FLEXERA SOFTWARE

FLEXERA SOFTWARE™
FlexNet Publisher®

FlexNet Publisher 2015 (11.13.1)
Getting Started with License File–Based Licensing

Legal Information

Book Name:	Getting Started with License File–Based Licensing]
Part Number:	FNP-11131-GSLF00
Product Release Date:	June 2015

Copyright Notice

Copyright © 2015 Flexera Software LLC. All Rights Reserved.

This product contains proprietary and confidential technology, information and creative works owned by Flexera Software LLC and its licensors, if any. Any use, copying, publication, distribution, display, modification, or transmission of such technology in whole or in part in any form or by any means without the prior express written permission of Flexera Software LLC is strictly prohibited. Except where expressly provided by Flexera Software LLC in writing, possession of this technology shall not be construed to confer any license or rights under any Flexera Software LLC intellectual property rights, whether by estoppel, implication, or otherwise.

All copies of the technology and related information, if allowed by Flexera Software LLC, must display this notice of copyright and ownership in full.

Intellectual Property

For a list of trademarks and patents that are owned by Flexera Software, see http://www.flexerasoftware.com/intellectual-property. All other brand and product names mentioned in Flexera Software products, product documentation, and marketing materials are the trademarks and registered trademarks of their respective owners.

Restricted Rights Legend

The Software is commercial computer software. If the user or licensee of the Software is an agency, department, or other entity of the United States Government, the use, duplication, reproduction, release, modification, disclosure, or transfer of the Software, or any related documentation of any kind, including technical data and manuals, is restricted by a license agreement or by the terms of this Agreement in accordance with Federal Acquisition Regulation 12.212 for civilian purposes and Defense Federal Acquisition Regulation Supplement 227.7202 for military purposes. The Software was developed fully at private expense. All other use is prohibited.

Contents

1 Introduction

This manual describes the basic concepts of FlexNet Licensing, license models, and concepts for designing license policies using license files. For a complete overview of FlexNet Publisher capabilities, also read the companion document, *Getting Started with Trusted Storage-Based Licensing*.

Table 1-1 • *Getting Started with License File–Based Licensing* Chapter Overview

Topic	Content
Introduction to License File–Based Licensing	Provides a general overview of FlexNet licensing using license rights held in license files.
Overview of the FlexNet Publisher Licensing Toolkit	Provides an overview of the structure and contents of the FlexNet Publisher toolkit.
Evaluating FlexNet Publisher on Windows	Walks you though the process of evaluating FlexNet licensing behavior on Windows, both from the end user's and the software publisher's perspective.
Evaluating FlexNet Publisher on UNIX	Walks you though the process of evaluating FlexNet licensing behavior on UNIX, both from the end user's and the software publisher's perspective
License Models	Overview of FlexNet licensing and the basic steps for creating a license model.
Publisher Decisions	Presents licensing policy concepts that you need to think about when designing a licensing policy for your product.
License Security Systems	Discusses license security, including issues and strategy for implementation.
License Fulfillment	Overview of license fulfillment and installation at the end-user site.
Licensing Case Study	Case study based on products and business strategies of a fictitious company, which provides insight into realistic FlexNet licensing scenarios.

FlexNet Publisher Product Documentation

The following documents provide information about FlexNet Publisher:

- *Release Notes* communicates information about changes to the application and supported third-party applications.
- *Installation Guide* describes how to install the Licensing toolkit.
- *Getting Started with License File-Based Licensing* contains information for those software vendors new to FlexNet Licensing using license file-based licensing. It includes an evaluation procedure that introduces these licensing concepts from both the software publisher and end-user’s perspective.
- *Development Environment Guide* describes how to integrate the toolkit into the development environment, build a production licensing toolkit, and distribute a FlexEnabled application.
- *Programming Reference for License File-Based Licensing* contains information about license file-base licensing. Certain information is relevant to both license file-based and trusted storage-based licensing. You should read this document and *Getting Started with License File-Based Licensing* to understand the basic concepts of FlexNet Licensing if you have not previously used it.
- *C/C++ Function Reference* summarizes and describes all publicly available C/C++ functions in the Licensing toolkit.
- *Licensing for Java Programming Reference Guide* contains guidelines for using the FlexNet Licensing for Java toolkit.
- *License Administration Guide* contains information on the setup and administration of a licensing system, including setting up an options file and using Licensing utilities.

The activation transactions described in this document may be used with FlexNet Operations or a license generator built using the License Generator toolkit. The manual response generator is provided as a test tool and can be used to generate the responses that in a live system are provided by FlexNet Operations or a License Generator. For more information, see the FlexNet Operations documentation and the *License Generator Toolkit Guide*.

2 Introduction to License File–Based Licensing

FlexNet Publisher is a product-licensing package that enables software publishers to license a product on a per-computer (or node-locked) basis, as well as on a concurrent-usage (or floating) basis. Using FlexNet Publisher, you can implement a wide variety of functionality within your licensing policy.

FlexNet Publisher features include:

- A broad range of licensing policy options, including:
    - Node-locked licenses
    - Demo licenses
    - Personal-use licenses
    - Floating licenses
    - Counted and uncounted licenses
    - Expiring licenses
- Administration tools for end-user license administrators
- Support for independent features from one or multiple software publishers
- Several publisher-definable fields for each product feature
- Operation in a heterogeneous network of supported computer systems
- Transparent reconnection of the FlexEnabled product when its license server becomes unavailable, including conditions of license server machine failure
- Simple configuration by using a single license file per network
- Configuration controls for system administrators

The FlexNet Licensing Ecosystem

The FlexNet Licensing ecosystem consists of:

- You, the **software publisher**—You develop the license policy and *FlexEnable* your product using FlexNet Publisher.
- The **end user**—Your customer to whom you deploy your FlexEnabled product and all related components.
- **FlexNet Publisher**—FlexNet Publisher is a toolkit that resides on a development machine at your site. You use it to integrate FlexNet licensing, using a client-side library, into your product; to generate license certificates; and to make customizations to the license server (if you use a served license model).
- **FlexNet Publisher components**—This includes your FlexEnabled product, as well as the license certificate and other license model components, that you deploy at your end user's site. The FlexNet Publisher components that are required depend on the license model that you choose to implement. See the following sections, License Model Overview and FlexNet Publisher Components, for more information.

License Model Overview

A major distinguishing characteristic of a license model is whether it requires a license server (served license model) or does not require a license server (unserved license model). The need for a license server may result from a FlexNet Publisher requirement, your requirement, your end user's requirement, or a combination of the three. The section Determining the Need for a License Server explores license server requirements in more depth.

The license model you choose—served or unserved—dictates which FlexNet Publisher components are required. The components are described in the section FlexNet Publisher Components.

Unserved License Model

An unserved license model does not require a license server. Examples of this type of licence model include: expiring evaluation licenses; site licenses available to an unlimited number of users; and single-use, node-locked licenses.

The unserved license model (shown in Figure 2-1) requires two components:

- FlexEnabled Application
- License Certificate

For an unserved license, the FlexEnabled application and license certificate reside on the same machine.

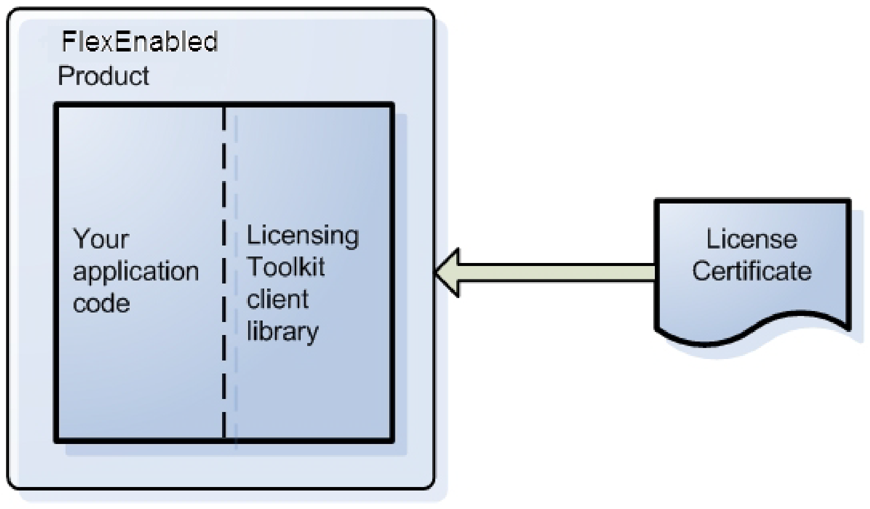


Figure 2-1: FlexNet Licensing unserved license model components

Implementing your licensing policy for unserved licenses requires you to:

- Instrument your product with calls to the FlexNet Publisher client library.
- Create a license certificate using keywords and values appropriate to your license model.

Served License Model

A served license model requires a license server. Served licenses allow a specified number of concurrent users access to your application at any one time. The licenses are not node-locked to a specific machine; instead, the licenses float on the network, which allows for network-wide usage. The license server keeps track of the number of concurrent users.

For served licenses, the following FlexNet Licensing components are required:

- FlexEnabled Application
- FlexNet License Server Manager `lmadmin` or `lmgrd`
- Vendor Daemon, which, along with the license server manager, composes the license server
- License Certificate

The following are optional components:

- Debug Log File
- Administrative Options File
- Report Log File

Figure 2-2 shows the relationship that these components have to one another. Each component is described in the section, FlexNet Publisher Components.

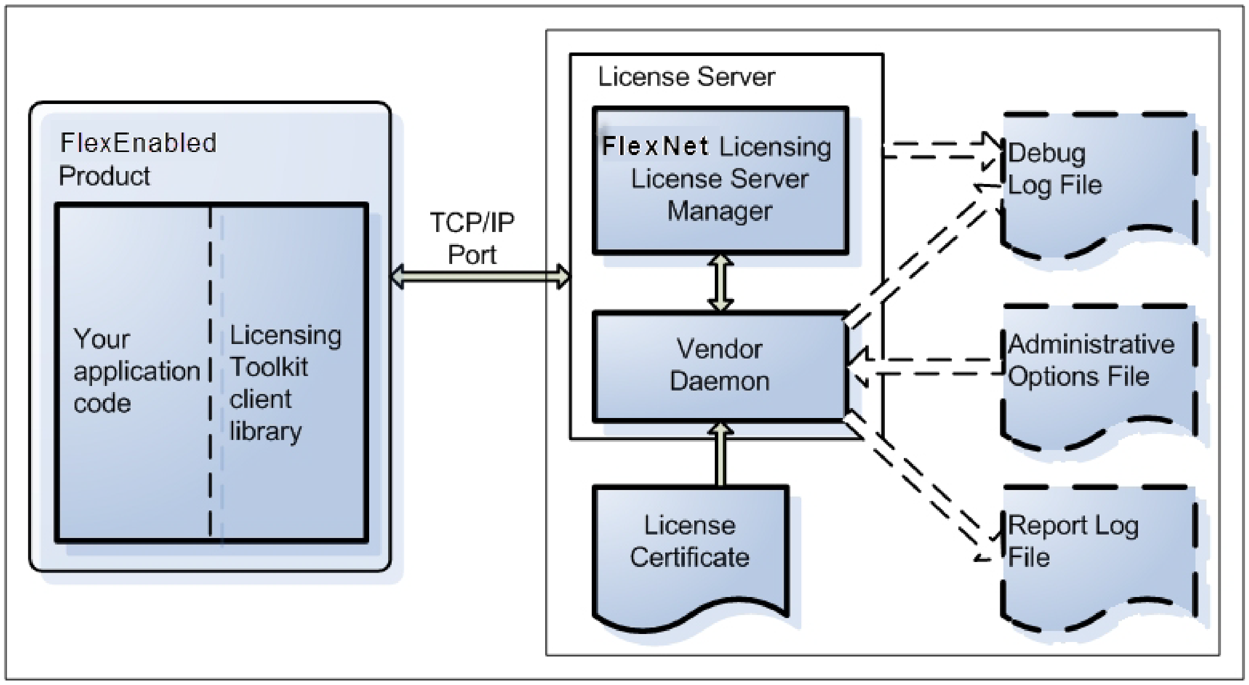


Figure 2-2: FlexNet Licensing served license model components

Implementing your licensing policy for served licenses requires you to:

- Instrument your product with calls to the FlexNet Publisher client library.
- Configure a vendor daemon using your publisher-specific information.
- Create a license certificate using keywords and values appropriate to your license model.

The license server components can reside on the same machine as the FlexEnabled product; however, typically they reside on a machine on the network and communicate with the product using a TCP/IP port.

At the end-user site, the license administrator optionally enables the debug and report logs, and configures the FlexNet Publisher end-user administrative options file.

FlexNet Publisher Components

The components described in this section are used in FlexNet Publisher licensing. The required components depend upon the license model that you choose to implement.

FlexEnabled Application

The FlexEnabled application is your product that has been instrumented with calls to the FlexNet Publisher client library. The client library provides routines that establish the product-side licensing policy, read the license file, and—in the case of served licenses—provide communication with the license server. This ultimately results in granting or denying usage.

License Certificate

The software publisher creates the license certificate, which can be installed as a license file by the license administrator or installed automatically as part of the product installation process. The license certificate contains at least one line of data (a feature definition line) for each discrete unit of capability in the software application for which you want to enable licensing. Each feature definition line begins with the keyword `INCREMENT` or `FEATURE`, and contains a license key or signature based on the data contained in that line.

For a served license model, information about the license server machines, their hostids, and vendor daemons are specified using `SERVER` and `VENDOR` lines. A FlexEnabled product implementing an unserved license model need only read a valid license file to run—it does not need a license server.

FlexNet License Server Manager

The FlexNet license server manager, `lmadmin` or `lmgrd`, handles the initial contact with the FlexEnabled product, passing the connection on to the appropriate vendor daemon. It also starts and restarts the vendor daemon.

Vendor Daemon

With FlexNet Publisher licensing, served licenses are granted by processes running on the license server machine. There is one process specific to each software publisher who has a FlexEnabled product on the network. This process is called the vendor daemon. The vendor daemon keeps track of how many of its licenses are checked out, and which users have them.

A FlexEnabled product communicates with a vendor daemon, usually through TCP/IP network communications. The FlexEnabled product and the daemon processes (the license server) can run on separate machines on your network, across any size wide-area network. The format of the traffic between the FlexEnabled product and the vendor daemon is machine independent, allowing for heterogeneous networks. The license server and the computer running a product can be either different hardware platforms or even different operating systems (Windows and UNIX, for example).

Debug Log File

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

Administrative Options File

The options file enables the license administrator to control various operating parameters of FlexNet Licensing within the limits of the product's licensing policy. The license administrator can:

- Administer named-user and named-host licenses:
    - Allow features to be used by specific users, hosts, and/or groups
    - Deny feature use to specific users, hosts, and/or groups
    - Reserve licenses for specific users, hosts, and/or groups
- Restrict the number of licenses available.
- Control the amount of information logged about license usage.
- Enable a report log file.

Report Log File

The report log file contains feature usage information and is generated by the vendor daemon. Report log output is encrypted and can be used by reporting products such as Flexera Software's FlexNet Manager.

License Request Process

The license request process differs based on whether the license is requested in a served or an unserved environment.

Unserved License Model Request Process

Unserved license models depend solely on the FlexNet Publisher client library routines in the application to grant or deny usage based solely upon the content of the license file.

Served License Model Request Process

When the FlexEnabled application runs in the context of a served license model, the following occurs:

1. The license module in the FlexEnabled application finds the host name of the license server machine and TCP/IP port number of the license server manager (`lmadmin` or `lmgrd`) from the license file, from an environment variable, or from a registry entry. The last two options are set by the end user for their machine.
2. The FlexEnabled application establishes a connection with the license server manager and tells it which vendor daemon it needs to talk to.
3. The license server manager determines which machine and TCP/IP port correspond to the master vendor daemon and sends that information back to the FlexEnabled application.
4. The FlexEnabled application establishes a connection with the specified vendor daemon and sends its request for a license.

5. The vendor daemon checks in its memory to see if any licenses are available and sends a grant or denial back to the FlexEnabled application.
6. The FlexNet Publisher client library routines in the application grant or deny use of the feature, as appropriate.

3

Overview of the FlexNet Publisher Licensing Toolkit

This chapter provides an overview of the FlexNet Publisher toolkit directory structure and the components that are required to perform the evaluations described in the following chapters.

FlexNet Publisher Components

The toolkit contains the following components that are used in FlexNet Publisher evaluations. Some required components differ, based on whether you are evaluating FlexNet Publisher on a system running Windows or UNIX.

Table 3-1 • FlexNet Publisher Components

Component	Windows	UNIX
Sample license files	• `counted.lic` • `expired.lic` • `uncounted.lic`	• `counted.lic` • `expired.lic` • `uncounted.lic`
Sample vendor daemon	`demo.exe`	`demo`
Sample FlexEnabled application	`lmflex.exe`	`lmflex`
License signing utility	`lmcrypt.exe`	`lmcrypt`
FlexNet Publisher utilities	• `lmtools.exe` • `lmutil.exe`	`lmutil`
DEMO license server	Composed of `lmadmin` (or `lmgrd`) and the `demo` vendor daemon	Composed of `lmadmin` (or `lmgrd`) and the `demo` vendor daemon

Identifying the Toolkit Directories

The evaluation procedures in the following chapters assume you have installed FlexNet Publisher, as described in the *Installation Guide*. Table 3-2 lists the directories that are involved in the evaluation exercises, along with their shortcuts. The shortcuts are used throughout the documentation.

Table 3-2 • Toolkit Directories

Shortcut Name	Default Location on Windows	Default Location on UNIX
`install_dir`	No default location on Windows or UNIX: You define the installation path when you install the toolkit.	
`platform_dir`	`<install_dir>\<platform_dir>`	`<install_dir>/<platform_dir>`
`machind`	`<install_dir>\machind`	`<install_dir>/machind`
`examples`	`<install_dir>\examples`	`<install_dir>/examples`

All evaluation operations take place in the `<platform_dir>` directory, unless otherwise noted.

Examining the Sample License Files

FLEXnet Publisher includes three sample license files—`counted.lic`, `expired.lic`, and `uncounted.lic`—which are used in the evaluation exercises. These files are located in the `<platform_dir>` directory. You can open the files using a text editor.

Uncounted License File

Open the file `uncounted.lic` in a text editor. The license file text is similar to the following:

```
FEATURE f2 demo 1.000 permanent uncounted HOSTID=ANY  \
    SIGN="0006 9603 9C50 609D 13F4 353F 4866 A300 739D \
    05F9 2D83 2B0C 0242 9542 A15F"
```

This license is for feature `F2` (up to version `1.000`) and is licensed to the vendor `demo`. The inclusion of the `permanent` keyword indicates that the license will not expire. It is `uncounted` and, therefore, does not require a license server. It is valid on any machine as `HOSTID=ANY`, which means that the feature can be used on any system.

The following figure shows where these items are located in the license file.

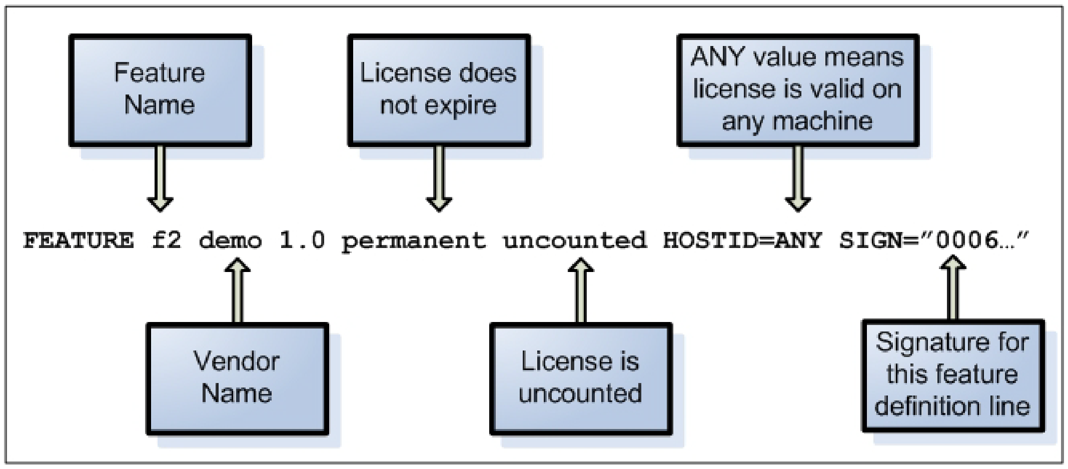

Feature Name
License does not expire
ANY value means license is valid on any machine
FEATURE f2 demo 1.0 permanent uncounted HOSTID=ANY SIGN="0006..."
Vendor Name
License is uncounted
Signature for this feature definition line

Figure 3-1: Uncounted FEATURE line

Served License File

Open the file `counted.lic` in a text editor. The license file text is similar to the following:

```
SERVER this_host ANY
VENDOR demo
USE_SERVER
FEATURE f1 demo 1.0 permanent 4 SIGN="AADD B062 54D1 47E3 \
    1FF6 145F 59CE 0200 8002 FE69 0FBC 0000 AA10 FFB3 6EB3"
```

This sample license file is valid on any machine because on the SERVER line, the server hostid is ANY. It contains one FEATURE line for `f1` (version `1.0`) that has a license count of `4` and is served by the vendor daemon `demo`.

The following figure shows where these items are located on the FEATURE line.

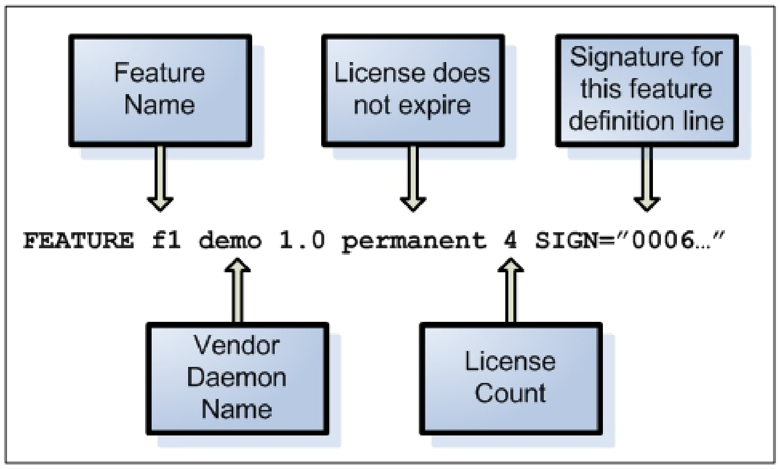

Feature Name
License does not expire
Signature for this feature definition line
FEATURE f1 demo 1.0 permanent 4 SIGN="0006..."
Vendor Daemon Name
License Count

Figure 3-2: Counted FEATURE line

Expiring License File

Open the file `expired.lic` in a text editor. The license file text is similar to the following:

```
FEATURE f4 demo 1.000 01-jan-2001 uncounted HOSTID=ANY \
    SIGN="0051 AC3E 1A55 609D 13F4 353D 4864 A301 719B \
    68F9 2D85 2B0C 0242 B5FE B55B"
```

This license is for feature `F4` (up to version `1.000`) and is licensed to the software publisher named `demo`. It is an expiring license that was set to expire on `01-jan-2001`. It is `uncounted` and, therefore, does not require a license server. It is valid on any machine as `HOSTID=ANY`.

The expiration date of this license is incorporated into the license signature for feature f4. Any attempt to modify the expiration date to extend the life of the license will invalidate the license because the feature’s signature cannot be authenticated. An attempt to check out the feature will result in a denial.

4

Evaluating FlexNet Publisher on Windows

This chapter walks you though a set of tutorial exercises that evaluate FlexNet Publisher licensing behavior on a Windows platform when licensing is based on license files. The exercises provide evaluations from the perspectives of the end user, the license administrator, and the software publisher.

Before you start this chapter, install FlexNet Publisher as described in the instructions in the *Installation Guide*.

All operations in this chapter take place relative to the `<platform_dir>` directory (such as `x64_n6`) within the toolkit installation directory (`<install_dir>`) unless otherwise noted.

Important • *Later exercises in this chapter use your demo keys (provided by Flexera Software) to rebuild the FlexNet Publisher toolkit. If you plan on performing these later exercises, note the following:*

- *A supported version of Microsoft Visual Studio is required to rebuild the toolkit for this part of the evaluation. See the latest* FlexNet Publisher Licensing Toolkit Release Notes *for an up-to-date list of the versions supported. (Windows Itanium requires a specific compiler, also specified in the* Release Notes.
- *If your demo vendor keys expire before you complete the evaluation, contact your Flexera Software salesperson.*

Evaluating the License-Administrator and End-User Experience

The evaluation exercises in this section take you through different actions that end users or license administrators at your customer sites commonly perform. Typical scenarios are demonstrated in the following sections:

Unserved license model demonstration:

- Using an Uncounted License

Served license model demonstration:

- Starting the License Server

- Checking the Status of the Demo License Server
- Using a Floating License
- Attempting to Check Out an Unlicensed Feature

All components required to perform the steps in this evaluation are included with the FlexNet Publisher toolkit (with the exception of a general Windows text editor). See Identifying the Toolkit Directories in Chapter 3, “Overview of the FlexNet Publisher Licensing Toolkit,” for the component locations.

Unserved License Model Demonstration

This demonstration of the unserved license model uses the following FlexNet Publisher components:

- `lmflex.exe`—Licensing toolkit test program
- `uncounted.lic`—License file used for unserved licenses for `lmflex`

Using an Uncounted License

Follow these steps to start the `lmflex` test program, and use it to check out a license for the feature **f2** using the license file `uncounted.lic`.

Task: ***To check out a license for feature f2***

1. Open a command window, and change to the `<install_dir>\<platform_dir>` directory.
2. At the prompt, enter `lmflex`. The following output is displayed:

   ```
   Enter "f1" to demo floating functionality
   Enter "f2" to demo node-locked functionality
   Enter feature to checkout [default: "f1"]:
   ```

3. Type `f` and `2`, and press **Enter** to check out a license for feature **f2**.

   If you have not used the licensing toolkit on this machine before, the FlexNet License Finder dialog is displayed. Do the following to browse to the file `uncounted.lic`:

   a. Select **Specify the License File**, and click **Next**.
   b. Type `uncounted.lic` or browse to this file, and click **Next**.
   c. Click **Finish**.

   The following message is displayed, confirming that a license for feature **f2** has been checked out:

   ```
   f2 checked out...press return to exit...
   ```

4. Press **Enter** to check the **f2** license back in and to exit `lmflex`.

***Note** • FlexNet License Finder dialog is used in Step 3 because `lmflex` does not have a facility that allows you specify the locations of license files before they are needed. If `lmflex` is accessing a specific license file for the first time, it prompts you for the license file location. Once the license file is used to check out a license successfully, `lmflex` stores the license-file path and uses it for subsequent license searches.*

Served-License Model Demonstration

The exercises in this section demonstrate the served-license model. They use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex.exe`
- Sample license file `lmadmin\demo\demo.lic` for use with the license server
- The license server—`lmadmin\demo\demo.exe` (the vendor daemon) and lmadmin\`lmadmin.exe` (the license server manager)

  ***Note** • For purposes of the exercises that follow, you must copy the vendor daemon `demo.exe` and its associated library from the `<install_dir>\<platform_dir>` directory to the `<install_dir>\<platform_dir>\lmadmin\demo` directory.*

The exercises use both `lmadmin`'s user interface and its command-line functionality to manage licenses.

Starting the License Server

Starting the license server involves importing the vendor daemon and starting `lmadmin`. For initial vendor-daemon startup, use `lmadmin` command line, as described in this exercise.

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

1. For purposes of the exercises that follow, copy `demo.exe` and `demo_libFNP.dll` from the `<install_dir>\<platform_dir>` directory to the `<install_dir>\<platform_dir>\lmadmin\demo` directory.
2. Open a new command window.
3. At a command prompt from the `<platform_dir>\lmadmin` directory, enter the following to import the `demo` vendor daemon:

   ```
   lmadmin -import demo\demo.lic -force
   ```

4. From the command prompt, enter the following to start the license server in the background and ensure that you can stop the license server from your machine:

   ```
   lmadmin -adminOnly no -allowRemoteStopServer yes
   ```

5. To access the `lmadmin` user interface, open your Web browser, and enter the following URL:

   `http://<server>:8090`

   where

   - `<server>` is the system name where the license server is running. For this demo, use `localhost`.
   - `8090` is the port for the `lmadmin` user interface. If this is not the port you have assigned, specify the correct one in the command.

6. From the `lmadmin` user interface's main window, click **Administration** to open the Administration page, and sign in with your administrator credentials.

7. Go to the Vendor Daemon Configuration tab.

   The vendor daemon status should display as **Up**.

   Important • *Keep the `lmadmin` user interface open for activities in the upcoming exercises.*

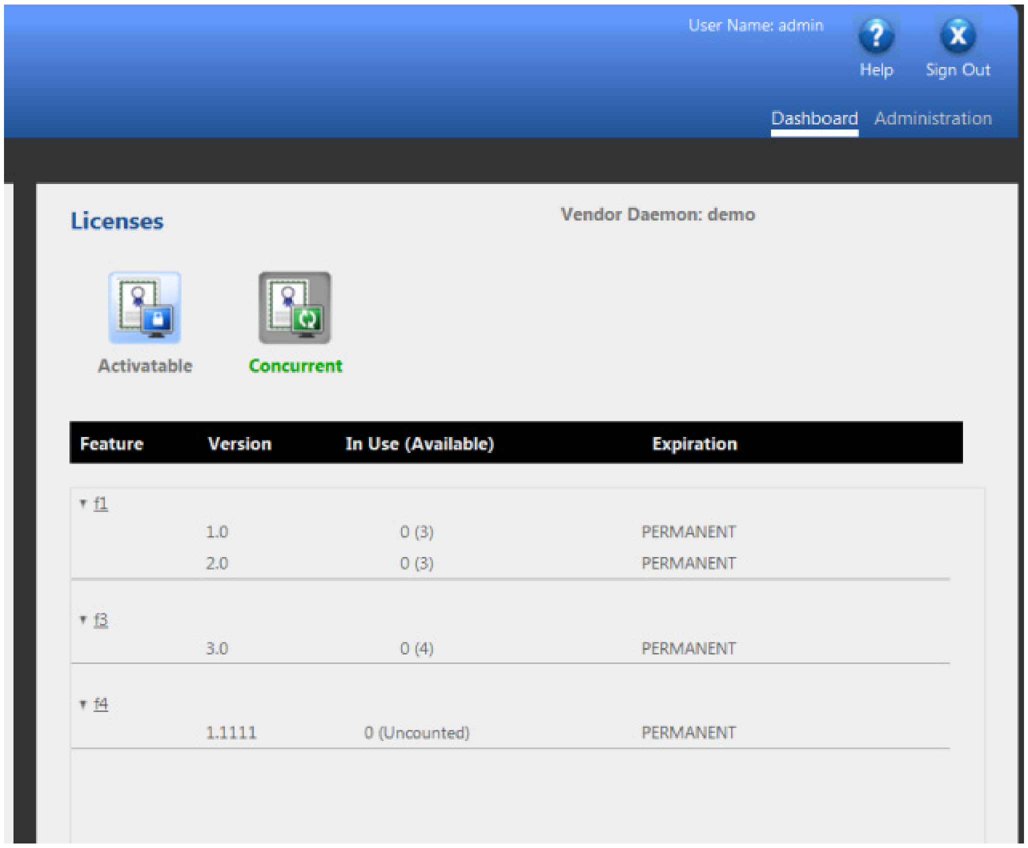
Checking the Status of the Demo License Server

Use the `lmadmin` user interface to check the status of the `demo` license server and determine the following:

- The number of licenses issued
- The number of licenses in use
- The users of each FlexEnabled feature

Task: ***To check the license server status***

1. From the `lmadmin` user interface's main window, click **Dashboard** to open the Dashboard page. This page displays information about the current status of the license server.
2. Click **Concurrent** to show current floating license activity:



Vendor Daemon: demo

Feature	Version	In Use (Available)	Expiration
f1			
	1.0	0 (3)	PERMANENT
	2.0	0 (3)	PERMANENT
f3			
	3.0	0 (4)	PERMANENT
f4			
	1.1111	0 (Uncounted)	PERMANENT

The display indicates the following:

- The vendor daemon running is `demo`.
- No users have any of the concurrent licenses for feature **f1** checked out.

Using a Floating License

Use the following exercise to start the test application, `lmflex.exe`, and use it to check out a concurrent (floating) license for the feature **f1** from license server.

Important • *To complete this section, make sure that you have completed the previous sections (that is, the* `demo` *license server must be configured properly and running).*

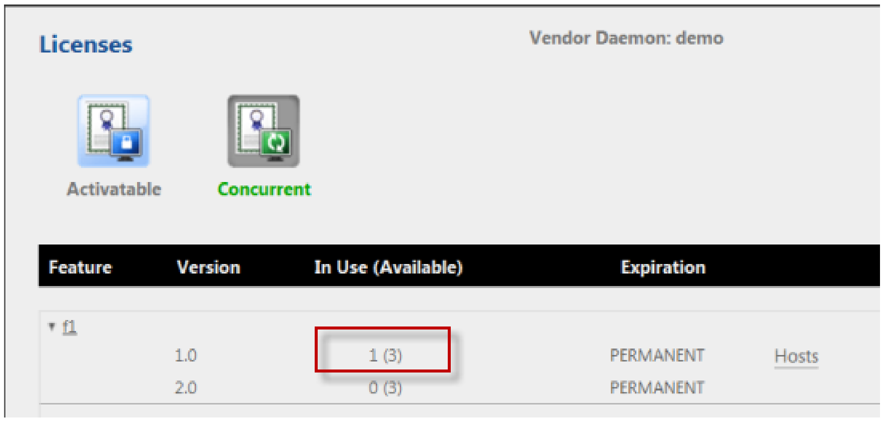
Task: ***To demonstrate the use of a floating license***

1. Open a new command window, and change to `<install_dir>\<platform_dir>`.
2. At the command prompt, enter `lmflex`. The feature checkout options are displayed.
3. Press **Enter** to check out a license for feature **f1** (the default checkout feature). The following confirmation message is displayed:

```
f1 checked out...press return to exit...
```

4. To check the status of the license server with a license checked out, go to the Dashboard page on the main window of the `lmadmin` user interface, and click **Concurrent** to refresh (or go to) the Concurrent display.

   The display shows that one **f1** floating license (out of three for Version 1.0) is in use:



Feature	Version	In Use (Available)	Expiration	
▾ f1				
	1.0	1 (3)	PERMANENT	Hosts
	2.0	0 (3)	PERMANENT	

5. To view the users who have this feature checked out, do the following:
   a. Click **Hosts** next to the checked-out license entry. A separate window opens, listing users who have checked out this feature. You should see your user name and machine as an entry.
   b. Close the Hosts window.
6. To view the entry for this checked-out license in the debug log, do the following:
   a. From the `lmadmin` user-interface main window, click **Administration** to go to the Administration page.
   b. Open the Vendor Daemon Configuration tab.
   c. Within the `demo` entry, click **Administer**.
   d. Click the **Vendor Daemon Log** section header.
   e. Click **View file externally**. The debug log content is displayed in another window. The entry at the end of the debug log shows the latest checkout action, similar to the following:

```
19:03:54 (demo) OUT:"f1" daniel@myhost
```

   f. Close the debug log window.
7. In the command window running `lmflex`, press **Enter** to check the **f1** license back in and to exit `lmflex`.

***Important** • Keep this command window open to perform more `lmflex` activities in the upcoming exercises.*

8. To verify that no licenses are currently checked out, go to the Dashboard page in the `lmadmin` user-interface main window, and click **Concurrent**. The display should show **0** (out of three licenses) checked out.

Attempting to Check Out an Unlicensed Feature

End users cannot use a feature for which they do not have a license. In this exercise, you attempt to check out a feature named **f5**, which is not included in the `demo.lic` license file.

Task: ***To attempt to check out an unlicensed feature using lmflex***

1. In the command window in which you run `lmflex`, enter `lmflex` at a prompt from the `<platform_dir>` directory.
2. Type `f` and `5` to indicate the feature you want to check out, and press **Enter.**

   An error dialog appears, telling you that the feature is not supported.
3. Click **OK** to dismiss the dialog.

Evaluating the Software Publisher Experience

This section of the evaluation continues the scenario started in the previous exercises that demonstrated end-user and license-administrator experiences. However, the exercises in this section simulate the publisher experience. They demonstrate some of the activities you perform in the course of implementing your license policy-- and show the versatility that FlexNet Publisher's options offer in developing this policy. The exercises include the following:

- Licensing Individual Features
- Testing Node-Locked, Uncounted Licenses
- Testing an Expiring License
- Building the FlexNet Publisher Toolkit

All components required to perform the steps in this evaluation are included with the FlexNet Publisher toolkit (with the exception of a general Windows text editor). See Identifying the Toolkit Directories in Chapter 3, "Overview of the FlexNet Publisher Licensing Toolkit," for the component locations.

Licensing Individual Features

To license a feature for usage, the feature must be identified in the license file in a `FEATURE` or `INCREMENT` line. Because you can specify license rights on a feature-by-feature basis, you can distribute the same binary to multiple customers, but define license rights using different license files for different customers.

The exercises in this section describe how to add a license for a feature to an existing license file and how to re-sign the license file so the feature definition lines are valid.

Required Components

These exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex.exe`
- Sample license file `lmadmin\demo\demo.lic` for use with the license server
- The license server—`lmadmin\demo\demo.exe` (the vendor daemon) and lmadmin`\lmadmin.exe` (the license server manager)

  ***Note** • The license server should already be running from the previous tutorials.*

- The license-signing utility `lmcrypt`
- The utility `lmreread`

The exercises also require a general text editor (such as Windows Notepad).

Editing the License File

In the previous exercise, the end user was unable to perform a checkout for the feature called **f5** because no license existed for that feature in the license file. To license a feature named **f5**, you must edit the sample license file to add a feature definition line.

Task: ***To edit the sample license file***

1. Open `<install_dir>\<platform_dir>\lmadmin\licenses\demo\demo.lic` in a text editor.
2. Locate the **f3** FEATURE line, copy it, and paste it after the original **f3** FEATURE line.
3. In the second **f3** FEATURE line, replace `f3` with `f5`.
4. Save and close `demo.lic`.
5. In the command window in which you have been running `lmflex`, enter `lmflex` at a prompt from the `<platform_dir>` directory.

6. Attempt to check out **f5** by typing `f` and `5` and pressing **Enter**.

   An error dialog is displayed, indicating that you cannot check out the license.

7. Click **OK** to close the dialog.

You added an **f5** FEATURE line to the license file by copying the existing FEATURE line for **f3** and changing the feature name to **f5** in the new line. However, the signature associated with the FEATURE line for **f5** is still authenticated for feature **f3**. In the following section, you re-sign the license certificate so it is authenticated for **f5**.

Re-signing and Re-reading the License File

The previous exercise demonstrated that it is not possible simply to edit the license file, as an end user might try to do, to create a FEATURE line for **f5**. In this exercise, you use the utility `lmcrypt` to re-sign the license file, thus re-generating the license file to contain valid FEATURE lines for both **f3** and **f5**. To complete the process of adding a new feature, you must have the license server re-read the modified license file.

Task: ***To re-sign the license file***

1. To run `lmcrypt`, open another command window, and change to the `<platform_dir>` folder.

2. At the command line, enter the following:

   ```
   lmcrypt .\lmadmin\demo\demo.lic
   ```

   The resulting license file now contains authenticated FEATURE lines for **f3** and **f5** similar to the following:

   ```
   FEATURE f3 demo 1.0 permanent 4 SIGN="0008 0B78 63B1 1A00 DCE8 132C \
      C70D 4A00 33E4 B817 F307 5041 8D0D 7A7A 2E47"
   FEATURE f5 demo 1.0 permanent 4 SIGN="00A3 42AC C1D2 8B9A 5C0A 7552 \
      B69D 6100 9EEC 4178 04C0 C49D 3270 A192 AA25"
   ```

3. Close the command window.

4. In the `lmadmin` user-interface main window, click **Administration** to open the Administration page.

5. Open the Vendor Daemon Configuration tab.

6. In the demo entry, click **Administer**.

7. Click `Reread License Files`. Wait until the reading is complete.

8. In the command window in which you have been running `lmflex`, restart `lmflex` and try to check out **f5**. Now that you have a valid license for **f5,** you should be able to check out its license.

9. Press **Enter** to check the license back in and to exit `lmflex`.

Stopping the License Server

The remaining publisher-specific exercises in this chapter pertain to activities for managing unserved licenses. Use the following procedure to shut down the license server from the `lmadmin` user interface.

Task: ***To stop the license server***

1. From the Administration page in the `lmadmin` user-interface main window, open the Server Configuration tab.
2. Click the **License Server Configuration** section header.
3. Click **Stop Server**.
4. Once the license server has stopped, close the `lmadmin` user interface.
5. Close the command window with which you started `lmadmin` and kept open until now for monitoring purposes.

Testing Node-Locked, Uncounted Licenses

This section demonstrates the process of locking an unserved license to a specific machine by simply providing the machine's identifier (`HOSTID`) in the `FEATURE` line and re-encrypting the license file. This process incorporates the `HOSTID` into the license signature. It is not necessary to modify either the toolkit or the FlexEnabled application to incorporate this license policy.

The following exercises describe how to node-lock the license for feature **f4** to your specific machine and test the license.

Required Components

The exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex.exe`
- Sample license file `lmadmin\demo\demo.lic`
- The license-signing utility `lmcrypt`
- The utility `lmhostid` (one of the `lmutil` utilities)

The exercises also require a general text editor (such as Windows Notepad).

Creating Node-Locked, Uncounted Licenses

To node-lock a license to your machine, you need to obtain the hostid for your machine and then incorporate this information into the signature for the **f4** license in the license file.

Task: ***To create the node-locked, uncounted license***

1. Open a new command window, and change to the `<install_dir>\<platform_dir>` directory.
2. Enter the following command to obtain the default hostid for your machine:

```
lmutil lmhostid
```

For Windows platforms, the output value is the machine's ethernet address. The following shows sample output from the command:

```
The FlexNet host ID of this machine is "000103e686e6"
```

3. Copy `lmadmin\demo\demo.lic` to the `<platform_dir>` directory.
4. Open `<platform_dir>` `\demo.lic` in a text editor.
5. Locate the `FEATURE` line for feature **f4**, and replace the current `HOSTID` value with the value of your machine's hostid. The feature definition line now looks something like:

```
FEATURE f4 demo 1.111 permanent uncounted
         HOSTID=000103e686e6 SIGN="00DA...FBE5"
```

6. Save and close `<platform_dir>` `\demo.lic`.
7. At the command prompt from the `<platform_dir>` directory, enter the following to re-sign the license file:

```
lmcrypt demo.lic
```

8. Close the command window.

Testing Licenses

Run `lmflex` and try to check out feature **f4**. A license for feature **f4** is successfully granted because the application is executing on the machine to which the license is locked. Check in the feature **f4.**

Because the license file for **f4** is in the same directory as `lmflex` and has an extension of `.lic`, its license is found and used. The license for **f4** is uncounted and therefore does not require a license server to allow it to run.

Security Considerations

As previously explained, the `HOSTID` value is incorporated in the license signature and used to authenticate the license. Any changes to the license signature, the license `HOSTID=`*`value`*, or the actual machine hostid invalidates the license. As a result, the FlexEnabled application fails authenticate the license, and any license requests fail. The following exercise demonstrates this concept.

Task: ***To demonstrate security considerations***

1. Reopen `<platform_dir>\demo.lic` in a text editor.
2. Locate the `FEATURE` line for feature **f4**, and change the value of `HOSTID` to a made-up value, simulating an unauthorized attempt to rehost the license. The feature definition line now looks similar to this:

```
FEATURE f4 demo 1.111 permanent uncounted
    HOSTID=000111111111 SIGN="00E3...EA16"
```

Important • *Do not re-sign the license.*

3. Run `lmflex` and try to check out feature **f4**. A FlexNet Licensing error dialog box appears to indicate that the license request is denied because the application detects the inconsistency between the `HOSTID=`*`value`* in the feature definition line and that of the actual machine used to create the signature.
4. Click **OK** to dismiss the dialog box.
5. Delete `demo.lic` from the `<platform_dir>` directory.

Important • *Make sure you delete the correct license file. Do not delete* `demo.lic` *from the* `lmadmin\demo` *directory.*

This last sequence demonstrates FlexNet Licensing's policy in the license technology. You, the software publisher, define the policy—in this case, locked to a specific machine—and create the license. You are in control of the license usage until you change it.

Testing an Expiring License

This section demonstrates how to set an license-expiration policy. To implement the policy, simply set an expiration date in the FEATURE line; you do not need to modify the licensing toolkit or the licensed application. When the license is signed, the expiration date is incorporated into the license signature. In the following steps, you modify the expiration date of the node-locked feature **f4** located in the `expired.lic` license file.

The expiration date is incorporated in the license signature and used for license authentication. Therefore, attempting to extend a license by simply moving the expiration date forward does not create a legitimate license. When a user moves the date forward, the FlexEnabled application fails to authenticate the license, and any license requests for the feature fail.

Required Components

This exercise uses the following FlexNet Publisher toolkit components:

- Sample test program `lmflex.exe`
- Sample license file `expired.lic`
- The license-signing utility `lmcrypt`

The exercise also requires a general text editor (such as Windows Notepad).

Testing the Expiring-License Policy

Task: ***To demonstrate an expiring-license policy***

1. Make sure that you have deleted the `demo.lic` file from the `<platform_dir>` directory.
2. In the command window in which you run `lmflex`, enter `lmflex` at a prompt from the `<platform_dir>` directory.
3. In the `<platform_dir>` directory, copy `expired.lic` to a file called `expired_lic.orig` to preserve the original license.
4. Open `<platform_dir>\expired.lic` in a text editor.
5. Locate the FEATURE line for feature **f4**, and change the expiration date to a date in the future, as shown in the following example:

```
FEATURE f4 demo 1.000 31-jan-2021 uncounted
    HOSTID=ANY SIGN="0051...B55B"
```

6. Save and close `expired.lic`.

Important • *Do not re-sign the license.*

7. Run `lmflex` and try to check out feature **f4**. Because the expiration date is different than the date encoded in the license signature, a FlexNet Licensing error dialog appears, stating that the license request is denied because the signature cannot be authenticated.
8. Click **OK** to closed the error dialog.
9. Open a new command window, and change to the `<platform_dir>` directory.
10. Re-sign the license file to incorporate the new expiration date into the license signature. To do so, enter the following at the command prompt:

```
lmcrypt expired.lic
```

11. Run `lmflex` and try to check out feature **f4**. A license for feature **f4** is successfully granted because the license has not expired.

    Because the license file for **f4** is in the same directory as `lmflex` and has an extension of `.lic`, its license is found and used. The license for **f4** is uncounted and therefore does not need a license server to allow it to run.
12. Press **Enter** to exit `lmflex` and to check in the license for feature **f4**.
13. Close all command windows.

Building the FlexNet Publisher Toolkit

In this section, you rebuild the FlexNet Publisher toolkit to change the strength of tamper-resistant licenses (TRL) authentication from 113 bits to 163 bits. TRL uses public-key technology from Certicom to make the signatures on `FEATURE` and `INCREMENT` lines more difficult to counterfeit. As shipped, the toolkit has TRL enabled at the 113-bit strength, denoting a public-key length of 113 bits.

Required Components

These exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex.exe`
- Sample license file `counted.lic`
- The build configuration file `lm_code.h`
- The license-signing utility `lmcrypt`

Additionally, you need the following components:

- Microsoft Visual Studio Command Prompt window for the build process (see the latest *FlexNet Publisher Licensing Toolkit Release Notes* for supported compiler versions)
- A general text editor (such as Windows Notepad)

Examining the Current License Signature Length

Task: ***To examine the current license signature length***

1. Open `<platform_dir>\counted.lic` in a text editor.
2. Observe the length of the signature (`SIGN`) value in the `FEATURE` line for **f1**. Based on a 113-bit strength, the value should have a length similar to the following:

```
FEATURE f1 demo 1.0 permanent 4 SIGN="003D B054 54D1 47E3 1FF6 \
    145F 59CE 0200 8002 FE69 0FBC F4FF AA10 EEB3 6EB3"
```

3. Optionally, save a copy of this file for future comparison.
4. Close `counted.lic`.

Setting Up the Microsoft Visual Studio Development Environment

***Note** • To build the FlexNet Publisher on a Windows platform (except for Windows Itanium), you must use a supported version of Microsoft Visual Studio. Refer to the latest* FlexNet Publisher Release Notes *for an up-to-date list of these versions. The Windows Itanium platform requires a specific compiler, also listed in the* Release Notes.

The Licensing toolkit is built using the Microsoft Visual Studio Command Prompt rather than the integrated development environment (IDE). Make sure that you have the Microsoft Visual Studio Command Prompt development environment correctly configured. Consult the Microsoft documentation for details about building from the command line.

Configuring the FlexNet Publisher Toolkit

For previous exercises, you used the prebuilt, “out of the box” FlexNet Publisher components. However, as shipped, the FlexNet Publisher toolkit is in an unconfigured state. You must configure it in order to rebuild it. Use the `demo` vendor keys provided to you by Flexera Software for the following configuration procedure.

Task: ***To configure the FlexNet Publisher Toolkit***

1. Open the `<install_dir>\machind` folder.
2. Locate the file `lm_code.h`, and open it in a text editor. Find the lines similar to the following:

```
#define VENDOR_KEY1 0x0
#define VENDOR_KEY2 0x0
#define VENDOR_KEY3 0x0
#define VENDOR_KEY4 0x0
#define VENDOR_KEY5 0x0
```

3. Replace all five instances of `0x0` with the demo vendor keys that you received from Flexera Software. If your demo vendor keys have expired, contact Flexera Software.
4. Keep `lm_code.h` open.

Changing TRL Strength

Task: ***To change the TRL strength***

1. In `lm_code.h`, find the `LM_STRENGTH` definition that looks similar to the following:

```
#define LM_STRENGTH LM_STRENGTH_113BIT
```

2. Redefine `LM_STRENGTH` to 163-bit strength, as shown:

```
#define LM_STRENGTH LM_STRENGTH_163BIT
```

3. Leave `TRL_KEYS` set to `0x0`. They are not set when using demo vendor keys.
4. Save and close `lm_code.h`.

Building the FlexNet Publisher Toolkit

For your changes to take effect, you need to build the FlexNet Publisher toolkit.

Important • *Make sure `lmflex` is closed and the license server is stopped before you attempt to rebuild the FlexNet Publisher toolkit.*

Task: ***To build your Licensing toolkit***

1. In the Microsoft Visual Studio Command Prompt window, change to the `<platform_dir>` folder.
2. At the command prompt, enter `build`.

   The `build.bat` file calls `nmake`, which in turn builds the `makefile`.

The demo toolkit now supports TRL at the 163-bit strength level. Because this modification changes the authentication method, all components—including the sample application, `lmflex`—have been rebuilt.

Re-signing the Sample License File

Regenerate the license file to create license signatures that support TRL authentication at the 163-bit level.

Task: ***To regenerate the license file***

1. Open a regular Windows command window.
2. At a prompt from the `<platform_dir>` directory, enter the following:

   ```
   lmcrypt counted.lic
   ```

3. Open `counted.lic` in a text editor.
4. Compare the length of the `SIGN` value in the **f1** `FEATURE` line with the length of the `SIGN` value that you examined before the toolkit rebuild. The new signatures are longer, similar to the following:

   ```
   FEATURE f1 demo 1.0 permanent 4 SIGN="01C6 1045 146D AE6C 92CB 0360 \
       1124 AC73 B79B 1785 C000 9608 773C EAAF 3815 55DF 0FAF 786A \
       AA19 E123 52A3"
   ```

5. Close `counted.lic`.

Conclusion

You have completed the evaluation of FlexNet Publisher on Windows. When you are ready to build your Licensing toolkit with your production vendor keys, see the manual *Development Environment Guide*.

To evaluate FlexNet Publisher on UNIX, see Chapter 5, "Evaluating FlexNet Publisher on UNIX."

Important • *Remember to set* `LM_STRENGTH` *to an appropriate value before you build your production licensing toolkit.*

5

Evaluating FlexNet Publisher on UNIX

This chapter walks you though a set of tutorial exercises that evaluate FlexNet Publisher licensing behavior on a UNIX platform when licensing is based on license files. The exercises provide evaluations from the perspectives of the end user, the license administrator, and the software publisher.

Before you start this chapter, install FlexNet Publisher as described in the instructions in the *Installation Guide*.

All operations in this chapter take place relative to the `<platform_dir>` directory (such as `sun4_u5`) within the toolkit installation directory (`<install_dir>`) unless otherwise noted.

Important • *Later exercises in this chapter use your demo keys (provided by Flexera Software) to rebuild the FlexNet Publisher toolkit. If you plan on performing these later exercises, note the following:*

- *A C development environment is required to rebuild the toolkit for this part of the evaluation. Consult the FlexNet Publisher Release Notes for your platform's C development environment requirements.*
- *If your demo vendor keys expire before you complete the evaluation, contact your Flexera Software salesperson.*

Evaluating the License-Administrator and End-User Experience

The evaluation exercises in this section take you through different actions that end users or license administrators at your customer sites commonly perform. Typical scenarios are presented for starting the license server, checking its status, and checking out licenses for features in the demo FlexEnabled application.

Required Components

The exercises in this section demonstrate the served-license model. They use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex`
- Sample license file `lmadmin/demo/demo.lic` for use with the license server
- The license server—`lmadmin/demo/demo` (the vendor daemon) and `lmadmin/lmadmin` (the license server manager)

The exercises use both `lmadmin`'s user interface and its command-line functionality to manage licenses.

Note • *For purposes of these exercises, you must copy the vendor daemon* `demo` *and its associated library from the* `<install_dir>/<platform_dir>` *directory to the* `<install_dir>/<platform_dir>/lmadmin/demo` *directory. See the procedure that follows for details.*

See Identifying the Toolkit Directories in Chapter 3, "Overview of the FlexNet Publisher Licensing Toolkit," for the component locations.

Starting the License Server

Starting the license server involves importing the vendor daemon and starting `lmadmin`. For initial vendor-daemon startup, use the command line, as described in this exercise.

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

1. For purposes of the exercises that follow, copy `demo` and `demo_libFNP.so` from the `<install_dir>/<platform_dir>` directory to the `<install_dir>/<platform_dir>/lmadmin/demo` directory.

   Note • *On AIX, copy* `demo` *and* `demo_server_libFNP_notr.so`.

2. In a command window, set the license file path that the `demo` vendor daemon uses to the `<platform_dir>/lmdamin` directory:

   ```
   setenv DEMO_LICENSE_FILE <platform_dir>/lmadmin
   ```

3. From the `<platform_dir>/lmadmin` directory, enter the following to import the `demo` vendor daemon:

   ```
   lmadmim -import demo/demo.lic -force
   ```

4. Enter the following to start the license server in the background:

   ```
   lmadmin -adminOnly no -allowRemoteStopServer yes
   ```

5. To access the `lmadmin` user interface, open your Web browser, and enter the following URL:

   ```
   http://<server>:8090
   ```

where

- `<server>` is the system name where the license server is running. For this demo, use `localhost`.
- `8090` is the port for the `lmadmin` user interface. If this is not the port you have assigned, specify the correct one in the command.

6. From the `lmadmin` user interface's main window, click **Administration** to open the Administration page, and sign in with your administrator credentials.
7. Go to the Vendor Daemon Configuration tab.

   The vendor daemon status should display as **Up**.

***Important** • Keep the `lmadmin` user interface open for activities in the upcoming exercises.*

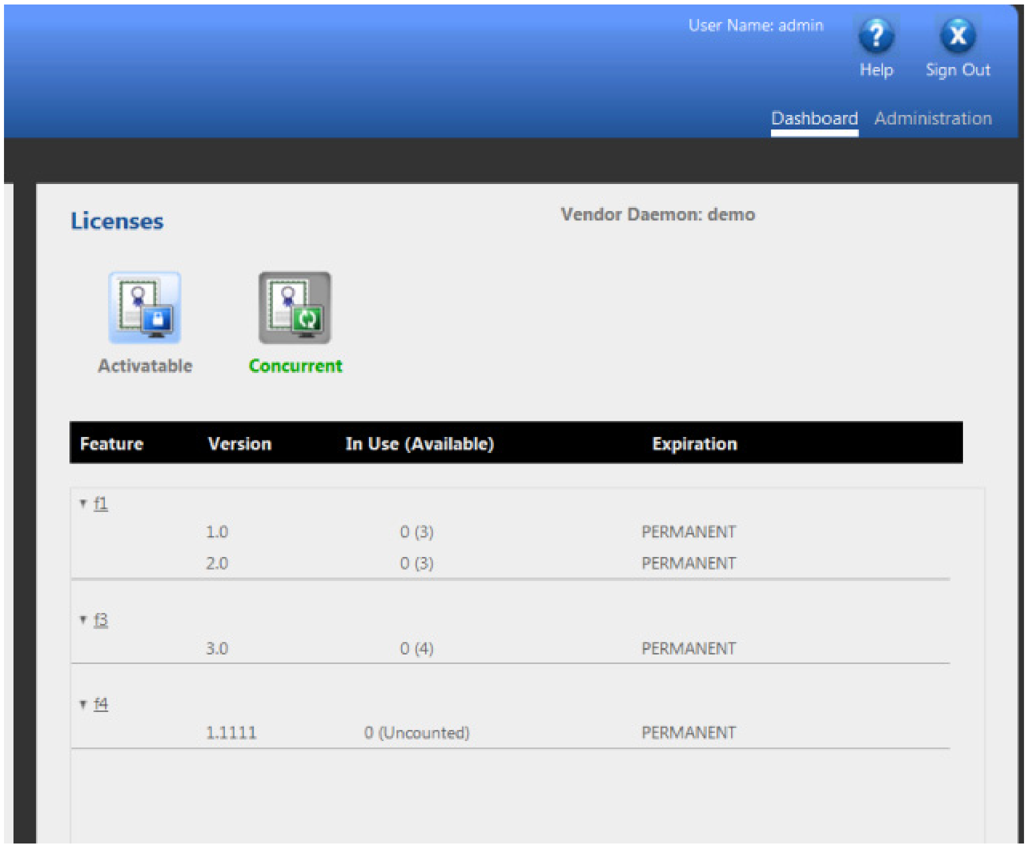
Checking the Status of the Demo License Server

Use the `lmadmin` user interface to check the status of the `demo` license server and determine the following:

- The number of licenses issued
- The number of licenses in use
- The users of each FlexEnabled feature

Task: ***To check the license server status***

1. From the `lmadmin` user interface's main window, click **Dashboard** to open the Dashboard page. This page displays information about the current status of the license server.
2. Click **Concurrent** to show current floating license activity:



Vendor Daemon: demo

Feature	Version	In Use (Available)	Expiration
▾ f1			
	1.0	0 (3)	PERMANENT
	2.0	0 (3)	PERMANENT
▾ f3			
	3.0	0 (4)	PERMANENT
▾ f4			
	1.1111	0 (Uncounted)	PERMANENT

The display indicates the following:

- The vendor daemon running is `demo`.
- No users have any of the concurrent licenses for feature **f1** checked out.

Using a Floating License

Use the following exercise to start the test application `lmflex` and use it to check out a concurrent (floating) license for the feature **f1** from license server.

Important • *To complete this section, make sure that you have completed the previous sections (that is, the* `demo` *license server must be configured properly and running).*

Task: ***To demonstrate the use of a floating license***

1. Set the link library path to the `<platform_dir>` directory:

```
% setenv LD_LIBRARY_PATH <platform_dir>
```

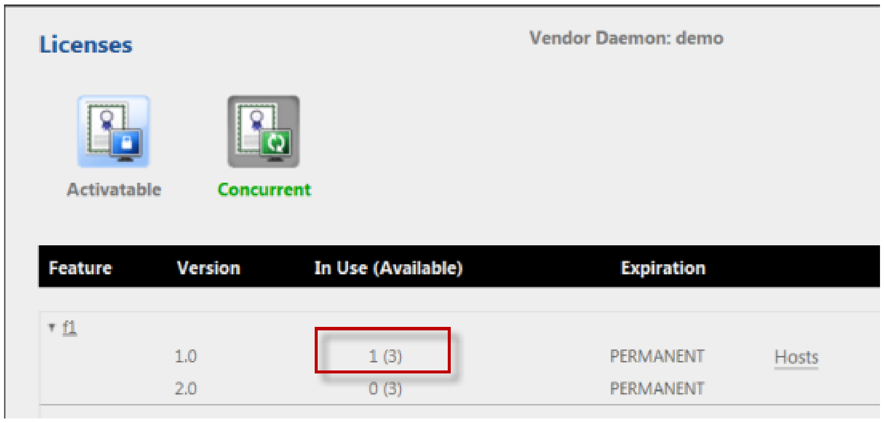
2. From the `<platform_dir>` directory, enter `lmflex`. The feature checkout options are displayed.

3. Press **Enter** to check out a license for feature **f1** (the default checkout feature). The following confirmation message is displayed:

```
f1 checked out...press return to exit...
```

4. To check the status of the license server with a license checked out, go to the Dashboard page on the main window of the `lmadmin` user interface, and click **Concurrent** to refresh (or go to) the Concurrent display.

   The display shows that one **f1** floating license (out of three for Version 1.0) is in use:



Feature	Version	In Use (Available)	Expiration	
▾ f1				
	1.0	1 (3)	PERMANENT	Hosts
	2.0	0 (3)	PERMANENT	

5. To view the users who have this feature checked out, do the following:

   a. Click **Hosts** next to the checked-out license entry. A separate window opens, listing users who have checked out this feature. You should see your user name and machine as an entry.

   b. Close the Hosts window.

6. To view the entry for this checked-out license in the debug log, do the following:

   a. From the `lmadmin` user-interface main window, click **Administration** to go to the Administration page.

   b. Open the Vendor Daemon Configuration tab.

   c. Within the `demo` entry, click **Administer**.

   d. Click the **Vendor Daemon Log** section header.

   e. Click **View file externally**. The debug log content is displayed in another window. The entry at the end of the debug log shows the latest checkout action, similar to the following:

   ```
   19:03:54 (demo) OUT:"f1" daniel@myhost
   ```

   f. Close the debug log window.

7. In the command window running `lmflex`, press **Enter** to check the **f1** license back in and to exit `lmflex`.

   Important • *Keep this command window open to perform more `lmflex` activities in the upcoming exercises.*

8. To verify that no licenses are currently checked out, go to the Dashboard page in the `lmadmin` user-interface main window, and click **Concurrent**. The display should show **0** (out of three licenses) checked out.

Attempting to Check Out an Unlicensed Feature

End users cannot use a feature for which they do not have a license. In this exercise, you attempt to check out a feature named **f5**, which is not included in the `demo.lic` license file.

Task: ***To attempt to check out an unlicensed feature using lmflex***

1. In the command window in which you run `lmflex`, enter `lmflex` at a prompt from the `<platform_dir>` directory.
2. Type `f` and `5` to indicate the feature you want to check out, and press **Enter.**

   An error dialog appears, telling you that the feature is not supported.
3. Click **OK** to dismiss the dialog.

Evaluating the Publisher Experience

This section of the evaluation continues the scenario started in the previous exercises that demonstrated end-user and license-administrator experiences. However, the exercises in this section simulate the publisher experience. They demonstrate some of the activities you perform in the course of implementing your license policy-- and show the versatility that FlexNet Publisher's options offer in developing this policy. The exercises include the following:

- Licensing Individual Features
- Testing Node-Locked, Uncounted Licenses
- Testing an Expiring License
- Building the FlexNet Publisher Toolkit

All components required to perform the steps in this evaluation are included with the FlexNet Publisher toolkit (with the exception of a general Windows text editor). See Identifying the Toolkit Directories in Chapter 3, "Overview of the FlexNet Publisher Licensing Toolkit," for the component locations.

Licensing Individual Features

To license a feature for usage, the feature must be identified in the license file in a `FEATURE` or `INCREMENT` line. Because you can specify license rights on a feature-by-feature basis, you can distribute the same binary to multiple customers, but define license rights using different license files for different customers.

The exercises in this section describe how to edit licenses for features in an existing license file and how to re-sign the license file so the feature definition lines are valid.

Required Components

These exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex`
- Sample license file `lmadmin/demo/demo.lic` for use with the license server
- The license server—`lmadmin/demo/demo` (the vendor daemon) and `lmadmin/lmadmin` (the license server manager)

  Note • *The license server should already be running from the previous tutorials.*

- The license-signing utility `lmcrypt`
- The utility `lmreread`

The exercises also require a general text editor.

Editing the License File

In the previous exercise, the end user was unable to perform a feature called **f5** because no license existed for that feature in the license file. To license a feature named **f5**, you must edit the sample license file to add a feature definition line.

Task: ***To edit the sample license file***

1. Open `<intall_dir>/<platform_dir>/lmadmin/licenses/demo/demo.lic` in a text editor.
2. Locate the **f3** FEATURE line, copy it, and paste it after the original **f3** FEATURE line.
3. In the second **f3** FEATURE line, replace `f3` with `f5`.
4. Save and close `demo.lic`.
5. In the command window in which you have been running `lmflex`, enter `lmflex` at a prompt from the `<platform_dir>` directory.
6. Attempt to check out **f5** by typing `f` and `5` and pressing **Enter**.

   An error dialog is displayed, indicating that you cannot check out the license.
7. Click **OK** to dismiss the dialog.

You added an **f5** FEATURE line to the license file by copying the existing FEATURE line for **f3** and changing the feature name to **f5** in the new line. However, the signature associated with the FEATURE line for **f5** is still authenticated for feature **f3**. In the following section, you re-sign the license certificate so it is authenticated for **f5**.

Re-signing and Re-reading the License File

The previous exercise demonstrated that it is not possible simply to edit the license file, as an end user might try to do, to create a FEATURE line for **f5**. In this exercise, you use the utility `lmcrypt` to re-sign the license file, thus re-generating the license file to contain valid FEATURE lines for both **f3** and **f5**. To complete the process of adding a new feature, you must have the license server re-read the modified license file.

Task: ***To re-sign the license file***

1. To run `lmcrypt`, open another command window, and change to the `<platform_dir>` folder.
2. At the command line, enter the following:

   ```
   lmcrypt ./lmadmin/demo/demo.lic
   ```

   The resulting license file now contains authenticated FEATURE lines for **f3** and **f5** similar to the following:

   ```
   FEATURE f3 demo 1.0 permanent 4 SIGN="0008 0B78 63B1 1A00 DCE8 132C \
      C70D 4A00 33E4 B817 F307 5041 8D0D 7A7A 2E47"
   FEATURE f5 demo 1.0 permanent 4 SIGN="00A3 42AC C1D2 8B9A 5C0A 7552 \
      B69D 6100 9EEC 4178 04C0 C49D 3270 A192 AA25"
   ```
3. Close the command window.
4. In the `lmadmin` user-interface main window, click **Administration** to open the Administration page.
5. Open the Vendor Daemon Configuration tab.
6. In the `demo` entry, click **Administer**.
7. Click `Reread License Files`. Wait until the reading is complete.
8. In the command window in which you have been running `lmflex`, restart `lmflex` and try to check out **f5**. Now that you have a valid license for **f5,** you should be able to check out its license.
9. Press **Enter** to check the license back in and to exit `lmflex`.

Stopping the License Server

The remaining publisher-specific exercises in this chapter pertain to activities for managing unserved licenses. Use the following procedure to shut down the license server from the `lmadmin` user interface.

Task: ***To stop the license server***

1. From the Administration page in the `lmadmin` user-interface main window, open the Server Configuration tab.
2. Click the **License Server Configuration** section header.
3. Click **Stop Server**.
4. Once the license server has stopped, close the `lmadmin` user interface.
5. Close the command window with which you started `lmadmin` and kept open until now for monitoring purposes.

Testing Node-Locked, Uncounted Licenses

This section demonstrates the process of locking an unserved license to a specific machine by simply providing the machine's identifier (`HOSTID`) in the `FEATURE` line and re-encrypting the license file. This process incorporates the `HOSTID` into the license signature. It is not necessary to modify either the toolkit or the FlexEnabled application to incorporate this license policy.

The following exercises describe how to node-lock the license for feature **f4** to your specific machine and test the license.

Required Components

The exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex`
- Sample license file `lmadmin/demo/demo.lic`
- The license-signing utility `lmcrypt`
- The utility `lmhostid` (one of the `lmutil` utilities)

The exercises also require a general text editor.

Creating Node-Locked, Uncounted Licenses

To node-lock a license to your machine, you need to obtain the hostid for your machine and then incorporate this information into the signature for the **f4** license in the license file.

Task: ***To create the node-locked, uncounted license***

1. Open a new command window, and change to the `<platform_dir>` directory.
2. Enter the following command to obtain the default hostid for your machine:

   ```
   lmutil lmhostid
   ```

   The output value shows the default hostid for your machine, as shown in this example:

   ```
   The FlexNet Licensing host ID of this machine is "808386e6"
   ```

3. Copy `lmadmin/demo/demo.lic` to the `<platform_dir>` directory.
4. Open `<platform_dir>/demo.lic` in a text editor.
5. Locate the `FEATURE` line for feature **f4**, and replace the current `HOSTID` value with the value of your machine's hostid. The feature definition line now looks something like:

   ```
   FEATURE f4 demo 1.111 permanent uncounted
           HOSTID=808386e6 SIGN="00DA...FBE5"
   ```

6. Save and close `<platform_dir>/demo.lic`.
7. At the command prompt from the `<platform_dir>` directory, enter the following to re-sign the license file:

   ```
   lmcrypt demo.lic
   ```

8. Close the command window.

Testing the License

Run `lmflex` and try to check out feature **f4**. A license for feature **f4** is successfully granted because the application is executing on the machine to which the license is locked. Check in the feature **f4.**

Because the license file for **f4** is in the same directory as `lmflex` and has an extension of `.lic`, its license is found and used. Additionally, the license for **f4** is uncounted and therefore does not require a license server to allow it to run.

Security Considerations

As previously explained, the `HOSTID` value is incorporated in the license signature and used to authenticate the license. Any changes to the license signature, the license `HOSTID=`*`value`*, or the actual machine hostid invalidates the license. As a result, the FlexEnabled application fails authenticate the license, and any license requests fail. The following exercise demonstrates this concept.

Task: ***To demonstrate security considerations***

1. Reopen `<platform_dir>/demo.lic` in a text editor.
2. Locate the `FEATURE` line for feature **f4**, and change the value of `HOSTID` to a made-up value, simulating an unauthorized attempt to rehost the license. The feature definition line now looks similar to this:

   ```
   FEATURE f4 demo 1.111 permanent uncounted
      HOSTID=000111111111 SIGN="00E3...EA16"
   ```

   Important • *Do not re-sign the license.*

3. Run `lmflex` and try to check out feature **f4**. A FlexNet Licensing error dialog box appears to indicate that the license request is denied because the application detects the inconsistency between the `HOSTID=`*`value`* in the feature definition line and that of the actual machine used to create the signature.
4. Click **OK** to dismiss the dialog box.
5. Delete `demo.lic` from the `<platform_dir>` directory.

   Important • *Make sure you delete the correct license file. Do not delete* `demo.lic` *from the* `lmadmin/demo` *directory.*

This last sequence demonstrates FlexNet Licensing's policy in the license technology. You, the software publisher, define the policy—in this case, locked to a specific machine—and create the license. You are in control of the license usage until you change it.

Testing an Expiring License

This section demonstrates how to set an license-expiration policy. To implement the policy, simply set an expiration date in the FEATURE line; you do not need to modify the licensing toolkit or the licensed application. When the license is signed, the expiration date is incorporated into the license signature. In the following steps, you modify the expiration date of the node-locked feature **f4** located in the `expired.lic` license file.

The expiration date is incorporated in the license signature and used for license authentication. Therefore, attempting to extend a license by simply moving the expiration date forward does not create a legitimate license. When a user moves the date forward, the FlexEnabled application fails to authenticate the license, and any license requests for the feature fail.

Required Components

This exercise uses the following FlexNet Publisher toolkit components:

- Sample test program `lmflex`
- Sample license file `expired.lic`
- The license-signing utility `lmcrypt`

The exercise also requires a general text editor.

Testing the Expiring-License Policy

Task: ***To demonstrate an expiring-license policy***

1. Make sure that you have deleted the `demo.lic` file from the <`platform_dir`> directory.
2. In the command window in which you run `lmflex`, enter `lmflex` at a prompt from the <`platform_dir`> directory.
3. In the <`platform_dir`> directory, copy `expired.lic` to a file called `expired_lic.orig` to preserve the original license.
4. Open <`platform_dir`>`/expired.lic` in a text editor.
5. Locate the FEATURE line for feature **f4**, and change the expiration date to a date in the future, as shown in the following example:

```
FEATURE f4 demo 1.000 31-jan-2021 uncounted
    HOSTID=ANY SIGN="0051...B55B"
```

6. Save and close `expired.lic`.

   Important • *Do not re-sign the license.*

7. Run `lmflex` and try to check out feature **f4**. Because the expiration date is different than the date encoded in the license signature, a FlexNet Licensing error dialog appears, stating that the license request is denied because the signature cannot be authenticated.
8. Click **OK** to closed the error dialog.
9. Open a new command window, and change to the `<platform_dir>` directory.
10. Re-sign the license file to incorporate the new expiration date into the license signature. To do so, enter the following at the command prompt:

    ```
    lmcrypt expired.lic
    ```

11. Run `lmflex` and try to check out feature **f4**. A license for feature **f4** is successfully granted because the license has not expired.

    Because the license file for **f4** is in the same directory as `lmflex` and has an extension of `.lic`, its license is found and used. The license for **f4** is uncounted and therefore does not need a license server to allow it to run.

12. Press **Enter** to exit `lmflex` and to check in the license for feature **f4**.
13. Close all command windows.

Building the FlexNet Publisher Toolkit

In this section, you rebuild the FlexNet Publisher toolkit to change the strength of tamper-resistant licenses (TRL) authentication from 113 bits to 163 bits. TRL uses public-key technology from Certicom to make the signatures on `FEATURE` and `INCREMENT` lines more difficult to counterfeit. As shipped, the toolkit has TRL enabled at the 113-bit strength, denoting a public-key length of 113 bits.

These exercises use the following FlexNet Publisher toolkit components:

- Sample test program `lmflex`
- Sample license file `counted.lic`
- The build configuration file `lm_code.h`
- The license-signing utility `lmcrypt`

The exercises also require a general text editor.

Important • *A C development environment is required to rebuild the toolkit for this part of the evaluation. Consult the FlexNet Publisher Release Notes for your platform's C development environment requirements.*

Examining the Current License Signature Length

Task: ***To examine the current license signature length***

1. Open `<platform_dir>/counted.lic` in a text editor.
2. Observe the length of the signature (`SIGN`) value in the `FEATURE` line for **f1**. Based on a 113-bit strength, the value should have a length similar to the following:

   ```
   FEATURE f1 demo 1.0 permanent 4 SIGN="003D B054 54D1 47E3 1FF6 \
       145F 59CE 0200 8002 FE69 0FBC F4FF AA10 EEB3 6EB3"
   ```

3. Optionally, save a copy of this file for future comparison.
4. Close `counted.lic`.

Configuring the FlexNet Publisher Toolkit

For previous exercises, you used the prebuilt, “out of the box” FlexNet Publisher components. However, as shipped, the FlexNet Publisher toolkit is in an unconfigured state. You must configure it in order to rebuild it. Use the `demo` vendor keys provided to you by Flexera Software for the following configuration procedure.

Task: ***To configure the FlexNet Publisher Toolkit***

1. Open the `<install_dir>/machind` folder.
2. Locate the file `lm_code.h`, and open it in a text editor. Find the lines similar to the following:

   ```
   #define VENDOR_KEY1 0x0
   #define VENDOR_KEY2 0x0
   #define VENDOR_KEY3 0x0
   #define VENDOR_KEY4 0x0
   #define VENDOR_KEY5 0x0
   ```

3. Replace all five instances of `0x0` with the demo vendor keys that you received from Flexera Software. If your demo vendor keys have expired, contact Flexera Software.
4. Keep `lm_code.h` open.

Changing TRL Strength

Task: ***To change the TRL strength***

1. In `lm_code.h`, find the `LM_STRENGTH` definition that looks similar to the following:

   ```
   #define LM_STRENGTH LM_STRENGTH_113BIT
   ```

2. Redefine `LM_STRENGTH` to 163-bit strength, as shown:

   ```
   #define LM_STRENGTH LM_STRENGTH_163BIT
   ```

3. Leave `TRL_KEYs` set to `0x0`. They are not set when using demo vendor keys.
4. Save and close `lm_code.h`.

Building the FlexNet Publisher Toolkit

For your changes to take effect, you need to build the FlexNet Publisher toolkit.

Important • *Make sure `lmflex` is closed and the license server is stopped before you attempt to rebuild the FlexNet Publisher toolkit.*

Task: ***To build your Licensing toolkit***

In a command window, change to the `<platform_dir>` folder, and enter the following:

```
make
```

The demo toolkit now supports TRL at the 163-bit strength level. Because this modification changes the authentication method, all components—including the sample application, `lmflex`—have been rebuilt.

Re-signing the Sample License File

Regenerate the license file to create license signatures that support TRL authentication at the 163-bit level.

Task: ***To regenerate the license file***

1. From the `<platform_dir>` directory, enter the following:

   ```
   lmcrypt counted.lic
   ```

2. Open `counted.lic` in a text editor.
3. Compare the length of the `SIGN` value in the **f1** `FEATURE` line with the length of the `SIGN` value that you examined before the toolkit rebuild. The new signatures are longer, similar to the following:

   ```
   FEATURE f1 demo 1.0 permanent 4 SIGN="01C6 1045 146D AE6C 92CB 0360 \
       1124 AC73 B79B 1785 C000 9608 773C EAAF 3815 55DF 0FAF 786A \
       AA19 E123 52A3"
   ```

4. Close `counted.lic`.

Conclusion

You have completed the evaluation of FlexNet Publisher on Windows. When you are ready to build your Licensing toolkit with your production vendor keys, see the manual *Development Environment Guide*.

To evaluate FlexNet Publisher on Windows, see Chapter 4, "Evaluating FlexNet Publisher on Windows."

Important • *Remember to set* `LM_STRENGTH` *to an appropriate value before you build your production licensing toolkit.*

6 License Models

A FlexNet license model employs a basic license model augmented with one or more license model modifiers. In addition, you frequently license the same product under several license models to best match license terms to the needs of your different types of users. A well-designed license model or set of license models enables you to expand your available market, enhance end-user loyalty, and increase account penetration, resulting in increased revenues and profitability.

With FlexNet Licensing's *Policy in the License* technology, you implement the license model in the license certificate that is delivered to a specific end-user organization. This enables you to make changes to license terms without requiring changes to your product's source code or binaries.

Creating a License Model

This chapter walks you through the following steps to create your license model:

1. Select the basic license model.
2. Decide on the settings for the required license modifiers.
3. Select optional license model modifiers for your license model. Note the license server requirement for the optional modifiers.
4. Determine your need for usage-based licensing.
5. Determine your support level for mobile licensing.
6. Determine whether you require a license server.

Basic License Models

Table 6-1 presents the basic FlexNet license model characteristics. You choose one or more of these and build on them to create a model or combination of models that represent your business strategy.

Table 6-1 • Basic License Model Characteristics

License Types	Uncounted	Counted
Node-locked	*Applicability:* Common. *Usage:* Allows for unlimited instances of the product to execute on the licensed machine. *Terminology:* Sometimes referred to as node-locked/uncounted or just node-locked.	*Applicability:* Uncommon. *Usage:* Allows for limited instances of the product to execute on the licensed machine. *Terminology:* Sometimes referred to as node-locked/counted.
Floating	Not defined.	*Applicability:* Common. *Usage:* Each instance of the product consumes a fixed count—usually one—of licenses. *Terminology:* Floating, network, non-node-locked, concurrent.

Node-Locked Uncounted Models

With this model, your product is licensed for unlimited concurrent use on a single computer system. The license file is available locally to the same machine on which the product is installed. This model can be used to effect different product configuration schemes:

- **Single feature**—A single feature license that enables the entire product.
- **Scaled functionality**—A single product has several levels of functionality built in; access to each level is controlled by a separately FlexEnabled feature. This also enables a single product to be licensed as a limited functionality **lite** version, a **standard** version and a **pro** version to give greater product depth.

Evaluation License Model

The node-locked uncounted model is one way to provide evaluation licensing for your product. It enables your product to be licensed for unlimited use on all systems. The license file is located on the same machine on which the product is installed. Characteristics of an evaluation license may include:

- Limited product functionality
- Limited number of uses

- Expiration date

Node-Locked Counted Models

A fixed number of licenses is available concurrently over a network, but the licenses may only be used on a specific list of hosts on that network. You get the specific host names from the end user and put them in the license certificate. These host names contribute to authenticating the license signature. The product may then be used on any computer defined in the license certificate. This model requires a license server.

Floating Models

This model represents classic networked, concurrent licensing and is sometimes referred to as non-node-locked/counted, network, or concurrent licensing. It requires a license server.

Entitlement is issued for a specific number of licenses. *Entitlement* refers to the rights you give your customer to use your FlexEnabled product. For license file–based licensing, the license certificates you give your customer define their entitlement. For a floating model, no more than the number of entitled licenses can be used at any one time on a network. However, there are features available to overcome denial of service if all licenses are in use. See Checkout Failure Behavior for further details.

License Model Modifiers

This section describes modifiers for the FlexNet Licensing basic license models. They augment the license models that are described in Basic License Models.

Required License Model Modifiers

Every license must have an expiration date and a version value.

Expiration Date

The license expiration date is specified in the license certificate either with a specific date or with the keyword, `permanent`.

- Specify a date for leased, subscription, or time-limited models.
- Specify `permanent` for non-expiring licenses.

Version

A version number defines right-to-use entitlement with versions less than or equal to the version specified in the license certificate. This is FlexNet Licensing's notion of the version and not necessarily the same as the product version. You assign your own meaning to the version number. Some examples include:

- **Product version**—The license version coincides with your product's version. Entitlement is based on the product's version being less than or equal to the version specified in the license certificate.

  The advantage of this model is that you maintain only a single version number. However, the disadvantage is that with each product release with a change in the version number, you may need to reissue the license certificate to your customers.

- **License version**—The license version is independent from the product's version. The notion of the license version is encoded inside the product binary. Entitlement is based on the product's internal license version being less than or equal to the version value specified in the license certificate. For example, your product release version might be "3.1.2", and the license version could be "3.0". The different numbers allow your customers to run a later version of the product using an older license certificate.

  This is the most commonly used model because it provides the flexibility to decide whether the license certificates containing older version numbers still work with the new release.

- **Date-based version**—The license version is based on a date. The version of the product binary is encoded using a date-based scheme. Entitlement is based on the product's internal date-based version being less than or equal to the version value specified in the license certificate.

  Note • *Date-based versioning pertains to the product version and specifies which version of the product is entitled to run. This modifier is distinct from license start and expiration date modifiers, which specify a date range within which the entitled version is licensed and allowed to operate.*

Optional License Model Modifiers

One or more of these modifiers can be used in combination with your basic model and the required modifiers to customize it for your particular business model.

These modifiers are represented in the FlexNet Licensing implementation by fields in the license certificate, FlexNet Licensing attributes set in the product, vendor daemon configuration, or end-user options. Table 6-2 lists the modifiers along with a brief description of each.

Table 6-2 • Optional License Model Modifiers

Modifier	Description	License Server Required
Capacity	Dynamically adjust entitlement based on system performance parameters.	✓

Table 6-2 • Optional License Model Modifiers

Modifier	Description	License Server Required
Duplicate grouping	Defines rules when duplicate requests from the same user, host, or display share one license. Requires a counted license. Sometimes referred to as license sharing.	✓
Linger	The license is held, rather than checked in, by the license server on behalf of the user after the product exits. For a product that is run over and over again in succession by the same user (for example, a compiler), lingering increases the chances that a license is available for that user.	✓
Named hosts	Limits usage to a publisher-defined number of hosts. The list of distinct host names is defined by the end-user license administrator, in the administrative options file.	✓
Named users	Limits usage to a publisher-defined number of users. The list of distinct user names is defined by the end-user license administrator, in the administrative options file.	✓
Network segments	Limits usage to hosts on a specific subnet.	
Overdraft	Specifies a number of additional licenses that your end user will be allowed to use, in addition to the licenses they have purchased. This is useful if you want to allow your customers to not be denied service when in a “temporary overdraft” state.	
Package	Defines a set of products or features to be licensed as a single package.	✓
Package suite	Provides a way for users to share floating licenses among components of a package.	✓
Platforms	Enables you to restrict usage to specified hardware platforms.	
Start date	By default, a license is valid as soon as it is signed. However, if a start date is specified, the license is invalid until the start date.	
Supersede	Replaces previously issued license rights.	✓
Terminal server support	Allows users to run the FlexEnabled application on Terminal Server Client machines or over Remote Desktop.	
Timeout	Causes the license to be returned to the general pool of available licenses if the FlexEnabled application is not active for a specified number of seconds.	✓

Table 6-2 • Optional License Model Modifiers

Modifier	Description	License Server Required
Vendor-defined	Used for publisher-defined entitlement such as token based.	

Usage-Based Licensing

Usage-based licensing is an implementation strategy for license management and is another modifier that can be applied to your basic license model. It provides you and your end user with the ability to monitor actual product usage patterns and bill or audit based on this usage data. Contact Flexera Software if you want to implement this license model. You may need to obtain additional Flexera Software products.

The report logging functionality of FlexNet Publisher is fundamental to this model, and, as such, requires a license server. FlexNet Publisher supports several usage-based models, which include:

- Overdraft
- Pay-per-use
- High water mark of past use

Mobile Licensing

End users often want to use a FlexEnabled product on a machine that does not have a continuous connection to a FlexNet license server. These situations include:

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

FlexNet Publisher supports several kinds of mobile licensing, as shown in the following table.

Table 6-3 • Supported Mobile Licensing

Licensing Scenario	Description
Node-locked to a laptop	If a license is to be used exclusively on one laptop computer, that license can simply be node-locked to a hostid of that computer. The license file would reside on the laptop computer.

Table 6-3 • Supported Mobile Licensing

Licensing Scenario	Description
Node-locked to a FlexNet ID dongle	If a license is to be moved between different machines, it can be node-locked to a FlexNet ID dongle that connects to a parallel or USB port. This license can be moved between several machines by installing a copy of the license file on each of those machines and moving the dongle from one machine to another. This applies to Windows, Mac OS, and Linux platforms only.
Node-locked to a user name	If a license is to be used exclusively by one user on different machines, that license can be node-locked to the user's user name. The license file is copied to the different machines on which the user might work. The user's user name must be identical on each machine.
Fulfilled from a prepaid license pool	The license is fulfilled from a prepaid number of license-days for the usage period. This model is like pay-per-use because each fulfillment is made from a decreasing number of license-days. It is different than other pay-per-use models because, once node-locked to a machine, that machine is allowed unlimited use of the product until the license expires.
Soft-mobile	Licenses are temporarily transferred to a license server on the mobile laptop. The FlexEnabled product uses an encrypted local file, placed there by the license server, to do checkouts during the usage period.
Hard-mobile	Mobile license usage is controlled by a FlexNet ID dongle. If the dongle is attached to a license server, then the use floats on the network. To temporarily transfer the license, the user moves the dongle from the license server to a standalone machine. This removes a license from the served pool until the dongle is replaced on the license server.
License rehosting	An end user may request license rehosting when they want to move a license without using one of the other mobile licensing methods. To enable license rehosting, you need to generate a new node-locked license certificate for each new machine. The end user is trusted to destroy the previous licenses so that only one is in force at any given time.

Determining the Need for a License Server

You need to decide if your licensing policy requires a license server. The need for a license server stems from a FlexNet Publisher requirement, your requirement, your end user's requirement, or a combination of requirements.

The term license server refers to a set of processes, not the machine on which they run. Licenses controlled by a license model requiring a license server are referred to as served licenses and those not requiring a license server, unserved licenses. A license server keeps track of the number of licenses being used for counted models, as well as a number of other parameters for both counted and uncounted models.

If your licensing policy or deployment requirements have any of the following characteristics, you must include a license server in your implementation.

- License model that requires a license server as noted in Node-Locked Counted Models, Floating Models, and Optional License Model Modifiers
- Requirement by you or your end user to capture debug output, which is useful for the end-user license administrator when setting up the system
- Requirement by you or your end user to capture license usage activity for analysis by Flexera Software products (for example, FlexNet Manager).
- Requirement by your end user to centralize licenses for availability over a network
- Requirement by the end user to enable end-user options.

7

Publisher Decisions

This chapter presents licensing policy concepts that you need to consider when designing a licensing policy for your product.

Summary of Publisher Decisions

Licensing policy decisions fall into two groups—core and license server. The core decisions (described in the following section, Core Decisions) define the fundamental behavior of your licensing policy and apply to both served and unserved license models. License server decisions, which are described in the section, License Server Decisions, apply only to served license models. You need to consider them only if you are implementing a license model that requires a license server.

Consider these decisions after you have decided on the license model. Chapter 6, "License Models," defines license models and guides you through the model-decision process.

Core Decisions

The decisions presented in this section need to be considered regardless of the license model.

- Determining Licensable Product Configurations
- Hostid Support
- Checkout Failure Behavior
- Check-in Considerations
- Optional Diagnostic Handling

Determining Licensable Product Configurations

You can license the entire product or just specific features. By designing a creative licensing policy, the same product binary can have several configuration levels such as lite, standard, and pro configurations of the product. Defining various product configurations, or features, is done at the source-code level within the FlexEnabled product using calls to the FlexNet Publisher client library. FlexNet Licensing does not otherwise understand features within an executable.

Additionally, individual products can be grouped together and licensed as a package, or as a package suite. Packages and package suites are defined in the license certificate rather than in the product. This enables you to change the components of a package until just prior to shipping the license or at any time in the future.

Hostid Support

A hostid is a means used to uniquely identify a specific system. There are two different contexts in which a hostid is used:

- **FlexNet license**—The license is bound (locked) to a hostid and is used to define which end-user hosts are licensed to run your product.
- **FlexNet license server**—One or more hostids are used to define which machines are authorized to run a license server that serves licenses to hosts on which the product will run.

Each platform supports one or more methods of determining its hostid. You need to know which system architectures are supported by your FlexEnabled product and decide which hostid scheme you want to support.

Table 7-1 provides an overview of the trade-offs associated with different hostid approaches.

Table 7-1 • Hostid Approaches

Approach	Pro	Con
Ethernet (MAC) address	Reasonably secure	Not available on all platforms
Disk volume serial number	Always present	Insecure
Dongle	• Secure • Mobile	Customers dislike and carries an additional cost
Vendor-defined	Ultimate in flexibility	Programming required
IP address	Allows end user easy control	Insecure
Proprietary notice	• Minimum burden to implement • Can be audited • Pirating risk	Insecure, especially from those who are never audited

Table 7-1 • Hostid Approaches

Approach	Pro	Con
Serial number	No vendor involvement	Only mild deterrent, but may be appropriate in certain circumstances

In addition to system-specific hostids, several special FlexNet Publisher hostids are supported. You need to decide which, if any, of these special hostids you will support.

Table 7-2 • FlexNet Publisher Special Hostids

Hostid	Description
ANY	Does not bind the license or license server to any system. ● In the context of a license, allows all systems use of the license. ● In the context of a license server, allows all systems to run a license server.
COMPOSITE	Locks the software to a hashed12-character hexadecimal value formed by combining the values of one or more simple hostids types, as defined by you. ***Note •*** *We do not support Dongles in composite_hostid.*
DEMO	Similar to ANY, but for use with unserved license models only. It cannot be used in the context of a license server.
DISPLAY	Binds the license to a specific display. ● On UNIX this is `/dev/ttyxx` 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.
HOSTNAME	Binds the license to a system with a specific host name. Allows only users on that host access to the FlexEnabled product.
USER	Binds the license to a specific user name. Allows only that user access to the FlexEnabled product.

See Hostids in the *Programming Reference for License File-Based Licensing*: This section provides a complete list of supported hostids.

Checkout Failure Behavior

The fundamental operation which allows a given feature to execute within the FlexEnabled product is to check out a license for the desired feature. This operation either succeeds or fails.

The behavior for a successful checkout operation should be well defined within your policy. However, your policy also needs to define the FlexEnabled product behavior in the event of a failed checkout. FlexNet Publisher provides several ways to handle a failed checkout.

Table 7-3 • Methods Used After Checkout Failure

Method	Description
Accept overdraft license usage	Even though no additional licenses exist, allow the checkout as an overdraft to proceed and decide how you want to handle license overusage. Setting up a mechanism to charge the end user for the overdraft is one way to handle the overusage.
Allow the product to run in a limited mode	Cripple the product in some manner that provides very basic features or is time limited.
Wait for a license to become available	In this case, the product will pend, waiting for a license to be checked in to the license pool.
Deny service: do not allow the product to run at all	The FlexEnabled product exits and control is returned back to the end user.
Allow the product to run	In this scenario, the FlexEnabled product operates as if the checkout succeeded.

FlexNet Publisher also provides a way to determine the reason for checkout failure. This enables you to handle each checkout failure differently, depending on its reason.

Check-in Considerations

When a FlexEnabled product exits, all licenses are, by default, automatically checked back in to the license pool. Licenses can be checked in asynchronously from the product exiting. You can control the point at which a license is checked back in to the license pool.

Additional check-in considerations as they relate specifically to served licenses are discussed in Check-in Considerations for Served Licenses.

Optional Diagnostic Handling

FlexNet Publisher provides an optional mechanism for displaying diagnostic messages to the end user. The messages come in two formats: long and short. When you implement FlexNet Publisher licensing in your product, you configure the diagnostic format. For Windows products, diagnostics can optionally be displayed in an error dialog box.

Both formats of error messages are extracted from the FlexNet Publisher client libraries into text files to provide for localization.

License Server Decisions

Read this section if your license model requires a license server (see "Determining the Need for a License Server," in Chapter 6, "License Models.") The following decisions apply to a served license model:

- Maintaining Consistency with the License Server
- License Sharing
- Check-in Considerations for Served Licenses
- License Server Redundancy
- Support for Server Virtualization

Maintaining Consistency with the License Server

A license server comprises a license server manager and a vendor daemon. The license server manager handles the initial contact with the FlexEnabled product, passing the connection on to the appropriate vendor daemon. It also starts and restarts the vendor daemon. The purpose of the license server manager is to:

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

There are decisions to be made which define the connection strategy between the vendor daemon and the FlexEnabled product. These decisions are implemented in a customized vendor daemon, the FlexEnabled product, or both. Each FlexEnabled product must establish at least one connection to a license server. This connection is managed by a job handle.

In a FlexNet license server environment, licenses are granted by vendor daemons. There is one vendor daemon for each software publisher who has a FlexEnabled product on the network.

Each vendor daemon keeps track of how many of its licenses are checked out, and who in the enterprise has those licenses. If a vendor daemon terminates for any reason, all users who are currently using licenses tracked by that vendor daemon lose their licenses (though this does not mean the product suddenly stops running). Users normally regain their licenses automatically when the license server manager restarts the vendor daemon, though the application may exit if the vendor daemon remains unavailable.

Connection/Reconnection Strategy

Upon daemon failure, the FlexEnabled product can:

- Retry the connection for a specified number of times.
- Retry forever to connect.
- Perform pre-reconnection processing.
- Perform post-reconnection processing.

- Perform post processing upon reconnection failure.

After a pre-set timeout interval, the vendor daemon reclaims inactive licenses.

Heartbeat Implementation Strategy

The FlexNet license server uses TCP/IP to communicate to the FlexEnabled product; it can detect when a client connection is gone and frees the FlexEnabled product's licenses automatically. However, a product needs to communicate regularly with the license server to detect that the server is still running. This communication is carried out using *heartbeats*, which are periodic messages initiated by the FlexEnabled application and acknowledged by the license server.

If the license server is shut down and restarted, the heartbeat automatically checks out the complement of licenses that is currently being used by the product. If your product does not send heartbeats to the license server, the product does not know that the license server has been shut down and restarted.

The FlexEnabled product does not detect that the license server is down until the time it takes two heartbeats to be exchanged has passed, because the heartbeat acknowledgment checked for by heartbeats is actually the acknowledgment of the previous heartbeat. If you shut down a license server, the next heartbeat succeeds, because the acknowledgment processed is of the previous heartbeat. If a heartbeat is not acknowledged by the license server, the product can decide what action to take—continue, warn, or terminate.

Deciding how the heartbeats occur and what action takes place when the license server is not running is an important part of incorporating FlexNet Licensing into a product.

Automatic Heartbeats

By default, regular or *automatic* heartbeats are automatically initiated when a license is checked out. Heartbeats are exchanged with the license server at a default interval of two minutes.

Automatic heartbeats are recommended and are implemented as a separate thread in the FlexEnabled product.

Manual Heartbeats

For various reasons, you may choose to implement manual heartbeats instead of automatic ones. Some considerations for implementing the manual heartbeats include:

- The product cannot be multithreaded.
- The product can be interrupted only at specific times. A product may have critical sections of code that cannot be interrupted. Manual heartbeats provide a way for the product to control when the heartbeat thread executes thereby not perturbing the critical sections.

See the *Programming Reference for License File-Based Licensing* for more details on heartbeats and their manual implementation.

License Sharing

License requests can be grouped so that multiple requests from the same entity use only one license. Subsequent feature checkouts are absorbed into the first checkout. This is referred to as *duplicate grouping*.

Duplicate requests are determined on the basis of user name, host name, display name, a combination of the three, or can be publisher defined. One way to implement site licensing is to specify `DUP_GROUP=SITE` for a feature.

Duplicate grouping requires a counted license model and a license server.

Check-in Considerations for Served Licenses

In addition to the Check-in Considerations, the following considerations apply to served licenses:

- Control whether the TCP/IP connection to the license server manager stays open after all licenses are checked in.
- Manage the check-ins from the vendor daemon by defining check-in callback and filtering routines.

License Server Redundancy

FlexNet Publisher supports two methods of redundancy:

- Using a license search path in the `LM_LICENSE_FILE` environment variable.
- Using a set of three redundant license servers.

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

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

Internal organization of license jobs within the FlexEnabled product affects the way the end user sets up license servers and the license files on those servers. For served licenses, all features checked out by a given license job need to be served by the same license server. This affects the usefulness of license search paths. You need to decide the organization of license jobs within the product.

Support for Server Virtualization

Server Virtualization technology, which allows datacenter hardware resources to be used more efficiently, is rapidly gaining popularity. Companies such as VMware, Microsoft, and Citrix provide Virtualization solutions. FlexNet Publisher supports the Virtualization of license servers.

Support for Cloud-Computing Models

FlexNet Publisher provides licensing solutions for the Amazon EC2 cloud-computing environment. These solutions, based on typical licensing use cases in a cloud environment, provide binding elements for both license clients and license servers. For more information, see the *Programming Reference for License File-Based Licensing*.

8

License Security Systems

There are several aspects of licensing security to consider. This chapter presents the issues and strategies. For implementation details, see the *Programming Reference for License File-Based Licensing*.

Overview

Licensing security means securely enforcing the licensing policy between you and your end user. End users can be grouped into two general categories: honest and dishonest. A useful security implementation ensures the honest end user does not, unknowingly, use extra licenses and the dishonest user can not easily disable your licensing.

FlexNet Publisher provides product licensing security in much the same way as a lock provides security to a door. A license key (to the lock) is needed to unlock the product (lock). The lock does not completely secure the door; it discourages thieves, but the door can be forced open.

Limiting Exposure

Generally speaking, additional security comes at the cost of inconvenience (or additional financial cost with dongles). FlexNet Publisher does not require that all end users lock their product the same way. You can decide to use low-security measures in certain countries or for selected important end users, while using more secure methods elsewhere.

FlexNet Publisher's *Policy in the License* technology enables you to use different security mechanisms for different market segments, while not requiring the development and support of multiple versions of your product. Thus, you control your exposure to security risks as appropriate.

From a practical point of view, most people with the intent of pirating your product will never pay you any money for it. Guarding against unintentional overuse and providing moderate licensing barriers have the most impact against software pirating. Trying to make the product secure from the dedicated cracker (who will never be a paying customer) can discourage the honest, but severely inconvenienced, end user from buying your product.

Server-Side Exposure

Introduce safeguards in your licensing policy to protect the license server from being hacked or replaced with an impostor. In the request/grant process, the FlexEnabled product depends on certain responses from the license server. A hacked or imposter license server could falsely provide favorable responses that grants unlimited licenses to your product.

Product-Side Exposure

- Implement safeguards to detect imposter license servers.
- Prevent strings used for feature names and other components of your license certificate from appearing as clear text within the product binary.
- Monitor the license server status using heartbeats. Ensure that the license server is not being stopped and restarted in an attempt to get extra licenses.
- Prominently display end-user identifying information when product starts up.

License Signature Encryption Method

FlexNet Publisher offers two signature encryption methods:

- Public/private key digital signature (known as Tamper-Resistant Licenses or TRL)
    - Industry-recognized asymmetric, public/private key digital signature
    - ECDSA algorithm
    - Three different signature lengths
    - Mathematically based public key
- Proprietary digital signature
    - A proprietary, symmetric, digital signature method
    - 48-bit long signatures

See the *Programming Reference for License File-Based Licensing* for implementation details on either method.

Expiring Licenses

You can issue a license certificate which either expires on a certain date or is permanent. Some scenarios where an expiring license certificate is prudent include:

- Demo licenses that are not tied to a specific host (`HOSTID=DEMO` or `HOSTID=ANY`). Because demo licenses do not lock your product to a specific host, specifying an expiration date is a way for you to control the entitlement.
- A license certificate used in potentially high-security-risk environments where you want tighter control on entitlement.
- Opportunities to manage revenue. Issue an expiring license certificate so that revenue can be recognized each time the certificate is renewed.

Expiring license maintenance and overhead is higher than with permanent licenses. The security risk for expiring licenses is low; however, security can be compromised by system date tampering.

Date Tampering

FlexNet Publisher performs security checks to prevent users from setting system dates back. In addition, Table 8-1 describes measures you can take to make date setback less attractive:

Table 8-1 • Security Measures to Protect Against Date Tampering

Security Measure	Description
Prominently display expiration date	Display the expiration date of the license in a prominent place so that the date-setback detection is more public.
Provide an insistent reminder	If it is an expiring evaluation version, periodically do something annoying—perhaps a popup that appears every few minutes that encourages the user to purchase the product.
Disable some functionality	An example is a word-processing program that alters saved files so that, when printed, the word “EVALUATION” is printed in large letters across every page. This allows evaluators full functionality, without reasonable utility.

HOSTID Choices

Using special hostid types, `ANY` and `DEMO`, in your licenses creates potential security risks. These hostid types specify the universal host, and as such, do not lock your product to a specific system, thus, allowing your product to run anywhere.

For a detailed description of how to choose a hostid type see Chapter 7, “Hostid Support,” which discusses the decisions you need to make with regard to choosing a hostid type. Table 7-1 compares different hostid types and their relative levels of security.

The advent of virtualization technology poses special security challenges. A license server that has been bound to a virtual hostid could easily be transferred to other physical hosts by simply copying the virtual machine in which it is running. FlexNet Publisher offers protection against this kind of license leakage, by allowing you to enforce virtualization policy regarding license servers.

Additionally, FlexNet Publisher now offers several ways to bind license clients and servers in various cloud-computing models. For more information about licensing in virtualized and cloud-computing environments, consult the *License Administration Guide*.

Building the Product: Dynamic vs. Static Linking

Use a statically linked model whenever possible for linking the FlexNet Publisher client library into your product. Dynamically linking the Licensing toolkit client library is insecure for the following reasons:

- It leaves the product vulnerable to being spoofed. The Licensing toolkit DLL could be replaced with an imposter.
- A "hacker" product can listen to messages transmitted between the product and the DLL, thereby gaining intelligence about the licensing policy.

To minimize the risk of a dynamically linked model being hacked, make your own DLL that implements Licensing toolkit client wrapper functions. Your product then calls your wrapper functions instead of the Licensing toolkit client functions. This technique provides another layer of security; the hacker would have to compromise your proprietary layer.

See *Development Environment Guide* for more information on dynamically linking your product.

Tracking End-User Usage

A FlexNet license server can produce report logs, which are files containing feature usage information. Report log file data can be used for billing, auditing, and market intelligence. Usage data in report log files is encrypted and, as such, provides a relatively secure method of tracking end-user usage.

Contact Flexera Software for information about products that process the report logs.

Usage Data Limitations

The report log file, while ASCII (so it can be easily emailed), is not human readable. Vulnerabilities that must be considered include:

- The end user may simply lose a report log file (either by accident or on purpose).
- Periods of usage data can be lost without detecting a file modification, although the fact that a time period is missing can be detected.

Consider implementing a policy for missing reporting periods. One example policy is: "More than *x* hours per month of missing license usage entries terminates the licensing contract." A similar policy is needed for files that have been altered.

9

License Fulfillment

In addition to your licensing policy, you need to decide on methods for license fulfillment and installation at the end-user site.

License Fulfillment

License fulfillment is the process of distributing a license certificate to an end user. There are several ways to accomplish this:

- Over the Internet, also known as automated fulfillment
- Using email to deliver the license certificate
- Other methods such as delivering the license certificate as a hard-copy document or verbally relaying the certificate information

The following sections describe these methods. You need to develop a license fulfillment strategy that is compatible with your business model.

Back-office management products, such as Flexera Software's FlexNet Operations, provide you with the capability to support a wide range of functionality within your licensing policy and to manage all the information pertaining to customers and their licenses. FlexNet Operations facilitates the distribution of the license certificate by various means.

Automated Fulfillment

One of the most effective ways for you to distribute and for end users to receive a license certificate is over the Internet. Your end users can gain access to licenses 24 hours a day, seven days a week. Some aspects of automated fulfillment include:

- Fulfill orders by dynamically building electronic licenses as end-user orders are received. These orders may come through online ordering systems, through manual order entry, or with registration cards.

- Translate product part numbers into electronic licenses based upon product definitions and descriptions of licenses.
- Manage the phased delivery of licenses so volume orders can be distributed and tracked as the end user takes delivery of licenses.
- Manage an accurate database of the licenses that have been distributed to specific customers.

Email

The license certificate can be delivered via an email message, either as an attachment or in the body of the message. FlexNet Publisher automatically ignores email headers and extraneous text when processing the certificate, making it convenient to save the entire email message without having to strip out just the license certificate.

Other Methods

It may be necessary to deliver the certificate by other methods, either because electronic means are not available to your end user or as a backup to your primary electronic means. Other methods include, but are not limited to:

- Mailing or faxing a hard copy of the license certificate
- Verbally relaying the license certificate information

Strategy for Delivering Rights Over Time

The initial license certificate distributed to the end user can subsequently be augmented in order to keep pace with changing requirements and usage patterns. With FlexNet Publisher's *Policy in the License* technology, there is usually no need to reissue the product binary.

For in-depth information on the following types of certificates, see the *Programming Reference for License File-Based Licensing*.

Upgrading

The license certificate provides entitlement for any license version of the product, up to and including the version specified in the certificate. A new UPGRADE certificate can be issued that enables entitlement to a later version of your product. Then, your end user is all set for migrating to the newer version of the product at any convenient time.

Incrementing

The number of licenses specified in an existing license certificate can be increased. A new `INCREMENT` certificate can be issued that supplies the end user with additional licenses.

Superseding

You can discontinue use of an older product by distributing a new `SUPERSEDE` certificate for the new product. A `SUPERSEDE` license certificate disables the old product licenses and, at the same time, provides licenses for the new product.

Support Entitlement

Your electronic license certificate, issued at the time of the product transaction, can include a vendor-defined support-through date. You can enable your customers to automatically download updates and upgrade their product through the initial support-through date. When they renew support, issue a new electronic license certificate, with a one-year (or 30-day or whatever your policy may be) extension to the support-through date.

Transferring Rights - License Rehosting

It is common for an end user to want a valid license certificate reissued in order to lock the product, license server, or both (as applicable to the license model) to a different machine. This process is called license rehosting.

There is no mechanism in FlexNet Publisher to prevent the original license from working after the license has been rehosted. In the license agreement with your end user, it is recommended that you specify constraints related to license rehosting. Consider the following:

- Issue annually expiring licenses to help minimize the impact of additional use from undiscarded license files.
- Limit the number of free hostid changes you make for an end user.
- Charge a fee for rehosting.
- Require a hardcopy from the end user’s site of the rehosting request approved by a corporate financial officer. This way, the request gains visibility within the end-user organization and may dissuade “convenience” rehosting requests.

Evaluation Licensing Strategy

There are many popular methods of handling evaluation licensing; this section discusses the most common. All evaluation-licensing models employ `HOSTID=DEMO` in the license certificate, enabling the product to run on any host without the need of a license server. FlexNet Licensing's evaluation license models allow you to conveniently distribute your product for evaluation and, at the same time, control entitlement, because:

- No license server is required.
- License installation is easy.
- License files are easy to distribute, since no end-user information is required.

Limited-Time Demos

With this model, a fully functional version of your product is available, but for a limited time. The policy is contained completely within the license certificate.

Advantage

No special coding is required in the product.

Disadvantage

Vulnerable to system date tampering, which can give unauthorized usage beyond the preset demo period.

Limited Functionality Demos

Provide your product with limited functionality or with full functionality that is crippled in some way. Provide a license certificate with `HOSTID=DEMO`. Then, instrument your product to execute with the limited functionality when it detects this hostid.

Advantage

Risk is low for unauthorized usage, provided the limited function implementation is obfuscated against hacking.

Disadvantage

Special coding is required in the product.

10

Licensing Case Study

This chapter presents a case study based on products and business strategies from a fictitious corporation, Widget Design Company. It assumes a basic familiarity with the material presented in previous chapters.

Background

Widget Design Company has discovered a growing problem with uncontrolled use of their flagship product, Widget Design Engine. There is "honest" overusage, as well as unauthorized usage. Additionally, each product configuration is currently delivered as a separate binary on its own CD.

Widget Design Company recognizes that they are losing potential revenue and may be spending more than is necessary on CD production. To maximize their income, Widget Design Company decides to:

- Use software licensing to better control and track product usage.
- Cut production and inventory costs by delivering all product configurations as a single binary on one CD.

Defining the License Policy

This software publisher decides to incorporate FlexNet Publisher license management into their software. FlexNet Publisher provides them with the usage control they need and enables them to provide finer granularity of product configurations using a single binary. An additional saving is a reduction in CD manufacturing costs.

After completing a FlexNet Publisher evaluation, Widget Design defines the following licensing policy objectives:

- Feature-based licensing
- Delivery of a single binary for all implementations
- Evaluation, node-locked, and floating license models as appropriate
- Time-limited licenses

- Free distribution of an evaluation version, with limited functionality and time-limited use

Until the software publisher has gained experience in using software licensing, they will not support the following:

- Mobile licensing or lingering
- Redundant servers
- Packages or package suites

Product Configurations

The Widget Design Engine product consists of a design engine that works with a proprietary data format, an editor, an advanced editor, and features to import and export data between proprietary and standard formats. They plan to offer three product configurations—an evaluation version, a standard version, and an advanced version. Table 10-1 summarizes the product features and their mappings to the different product configurations.

Table 10-1 • Widget Design Engine Feature Matrix

	Product Configurations		
Product Feature	**Evaluation**	**Standard**	**Advanced**
Widget Design Engine	✓	✓	✓
Editor	✓	✓	
Advanced Editor			✓
Import Capability	✓	✓	✓
Export Capability		✓	✓

Licensing Configurations

Because Widget Design Company plans to distribute the same binary for all product configurations, it must determine how to configure the product versions using FlexNet Publisher. There are three licensing configurations for this product; these correspond to the three product configurations.

Table 10-2 • Licensing Configurations

Licensing Configuration	Description
Evaluation	Available for any machine for a one-month time period. The evaluation license configuration provides a limited set of features for the end user to evaluate the design capabilities of the product. The evaluation license provides for no export capability, which is an important piece of product functionality. The limited functionality, combined with the time-limited evaluation period, is intended to encourage evaluation customers to purchase fully functional standard or advanced versions of the product.
Standard	Available for node-locked or floating usage. This licenses a fully functional product that includes the standard editor and allows both import and export.
Advanced	Available for node-locked or floating usage. This licenses a fully functional product that includes the advanced editor and allows both import and export.

License Policy Definition

Based on Widget Design’s business model and product structure, their policy design committee formulates the licensing policy. Their decisions are presented in the following sections.

Basic License Model

This publisher has identified two distinct end-user profiles and plans to incorporate this knowledge into their sales model. Thus, they have decided to support both node-locked, uncounted and floating models. See Basic License Models in Chapter 6, “License Models,” for more information.

License Model Modifiers

Widget Design will use the following license model modifiers in their implementation.

Required Modifiers

- **Start Date**—An explicit start date is specified in the license certificate to support their subscriptions.

- **Expiration Date**—An explicit expiration date is specified in the license certificate to support their subscriptions and to limit the availability of evaluation versions.
- **Version**—An independent license version scheme, apart from the product version, is implemented.

See the section Required License Model Modifiers in Chapter 6, "License Models."

Optional Modifiers

- A vendor-defined string is used to specify the end-user organization name.
- Supersede functionality is used to support upgrading from the Widget Design Engine Standard configuration to Widget Design Engine Advanced.

See the section Optional License Model Modifiers in Chapter 6, "License Models."

License Server Requirement

Widget Design plans to include a floating license model in their licensing policy offering. Additionally, there could be a future need to track license usage activity. For these objectives, a license server is required.

See the section Determining the Need for a License Server in Chapter 6, "License Models."

Publisher Decisions

When implementing FlexNet Publisher licensing, the software publisher needs to make several decisions regarding product configurations, hostids, check-in and checkout behavior, and the license server.

Licensable Product Configurations

Table 10-1 defines the product configurations and corresponding configuration feature names. Because Widget Design Company wants to distribute the same binary for all product offerings, the product configurations are defined in the license certificates for each product configuration.

Table 10-3 • Widget Design Engine Product Configurations

Product Configuration	Feature Names Included in the License Certificate
Evaluation	• designEngine • editor • import

Table 10-3 • Widget Design Engine Product Configurations

Product Configuration	Feature Names Included in the License Certificate
Standard	• designEngine • editor • import • export
Advanced	• designEngine • editorAdvanced • import • export

Hostid Support

The Ethernet address is the architecture hostid type of choice for each of the three supported platforms: Windows, Solaris, and Linux. In addition, the special hostid type, DEMO, is supported for the evaluation configuration. See Hostid Support in Chapter 7, "Publisher Decisions," for more information.

Checkout Failure Behavior

For checkout failures with the standard or advanced product configurations, Widget Design is allowing the product to run, but only at the evaluation level, which means that export functionality is disabled. Additionally a message, indicating the checkout failure, is displayed to the user.

Checkout failures with the evaluation configuration indicate that the license has expired, so usage is denied. A warning message is displayed to the user with information about upgrading to a full version. See Checkout Failure Behavior in Chapter 7, "Publisher Decisions," for more information.

Checkin Considerations

All licenses remain checked out for the duration of product execution. In a floating license model, the licenses are checked in and returned to the general pool upon product exit. There are no vendor daemon callback or filtering routines—all check-ins are handled in a simple manner.

See Check-in Considerations and Check-in Considerations for Served Licenses in Chapter 7, "Publisher Decisions," for more information.

License Server Consistency

Widget Design has decided to use the default license server reconnection and heartbeat strategies:

- The default operation of the FlexEnabled product when the connection to the license server is lost is to try five times to reconnect and then exit the product.
- Regular heartbeats are automatically initiated when a license is checked out. Heartbeats are exchanged with the license server at a default interval of two minutes.

See Maintaining Consistency with the License Server in Chapter 7, "Publisher Decisions," for more information.

Support for Virtual Machines

Widget Design has opted to allow running of their FlexEnabled product, as well as the license server, on a virtual machine platform. This decision was based on the usage patterns of the product and the trusted geographies in which their product is deployed.

Licensing Security

A major objective for Widget Design is to gain control over unauthorized product usage and thereby increase revenues. They feel that FlexNet Publisher license management gives them most of the security they need against piracy. Considering their marketplace and usage patterns, they are employing additional security measures:

- Using Tamper-Resistant Licenses (TRL) signature encryption.
- Prominently displaying information that identifies the end-user organization when the product starts.
- Issuing expiring licenses, which effectively implements product subscriptions that are renewed on a regular basis.
- For the Windows platform, implementing a static-linked product build model.
- Using the special hostid type, `DEMO`, with the evaluation product configuration.

See Chapter 8, "License Security Systems," for more information about licensing security.

License Fulfillment

Widget Design supports license certificate fulfillment over email and—using the FlexNet Operations Automated Fulfillment option—over the Internet. See License Fulfillment in Chapter 9, "License Fulfillment," for more information.

Usage Rights over Time

Entitlement is controlled using the expiration date in the license certificate and the license version in the product. End users are allowed to download new product versions and corresponding license certificates within their subscription period.

See Strategy for Delivering Rights Over Time in Chapter 9, "License Fulfillment," for more information.

License Rehosting

License rehosting is supported through Widget Designs Help Desk. See Transferring Rights - License Rehosting in Chapter 9, "License Fulfillment," for more information.

Evaluation Licensing Strategy

The licenses for the evaluation product configuration are both time and functionally limited. Providing time-limited license certificates increases the chance that the end user will stay in contact with Widget Design and eventually purchase a fully functional product.

See Evaluation Licensing Strategy in Chapter 9, "License Fulfillment," for more information.

License File Location

The product specifies the location where it expects to find the license certificate. For ease of coexisting with other publishers' FlexEnabled products, the end user can override this by setting the `LM_LICENSE_FILE` environment variable.

Sample License Certificates

Taking into consideration all of the license model decisions Widget Design has made, this section presents sample license certificates for each of their license models.

Evaluation Product Configuration Certificate

The following license certificate is for the evaluation configuration of Widget Design Engine:

```
INCREMENT designEngine widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT editor widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT import widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
```

The following table describes the items in the evaluation license certificate, for the first feature licensed in the certificate. Recall from the section Licensable Product Configurations, that the evaluation configuration includes the designEngine, editor, and import features.

Table 10-4 • Evaluation Product License Certificate Elements

Element	Description
`INCREMENT`	The keyword `INCREMENT` enables the feature whose name immediately follows it. The keyword `FEATURE` may also be used here. The keyword `INCREMENT` (or `FEATURE`) and the information that defines the feature's license is sometimes referred to as a *feature definition line*.
designEngine	The name of the feature licensed in this feature definition line.
widget	The vendor daemon name.
2.0	Specifies entitlement up to and including version 2.0 of the product feature.
dd-mm-yyyy	Specifies license certificate expiration date. For an evaluation license, this will be one month from the start date.
uncounted	Specifies entitlement for unlimited number of licenses.
`START=`*dd-mm-yyyy*	Specifies the date on which entitlement for the feature starts. The license is disabled before this date.
`HOSTID=`*hhhhhh*	Specifies entitlement for the named host.
customer_name	Specifies the customer organization identifier. This information is displayed at product startup.
signature	Specifies the authenticated signature based on all previous fields in the feature definition line. Changing any of the values that precede the signature will invalidate the license.

Because the evaluation license certificate is uncounted, it does not require a license server. The license certificate is installed on the client when the evaluation version of the Widget Design product is installed.

Standard Product Configuration Certificates

Floating License

The floating license is counted and requires a license server, which is identified in the `SERVER` line of the license certificate.

```
SERVER catalina hhhhhh
VENDOR widget vendor_daemon_path

INCREMENT designEngine widget 2.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT editor widget 2.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT import widget 2.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT export widget 2.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
```

The following table describes the items in the standard license certificate for a floating license model, for the first feature licensed in the certificate. Recall from the section Licensable Product Configurations, that the standard configuration includes the designEngine, editor, import, and export features.

Table 10-5 • Standard Product License Certificate for Floating Licenses

Element	Description
`catalina` *hhhhhh*	Specifies the license server to which the certificate is locked, along with the license server's host ID.
`VENDOR widget`	Specifies the name and location of the widget vendor daemon.
`INCREMENT`	The keyword `INCREMENT` enables the feature whose name immediately follows it. The keyword `FEATURE` may also be used here. The keyword `INCREMENT` (or `FEATURE`) and the information that defines the feature's license is sometimes referred to as a *feature definition line*.
designEngine	The name of the feature licensed in this feature definition line.
widget	The vendor daemon name.
2.0	Specifies entitlement up to and including version 2.0 of the product feature.
dd-mm-yyyy	Specifies license certificate expiration date. For a standard license, this will be two years from the license start date.
5	Specifies entitlement for five concurrent licenses.
START=*dd-mm-yyyy*	Specifies the date on which entitlement for the feature starts. The license is disabled before this date.
customer_name	Specifies the customer organization identifier. This information is displayed at product startup.
signature	Specifies the authenticated signature based on all previous fields in the feature definition line. Changing any of the values that precede the signature will invalidate the license.

Node-locked License

A node-locked license is bound to the host that is identified by the `HOSTID` value. Because this license is uncounted, it does not require a license server.

```
INCREMENT designEngine widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT editor widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT import widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
INCREMENT export widget 2.0 dd-mm-yyyy uncounted START=dd-mm-yyyy \
    HOSTID=hhhhhh VENDOR_STRING="For the use of customer_name" SIGN=signature
```

The following table describes the items in the standard license certificate for a node-locked license model, for the first feature licensed in the certificate. Recall from the section Licensable Product Configurations, that the standard configuration includes the designEngine, editor, import, and export features.

Table 10-6 • Standard Product License Certificate for Node-locked Licenses

Element	Description
`INCREMENT`	The keyword `INCREMENT` enables the feature whose name immediately follows it. The keyword `FEATURE` may also be used here. The keyword `INCREMENT` (or `FEATURE`) and the information that defines the feature's license is sometimes referred to as a *feature definition line*.
designEngine	The name of the feature licensed in this feature definition line.
widget	The vendor daemon name.
2.0	Specifies entitlement up to and including version 2.0 of the product feature.
dd-mm-yyyy	Specifies license certificate expiration date. For a standard license, this will be two years from the license start date.
uncounted	Specifies entitlement for unlimited number of licenses.
`START=`*dd-mm-yyyy*	Specifies the date on which entitlement for the feature starts. The license is disabled before this date.
`HOSTID=`*hhhhhh*	Specifies entitlement for the named host.
customer_name	Specifies the customer organization identifier. This information is displayed at product startup.
signature	Specifies the authenticated signature based on all previous fields in the feature definition line. Changing any of the values that precede the signature will invalidate the license.

License certificates for the Widget Design Advanced product configuration look the same as the license certificates provided for the Widget Design Standard product configuration, except that the advanced product configuration substitutes the advanced editor (feature name `editorAdvanced`) for the standard editor (feature name `editor`).

Updating the License Certificate

An end user may require an updated license certificate—for example, in the case of upgrading to a later version than they originally purchased or when upgrading from a standard version of the product to an advanced version with more functionality.

Upgrading to a New Product Version

When Widget Design upgrades their software from version 2.0 to version 3.0, they issue a new license certificate that upgrades the end user's older certificate. The certificate, a portion of which is shown below, disables the original five floating licenses for the standard product configuration and replaces them with five new licenses good for Widget Design Standard versions up to and including version 3.0.

Note that there is an `UPGRADE` line for every feature that is included with the standard product configuration.

```
UPGRADE designEngine widget 2.0 3.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
UPGRADE editor widget 2.0 3.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
UPGRADE import widget 2.0 3.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
UPGRADE export widget 2.0 3.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SIGN=signature
```

Upgrading from a Standard to Professional Configuration

An end user may decide to move from the standard configuration to the professional configuration. To support this change, Widget Design would issue a superseding license certificate that disables the standard configuration licenses and activates the professional configuration, as shown in the following example:

```
INCREMENT editorAdvanced widget 2.0 dd-mm-yyyy 5 START=dd-mm-yyyy \
    VENDOR_STRING="For the use of customer_name" SUPERSEDE=editor \
    SIGN=signature
```

This license certificate is appended to the original license certificate. Because the only configuration difference between the standard and the advanced configuration is the replacement of the standard editor (feature name `editor`) with the advanced editor (feature name `editorAdvanced`), the new license contains a feature definition line that supersedes the feature definition line used for standard versions of the product.

Product Deliverables

Widget Design ships various components with the FlexEnabled product to the end users, including:

- FlexEnabled application binary for the Widget Design product
- License certificates, depending on the product configurations the end user purchases

- FlexNet Publisher utilities—for Windows platforms, LMTOOLS
- *License Administration Guide*—either as provided by Flexera Software or customized
- List of implementation decisions, as appropriate

Additionally, Widget Design ships the following components as required, for served licenses:

- License server manager—`lmadmin` or `lmgrd`
- Vendor daemon
- For information on error return values see *Flexible API Error Return Values* section of *C/C++ Function Reference guide*.

Glossary

Term	Meaning
activation specification record (ASR)	A signed XML file used with local activation and short code activation that contains information about license rights, configuration settings for the publisher's trusted storage segment, and additional information required by the activation transaction.
activatable license	A license right in trusted storage that can be activated to a FlexEnabled client. See also license group.
activation	In FlexNet Publisher Licensing Toolkit, this is a process used to load license rights into trusted storage. Different activation types include: • programmatic activation • manual activation • short code activation • local activation In FlexNet Operations, this is a process used to generate license rights in either trusted storage and in license files.
activated feature line	See feature definition line.
Activation API	Functions available in the FlexNet Publisher Licensing Toolkit libraries that allow you to initiate transactions and manage license rights in trusted storage. The Activation API is used to develop the functionality in an activation utility.

Term	Meaning
activation client	The system that works with an activation server to receive license rights. This system can be either a license server or a FlexEnabled client. The activation client initiates an activation, repair, or return transaction with an activation server, using an activation utility. The following transaction types can be initiated by an activation client: ● activation transaction ● return transaction ● repair transaction
activation request	A message sent by an activation utility to an activation server that initiates an activation transaction.
activation response	A message sent from an activation server to an activation client, in response to an activation request. This message defines the license rights.
activation server	A system that responds to requests for license rights from an activation client. The activation server can be one of the following: ● a license server ● FlexNet Operations ● an LGT-based activation server An activation server can also perform repair and return transactions.
activation transaction	A sequence of activation request and activation response messages, and related actions that result in license rights being loaded into trusted storage.
activation utility	A utility that initiates an activation transaction, repair transaction, or return transaction by sending a request to an activation server. This utility also processes the response received from the activation server. It uses the Activation API to perform these capabilities. It is written by the software publisher to meet specific business process requirements. This application can be either separate from or embedded in the FlexEnabled application.
anchor	A link between the system and the trusted storage that is used to detect whether trusted storage has been tampered with, deleted, or restored. Anchors are also used to ensure that information in trusted storage about trial licenses persists beyond the expiration date.
ASR file	See activation specification record (ASR).
binding	A capability used with trusted storage where one or more properties of a system, combined together into a unique identifier, are used to ensure that license rights held in trusted storage are not copied to another system.

Term	Meaning
borrowing	A capability available in FlexEnabled applications that enables a user to temporarily obtain a license in such a way that the license can be used locally for a limited period of time while the user is disconnected from the local network.
bulk entitlement	An entitlement created for an order of multiple copies of a single product or suite. Bulk entitlements are used for sales through reseller channels.
checkin	The transaction that a FlexEnabled application performs to return a license after it has been checked out. This operation is not the same as an activation transaction. FlexEnabled applications must perform a checkout and checkin transaction to use a license.
checkout	The transaction that a FlexEnabled application performs to request a license. This transaction is different than activating a license. FlexEnabled applications must perform a checkout and checkin transaction to use a license.
client system	A general term used to define any system that interacts with a server in a client-server relationship. When using FlexNet Publisher Licensing Toolkit there are two types of clients: ● activation client ● FlexEnabled client
concurrent license	A license that can be shared among users by allowing each user to check out and then return the license. These licenses are served to FlexEnabled applications from a license server. If all licenses are being used, an additional user cannot run the FlexEnabled application until one of the other users check in their license. When one user finishes using the license, another user can begin using it. Also called floating license or served license rights.
config response	A response to any request from an activation client that contains information about how the publisher's trusted storage segment must be configured (called the trusted storage segment configuration). This response is processed by the activation utility and results in a segment of trusted storage being created or re-configured for that publisher.
consumer	An individual or small business who purchases a license for their individual use. Compare with enterprise.
counted	A type of license right or license model that defines a multiple quantity of licenses and allows concurrent use of the licenses. This term is taken from the fact that the license server counts the number of licenses that are being used.
debug log file	A file used by the license server to record status and error messages that are useful for debugging the license server. Each license server can have one or more of these files.

Term	Meaning
demo license	See trial license.
duplicate grouping	A capability in the FlexNet Publisher Licensing Toolkit used to uniquely identify a single client (or client checkout request) for the purpose of counting the number of licenses that are used. Duplicate grouping defines a set of rules under which multiple checkout requests can share the same license(s).
embedded activation	See local activation.
encryption seed	A set of values defined by a software publisher that are used when creating a publisher-specific vendor daemon.
end-user	A person who uses a FlexEnabled application. This role is not the same as a license administrator.
end-user system	See client system.
end-user license administrator	See license administrator.
enterprise	An organization that purchases licenses on behalf of a number of individual users. Compare with consumer.
entitlement	A representation of the license model and product that a customer has bought. It is a definition only. The customer will need to fulfill the entitlement so that the license rights can be populated to license files or trusted storage. In FlexNet Operations, there are two types of entitlements: • bulk entitlement • simple entitlement Each entitlement is uniquely identified by an entitlement ID.
entitlement ID	In FlexNet Publisher Licensing Toolkit, an attribute in trusted storage that stores a value used to identify the customer’s entitlement. In FlexNet Operations, a value that uniquely identifies the entitlement.
expiring license	License rights granted to a customer for a limited duration. When the license rights have expired, the licenses can no longer be checked out.

Term	Meaning
feature	In the context of a software application, this is a single unit of capability. In FlexNet Publisher Licensing Toolkit, this is an identifier used to associate a license model to a unit of capability in a software application. The software publisher defines which features in their software application map to a single feature in FlexNet Publisher Licensing Toolkit. For example, a feature in FlexNet Publisher Licensing Toolkit could represent any of the following: • The entire software application. • A subset of menu items. • A specific set of data available from the software application. • A process in the software application. The software publisher also defines the license model for each FlexNet Publisher Licensing Toolkit feature. This gives software publishers the ability to define a different license model for each feature in the software application.
feature bundle	A collection of features in FlexNet Operations. It exists primarily to enable modularity and re-use.
feature definition line	An entry in a license file or fulfillment record that describes a feature. Each line begins with the keyword FEATURE, INCREMENT, or UPGRADE. In license files, feature definition lines are used with SERVER and VENDOR lines to define the license model. In trusted storage, feature definition lines are used with the fulfillment record properties to define the license model.
feature line	See feature definition line.
file-based activation	See manual activation.
FlexEnabled application	A software application that uses FlexNet Publisher Licensing Toolkit to implement its license models.
FlexEnabled client	A client system where a FlexEnabled application is installed.
FLEXible API	Functions available in the FlexNet Publisher Licensing Toolkit libraries that allow FlexEnabled clients to access license rights.
FlexNet Agent	A utility that runs on a license server and allows communication between the license server and other FlexNet products.
FlexNet Manager	This is a license server management and license management decision support solution in the FlexNet product family that allows companies to centrally manage license servers and license usage across the enterprise.
FlexNet Operations	A product in the FlexNet family of products that manages entitlements and generates license rights.

Term	Meaning
floating license	A license that can be shared among users and is served to FlexEnabled applications from a license server. When a user finishes using the license, another user can begin using it. Also called concurrent license or served license rights.
fulfill	The action of getting license rights.
fulfillment record	In FlexNet Publisher Licensing Toolkit, a data structure specific to trusted storage that defines license rights.
heartbeat	A type of message sent periodically between two systems (for example, a FlexEnabled application and a license server) to identify whether each is running and able to communicate with the other.
hop count	In FlexNet Publisher Licensing Toolkit, the number of times that license rights in a single fulfillment record have been transferred from one license server to another.
hostid	In FlexNet Publisher Licensing Toolkit, the value of a specific system attribute that uniquely identifies the host. For example, if IP address is a hostid type, then the hostid might be 129.87.33.101. Hostids are used in node-locked license.
hostid type	A type of attribute that is available to uniquely identify a system. For example, disk serial number, IP address, or MAC address.
hybrid license	In FlexNet Publisher Licensing Toolkit, a type of license right in the license group on trusted storage that can function as either a concurrent license or as an activatable license.
internet activation	See programmatic activation.
keyboard activation	See short code activation.
license administrator	A person in an enterprise who is responsible for installing the FlexEnabled application and administering the licenses and license servers.
license file	A text file, usually with the .lic extension, that contains license certificates from one or more publishers.
license-file list	See license search path.
license generator	A general term used to describe any mechanism a software publisher uses to generate license rights for their customers. The following FlexNet capabilities and products can function as a license generator: ● LGT-based activation server: generates license rights for trusted storage. ● FlexNet Operations: generates license rights for trusted storage and license files. ● lmcrypt: generates license rights for license files.

Term	Meaning
License Generator API (LGAPI)	A set of functions in the FlexNet Publisher License Generator Toolkit (LGT) used to create an activation server.
License Generation Toolkit (LGT)	A product in the FlexNet product line used to create an activation server.
license group	A mechanism used to describe licenses rights held in server-side trusted storage. These groups are not used with license rights held in license files. ● **activatable**—A license right in trusted storage that can be activated to a FlexEnabled client. ● **concurrent**—license rights in this group can be shared among users by having a FlexEnabled client to check out, and then return the license. ● **hybrid**—these license rights have the same characteristics as both the activatable license group and concurrent license group. They can be activated to client-side trusted storage on a FlexEnabled client or checked out by a FlexEnabled client. Although you see these group names used to describe license rights in client-side trusted storage, the capabilities of the license rights do not change. Once license rights are in client-side trusted storage, you cannot transfer, serve, or activate them to another system. The FlexEnabled application can only check out and check in these licenses. The group names are used when a license right is returned to the license server, so that it is assigned to the correct license group on the license server.
license model	The set of characteristics that determine the conditions under which (for example, how, when, where, and by whom) a FlexEnabled application can be used.
license rights	Information in trusted storage or license files that defines the license model and controls user access to a specific FlexEnabled application.
license search path	A property that the FlexEnabled application uses to locate where license rights are stored. Sometimes known as a license-file list.

Term	Meaning
license server	A set of files from the FlexNet Publisher Licensing Toolkit, that work together to manage and serve licenses to FlexEnabled applications. A license server is required to support a concurrent license model. The license server consists of a license server manager, one or more vendor daemons, zero or more options files, zero or more debug logs, and, optionally, a report log. These components are available in the FlexNet Publisher Licensing Toolkit. If the license server supports license rights held in trusted storage, it also includes an activation utility. It may, optionally, act as an activation server. ***Note •*** *A license server can accept a maximum of 10,000 connections when running on Windows platform.*
license server system	See license server.
license server manager (lmadmin or lmgrd)	A license server manager `lmadmin` or `lmgrd` executable that forwards connections from a FlexEnabled application to the correct vendor daemon. Because the vendor daemons are responsible for authenticating requests and serving licenses, a single license server manager can be used with multiple vendor daemons on the license server.
license sharing	A capability available in FlexNet Publisher Licensing toolkit where multiple license requests from the same user, host, or display results in only one license being used. More commonly known as duplicate grouping.
licensing client	See FlexEnabled client.
lmadmin	The license server manager introduced in version 11.6 which includes a Web-based GUI.
lmgrd	A command-line-based license server manager.
local license rights	License rights that are stored on the same system where the FlexEnabled application is running, rather than on a license server.
local activation	A type of activation where license rights are loaded using an ASR file. The ASR file must reside with the FlexEnabled application. This does not require user intervention.
machine ID	A value, generated using attributes of the system, that is used to bind trusted storage to that system.
maintenance	A maintenance represents a customer's right to obtain updates and upgrades to a particular licensed product or suite for a specified duration. Used in FlexNet Operations.

Term	Meaning
manual activation	An type of activation where license rights are transmitted from an activation server to another system using a set of request and response messages that are formatted as XML. These messages are saved to and read from physical XML files. An activation utility loads the license rights from an XML file into trusted storage.
node-locked license	A license that can be used only when the FlexEnabled application is run on a specific system or by a specific user (as defined by the hostid).
options file	A configuration file available on the license server that license administrators can use to either change license server behavior or implement certain licensing policies. Each software publisher has its own separate options file that works with a specific vendor daemon.
orderable	Anything that can be purchased by a customer. Orderables can include suites, products, documentation, and maintenances.
package	In FlexNet Publisher Licensing Toolkit, a set of features that can be grouped and licensed together.
part number	A unique identifier, typically correlated with an order system, assigned by a publisher to a product.
permanent license	License rights granted to a customer that do not expire.
product	Anything that can be sold by a publisher and can be licensed.
product ID	In FlexNet Publisher Licensing Toolkit, one of the values that uniquely identifies the license rights in trusted storage. See also entitlement ID.
programmatic activation	A type of activation where license rights are transmitted from one system to another across a network using a set of request and response messages that are formatted as XML. This process is the identical to manual activation, except that it does not generate a physical XML file. An activation utility loads the license rights from the XML message into trusted storage. This is an automated process and does not require user intervention.
publisher ID	A value used by trusted storage to uniquely identify the software publisher's logical and physical licensing components. Flexera Software distributes this identifier to software publishers. It is used to prepare certain FlexNet Publisher Licensing toolkit files that access trusted storage. This value is not the same as the vendor daemon.
quorum	A condition specific to three-server redundancy that is met when at least two of the three license servers are running and can communicate with each other using heartbeats.

Term	Meaning
rehost	The process of reissuing a license for a different system. This process requires software publishers to perform certain activities to redefine the license rights so that they will work properly on the new system.
repair request	A message sent by an activation utility to an activation server that initiates a repair transaction. The activation server provides the information that repairs the license rights in trusted storage. An activation utility is an application that initiates the repair transaction.
repair response	A message sent from an activation server to an activation client, in response to a repair request. This message contains information that is used to the repair the license rights.
repair transaction	A sequence of repair request and repair response messages, and the related actions that result in license rights being repaired in trusted storage.
report log file	A file that runs on a license server that contains data about the features used by a single vendor daemon. Report logs are encrypted and cannot be read by a person, but are used by FlexNet Manager to produce reports.
return request	A message sent by an activation utility to an activation server that initiates a return transaction. The activation server receives the license rights returned by an activation client. An activation utility is an application that initiates the return transaction.
return response	A message sent from an activation server to an activation client, in response to a return request.
return transaction	A sequence of return request and return response messages, and the related actions, that result in license rights being returned from trusted storage to the activation server.
served license rights	License rights that are stored on a license server and are checked out when needed by FlexEnabled applications that run on client systems. These license rights are used for concurrent licensing.
short code	A multi-character code used with a short code activation transaction, a repair transaction, or a return transaction.
short code activation	A type of activation where license rights are loaded from an ASR file using an activation transaction that requires a specific code (called a short code). Often the user enters this code into a prompt from the application. This is also is used for licensing via automated telephony.
signature	A secure multi-character string added to the license certificate that ensures it has not been modified.

Term	Meaning
simple entitlement	An entitlement for a single, specific customer.
suite	A set products grouped together that are functionally complementary and associated with one or more license models.
supersede	A capability in FlexNet Publisher Licensing Toolkit, used to replace the characteristics of old license rights with new characteristics.
three-server redundancy	A capability in the FlexNet Publisher Licensing Toolkit that enables license administrators to configure three license servers in a failover cluster.
triad	A set of three license servers which operate together to support three-server redundancy.
trial license	License rights granted to a user for a limited duration so that the user can evaluate the software application.
trusted storage	A tamper-proof area on the system that stores license rights. This storage location is different than license files and requires a different mechanism for loading and managing license rights. Compare with license file.
trusted configuration	A set of attributes that define the security configuration of a segment of trusted storage. Each software publisher can configure their segments of trusted storage differently. This configuration defines the settings for binding, anchoring, and windback detection.
trusted configuration file	A file that stores the attributes that define a software publisher's trusted storage segment configuration.
trusted storage configuration	See trusted configuration.
uncounted	A license right attribute that allows an unlimited number of concurrent uses of a FlexEnabled application. Compare with counted.
unserved license rights	These are license rights that are not served from a license server. These license rights are located with the FlexEnabled application. Also known as local license rights.
upgrade	A process where a user installs a newer version of a FlexEnabled application and also updates the license rights so that they can use this newer version.
upsell	The process of transitioning from a product of lesser functionality to one of greater functionality in the same product line.

Term	Meaning
vendor daemon	One of the files that is a part of the license server. This executable is customized and built by the software publisher using files in the FlexNet Publisher Licensing toolkit. The vendor daemon is responsible for communicating with the FlexEnabled application and issuing licenses.
Vendor Certificate Generator	A publisher-built component, that is a part of FlexNet Operations, that creates license certificates or signed feature lines in fulfillment records. This functions similarly to lmcrypt.
vendor name	A value used to uniquely identify a software publisher's logical and physical licensing components. It is used to prepare certain files in the FlexNet Publisher Licensing toolkit that manage license rights in both trusted storage and license files. Flexera Software distributes this identifier to software publishers. This value is not the same as the publisher ID.
windback	A condition where the system clock has been set to an earlier date or time. One reason users change the system clock is to extend the license duration.
windback detection	A capability in FlexNet Publisher Licensing Toolkit that detects whether a user has reset the system clock. This prevents a user from extending the duration of the license.
windback tolerance	An acceptable amount of windback to a system clock that won't trigger windback detection. A certain tolerance is normally allowed to account for time zone differences or Daylight Saving Time.

Index