4—January 2003

- Support for reserved package suites

Version 9.0—March 2003

- Support for COMPOSITE= hostid type

Version 9.2—July 2003

- Options file keywords added: `GROUPCASEINSENSITIVE` and `MAX_BORROW_HOURS`

Version 9.5—November 2004

- New environment variable: `LM_UTIL_CASE_SENSITIVE`

Version 10.0—April 2004

- Released as FLEXnet Licensing
- Support for fully qualified domain names

Version 10.1—November 2004

- Additional FLEXid driver support for USB dongles

Version 10.8—April 2005

- IPv6 address support for hostids
- Enhanced three-server redundant configuration support
- Support for common vendor daemons

Version 11.1—November 2005

- Support for license rights in trusted storage
- IPv6 support for hostids reverted in this release

Version 11.5—November 2007

- Support new attribute for the NOLOG Option keyword

- IPv6 support
- New error codes
- New feature definition line keywords–ONE_TS_OK and SUPERSEDE_SIGN

Version 11.6—May 2008

- New license server manager, `lmadmin`, which requires components with a minimum version of 9.2.
- Support for multiple ethernet hostids on Linux platforms.

Index

F

G

H

I

L

S

T

U

V