FLEXERA SOFTWARE

FLEXERA SOFTWARE™
FlexNet Publisher®

FlexNet Publisher 2016 (11.14.0)

Getting Started with Trusted Storage–Based Licensing

Legal Information

Book Name: Getting Started with Trusted Storage–Based Licensing

Part Number: FNP-11140-GSTS00

Product Release Date: May 2016

Copyright Notice

Copyright © 2016 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 Getting Started with Trusted Storage Licensing 7

Basic Tutorial Guidelines 7

Windows Platform Used 7

Activation Utility Command-Line Arguments Used 7

License Server Manager Used 8

Locating the Toolkit Directories 8

Intended Audience 9

FlexNet Publisher Licensing Toolkit Product Documentation 10

2 FlexNet Publisher Licensing Concepts 13

Trusted Storage Licensing Concepts 13

License Rights Representation 13

License Lifecycle vs. License Enforcement 14

License Groups 14

Run-time Architecture 14

Trial Licenses 15

Paid Licenses 15

Paid Volume Licenses 16

Local Licenses 16

3 EZCalculator Demonstration 17

Before Starting the Tutorials 18

Configuring Your Machine for Trusted Storage 18

Installing the Licensing Service (on Windows) 18

Initializing the Trusted Storage Directory (on UNIX) 19

Installing a Trial License 19

Trial Licensing Components 20

Application Activation Utility 20

FlexEnabled Application . . . 20
Activation Specification Record . . . 20
Activating Local License Rights . . . 20
Resetting Trusted Storage . . . 21
Using EZCalculator Without License Rights . . . 22
Adding License Rights . . . 22
Viewing Trusted Storage . . . 23
Checking Out a License . . . 24
Attempting to Extend the Trial Without Authorization . . . 24
Resetting the Trial . . . 25
Adding License Rights to a License Server . . . 26
What You Should Know About the Tutorial . . . 26
Prepped Trusted Configuration Used . . . 26
Same Machine Used for License Server and Client . . . 26
Deleting Client-Side and Server-Side Trusted Storage . . . 27
Loading License Rights into Server Trusted Storage . . . 27
Verifying That No Server-Side Trusted Storage Information Exists . . . 28
Loading License Rights From the ASR . . . 28
Checking Out a Hybrid License . . . 29
Starting the FlexNet License Server . . . 30
Viewing License Server Trusted Storage Remotely . . . 31
Checking Out a License . . . 32
Activating a Local License . . . 32
Activating a License . . . 33
Backing Up Trusted Storage . . . 36
Returning a Local License . . . 36
Reinstating Unauthorized Local License Rights . . . 38
Repairing Local Trusted Storage . . . 39
Using Activation Policies for Virtualization . . . 42
What You Should Know About the Tutorial . . . 42
More About the Virtualization Policy . . . 42
Preparing for the Tutorial . . . 42
Verify if your Machine is Virtual . . . 43
Verify Your FlexNet Publisher Toolkit . . . 43
Install the Licensing Service on the Virtual Windows Machine . . . 43
Using Activation Policies for Virtualization . . . 44
Viewing the Virtualization Policy in ASRs . . . 44
Removing Client-Side Trusted Storage . . . 44
Loading Virtualization-Only License Rights on Your Virtual Machine . . . 45
Attempting To Load Physical-Machine-Only Licenses on a Virtual Machine . . . 46
The Next Step . . . 46

4 Additional EZCalculator Exercises . . . 47
What You Should Know About the Tutorial . . . 47
Composite Transactions Used . . . 47

Same Machine Used for License Server and Client . . . 47
Manual Response Generator . . . 48
Preliminary Tasks . . . 48
Reviewing Tutorial Basics . . . 48
Shutting Down the Vendor Daemon . . . 48
Removing Trusted-Storage Files . . . 49
Performing Multiple Activations . . . 49
Creating the Request . . . 49
Generating the Response . . . 50
Activating the Licenses . . . 50
Performing a Partial Upgrade . . . 52
Creating the Request . . . 52
Generating the Response . . . 52
Upgrading the Licenses . . . 52
Performing a Partial Return . . . 54
Creating the Request . . . 54
Generating the Response . . . 54
Returning the Licenses . . . 54
Conclusion . . . 55

5 Short-Code Transaction Scenarios . . . 57
Introduction . . . 57
Overview of the short code mechanism . . . 57
Server and retail transactions . . . 58
Server transactions Only . . . 58
Case 1: An activation transaction . . . 60
Case 2: An incremental activation transaction . . . 61
Case 3: A return transaction . . . 61
Case 4: A repair transaction . . . 62
Managing pending requests . . . 63
Customizing the tutorial for alphanumeric codes, UMN length and different signing strengths . . . 64
Using different response signing strengths . . . 64
Alphanumeric codes option . . . 65
To include different length UMNs in the request . . . 65
Other platforms . . . 65
Troubleshooting . . . 66
The short code activation API . . . 66

6 FlexNet Publisher Implementation Methodology . . . 71
Methodology Roles and Goals . . . 71
Phase One: Defining Product Features, Orderables, and License Models . . . 73
Phase Two: Implementing Licensing . . . 75
Building the Licensing Toolkit . . . 75
Protecting Product Features with Licensing Routines . . . 75
Configuring Secure Data Types . . . 76

Building Your FlexEnabled Application 76
Processing Your FlexEnabled Application 76
Phase Three: Automating the License Lifecycle 77
About Activation Utilities 77
Creating the Activation Utilities 78
Activation Utilities for EZCalculator 78
Configuring Trial Licensing 79
Creating the Installation Package 80
Phase Four: Testing Licensing Behavior 80
Defining Licensing and Lifecycle Test Scenarios 80
Creating the Testing Environment 80

Glossary 83

Index 95

1

Getting Started with Trusted Storage Licensing

This manual provides an introduction to FlexNet Publisher Licensing Toolkit's trusted storage functionality. The manual discusses the following:

- Licensing concepts and methodologies that showcase the most significant capabilities of FlexNet Publisher
- A licensing team model, covering the organizational roles involved in the development of a licensing lifecycle
- A glossary of FlexNet Publisher terms

Basic Tutorial Guidelines

Windows Platform Used

To get the most out of the tutorials presented in this chapter, you need a copy of the FlexNet Publisher Licensing toolkit installed on a Windows platform (either the i86_n3 or x64_n6 toolkit). Each tutorial uses sample utilities and executables that ship with the Windows toolkit. These same components ship with UNIX toolkits. However, to run the tutorials on a UNIX machine, you must adjust the path and file-name syntax for the UNIX environment.

Activation Utility Command-Line Arguments Used

Activation utilities are designed so that the end-user can enter required input via explicit command-line arguments. These tutorial use command-line arguments to issue activation commands. These arguments represent just one way of supplying input to the utilities; this method can differ from your specific requirements as a publisher.

Additionally, all commands in this tutorial are issued in relation to the `<platform_dir>` directory under your installation directory. For example, if you have installed the Windows 64-bit toolkit, you issue commands from a command prompt at the `x64_n6` directory.

License Server Manager Used

When a tutorial exercise requires a license server manager, the steps use `lmadmin`. See the *License Administration Guide* and the `lmadmin` Online help for details about this license server manager.

Note • *If necessary, you can use the alternative license server manager,* `lmgrd`*, in place of* `lmadmin` *in the tutorials.)*

Additionally, the tutorials are designed to use the sample vendor daemon, called `demo`, which ships with the Licensing toolkit.

Locating the Toolkit Directories

The procedures in these tutorials assume you have installed the FlexNet Publisher Licensing Toolkit on a Windows machine. Table 1-1 lists the directories that are involved in the tutorial exercises along with their shortcuts. Take a few moments to identify the location of each shortcut for your particular installation before proceeding. These shortcuts are used throughout this document.

Table 1-1 • Toolkit Directories

Shortcut Name	Default Location
install_dir	No default. You define this path at installation time.
platform_dir	`<install_dir>\i86_n3` (or `<install_dir>\x_64_n6`)
`machind`	`<install_dir>\machind`
`examples`	`<install_dir>\examples`
`publisher`	`<install_dir>\i86_n3\publisher` (or `<install_dir>\x64_n6\publisher`)

Intended Audience

The information and tutorials in this chapter are intended for those individuals at your organization who are responsible for enabling licensing in your product. The tutorials assume some prior experience with product-licensing implementations using FlexNet Publisher Licensing toolkits. The chapter contains the following sections:

Table 1-2 • *Getting Started with Trusted Storage-Based Licensing* Chapter Overview

Topic	Content
Getting Started with Trusted Storage Licensing	Provides an overview of the concepts and tutorials presented in this guide.
FlexNet Publisher Licensing Concepts	Gives an overview of trusted-storage-based licensing concepts, including license rights representations, license lifecycle, license enforcement, license groups, and run-time architecture.
EZCalculator Demonstration	Demonstrates various licensing-activation activities. The tutorials simulate local activation for a trial license, license activation from a publisher's activation server to a FlexNet license server, and license activation from a licence server to a FlexEnabled client: • Installing a Trial License • Adding License Rights to a License Server • Checking Out a Hybrid License • Activating a Local License • Returning a Local License • Repairing Local Trusted Storage Additionally, this chapter contains a tutorial that simulates how policies can restrict the installation of a trial-license to virtual machines only or physical machines only: • Using Activation Policies for Virtualization
Additional EZCalculator Exercises	Presents tutorials that use composite transactions to perform multiple activations, partial upgrades, or partial returns in a single step: • Performing Multiple Activations • Performing a Partial Upgrade • Performing a Partial Return
Short-Code Transaction Scenarios	Provides tutorials that demonstrate short-code transactions between a FlexEnabled client and a publisher's activation server: • Case 1: An activation transaction • Case 2: An incremental activation transaction • Case 3: A return transaction • Case 4: A repair transaction

Table 1-2 • *Getting Started with Trusted Storage-Based Licensing* Chapter Overview

Topic	Content
FlexNet Publisher Implementation Methodology	Gives you a preview of how to implement FlexNet Publisher licensing technology into your own products.
Glossary	Serves as a reference to key FlexNet Publisher terms used throughout FlexNet Publisher documentation.

FlexNet Publisher Licensing Toolkit Product Documentation

The following additional documents provide information about FlexNet Publisher Licensing Toolkit:

- *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 the type of FlexNet Publisher licensing that uses license files. The chapter includes an evaluation procedure that introduces these licensing concepts from both the software publisher and end-user's perspective.
- *Getting Started with Trusted Storage-Based Licensing* is an introduction to the type of FlexNet Publisher licensing that uses trusted storage. The chapter includes tutorials that use sample files in the Licensing toolkit to demonstrate how licenses are activated and managed in various licensing models.
- *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 programmer information about FlexNet Publisher license-file-based 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 Publisher licensing if you have no previously experience its implementation.
- *Programming Reference for Trusted Storage-Based Licensing* contains programmer information about FlexNet Publisher trusted-storage-based licensing. If you have no previous experience in implementing this type of licensing, you should also read *Getting Started with Trusted Storage-Based Licensing* to understand the basic trusted-storage-based licensing concepts.
- *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 Publisher licensing toolkit for Java.
- *License Administration Guide* contains information about the setup and administration of a licensing system, including setting up an options file and using licensing utilities.

- For information on error return values, see *Flexible API Error Return Values* section of *C/C++ Function Reference guide*.

2

FlexNet Publisher Licensing Concepts

Before working with trusted storage, you should be familiar with the following FlexNet Publisher concepts:

- Flexible API for licensing
- License files
- License server
- Current manual process of installing license files that are typically bound to a hostid

Trusted Storage Licensing Concepts

Trusted storage licensing includes many concepts, such as License Rights Representation, License Lifecycle vs. License Enforcement, and License Groups.

License Rights Representation

In addition to representing license rights in license files, FlexNet Publisher can also represent license rights in *trusted storage*, a secure location on an end-user or license-server machine in which a publisher can install license rights in the form of fulfillment records. Unlike a license file, which is immutable once it is delivered to an end user, license rights in trusted storage can be updated by the publisher. This capability enables new business models. For example, a publisher can deactivate a license on an end user's machine if the end user so desires. Other examples of new capabilities enabled by trusted storage include:

- **Trial licensing**—Install end-user license rights without interaction with the publisher
- **License rights activation**—Fulfill license rights from a publisher server

License rights in trusted storage can be equivalent to those expressed in license files. Just as license files can be placed on both end-user machines and license-server machines, trusted storage can be created on end-user machines and license-server machines.

License Lifecycle vs. License Enforcement

A clear separation exists between the API used for license enforcement and that used for the license lifecycle.

- Use Flexible API to enforce the license terms and conditions that you have configured in your FlexEnabled application.
- To manage license lifecycles from first-time activation to subsequent transactions (such as returns, rehosts, upgrades, and transfers), use Activation API. Basically, you use Activation API to create an activation utility that contains the business logic used to automate and manage these license-lifecycle processes.

License Groups

License rights in the license server's trusted storage are divided into three license groups. As part of your license model, you decide the distribution of licenses into these groups.

- **Concurrent licenses**—License rights available network wide from a license server that can be shared by a predetermined number of simultaneous instances of the FlexEnabled application.
- **Activatable licenses**—License rights that have been delivered to a license server, but not yet made available for checkout. After being activated on an end-user machine, the rights are available for checkout.
- **Hybrid licenses**—License rights that are available as concurrent licenses. However, when all licenses in the Activatable group are in use, hybrid licenses can be made available as Activatable licenses.

Run-time Architecture

Figure 2-1 depicts the high-level run-time architecture of FlexNet Publisher. Subsequent sections of this chapter explain how the architecture works in the context of different use cases:

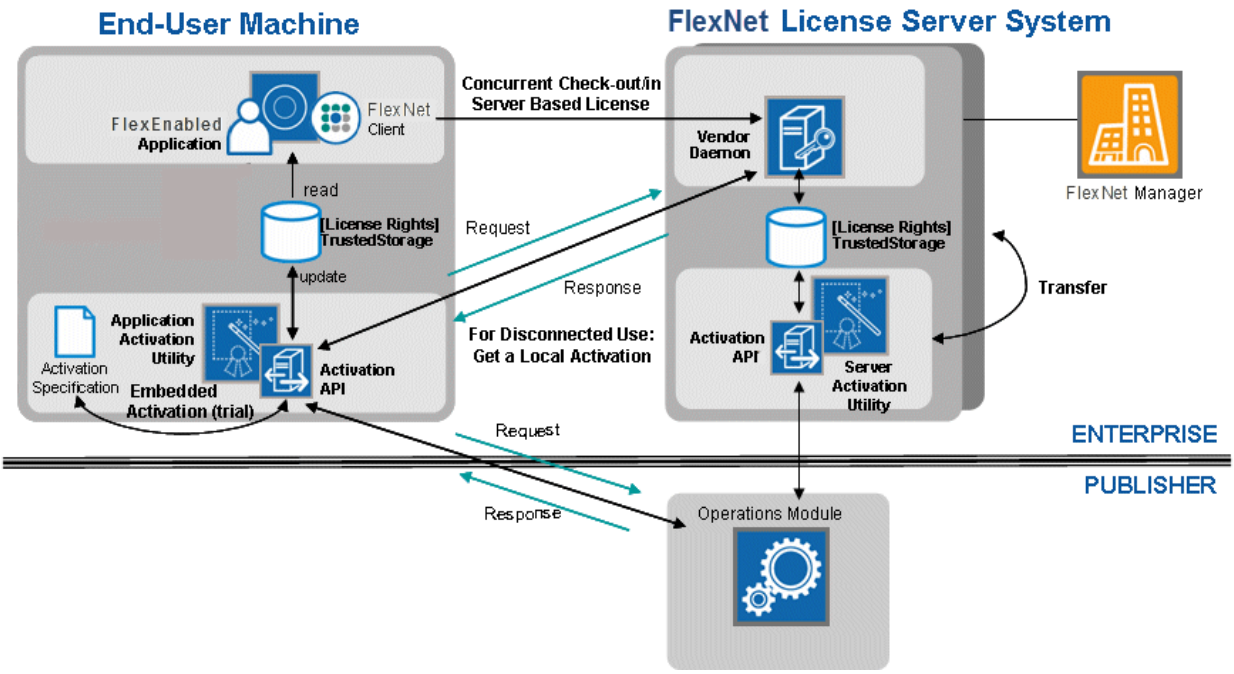

Figure 2-1: FlexNet Publisher Run-Time Architecture

Trial Licenses

For trial licensing, the end user installs a FlexEnabled application. This application includes an activation utility that prompts the end user to install trial license rights for a set duration—for example, 30 days. If the end user wants to activate the trial, the activation utility passes an activation specification record (ASR), included with the installation, to the Activation API. The ASR is a signed XML file that contains details such as the features the trial license contains and the duration. The Activation API uses the contents of the ASR to create trusted storage on the end-user machine with a trial license. When the end user launches the FlexEnabled application, the FlexNet client library finds the license rights in trusted storage and grants or denies usage of the product in the usual manner.

Likewise, a license administrator can obtain an ASR to activate trial license rights on a FlexNet license server, which then serves these licenses to FlexEnabled applications installed on client machines. The license server includes an activation utility that processes the ASR and installs the license rights for a set duration (similar to the ASR-method that installs licenses for FlexEnabled applications on end-user machines).

Paid Licenses

When the specified trial duration (for example, 30 days) has passed, the trial license rights expire. The activation utility that you provide prompts the user to let him know that his trial license has expired. The end user may then be prompted to purchase a paid license from the publisher. Once the end user obtains an Entitlement ID representing a paid license, the activation utility prompts the user to enter the ID. The activation utility then calls the Activation API, which then creates a activation request, calls the publisher's operations system, gets back an activation response, and updates trusted storage with a paid license.

Paid Volume Licenses

Imagine that you want to enable a license model where the enterprise purchases 100 copies of a product and is allowed to deploy these licenses on end-user machines within the enterprise. In this case, the enterprise would install a license server, and charge it up with 100 hybrid licenses. To that end, they would use the server activation utility as shown in Figure 2-1.

The server activation utility, when launched, prompts an enterprise administrator to enter a Entitlement ID, and possibly a count, and expiration. It then:

- Passes this information to the Activation API
- Transmits a request to the publisher's operations system
- Receives an activation response
- Updates trusted storage on the license server with a paid license for 100 hybrid seats

Publishers can also issue an ASR to load these licenses in trusted storage on the license server.

Local Licenses

Going back to the paid volume licensing example, an end user may want to obtain one license from the pool of 100 licenses residing on the license server and install it locally on their machine to allow usage of the product even when the end user is off the network. To accomplish this, the end user uses the application activation utility as shown in Figure 2-1 to make the activation request. The activation utility, when launched, passes the request to the Activation API, which then:

- Transmits the request to the license server.
- Gets back an activation response.
- Updates trusted storage on the end-user machine with one license. These local license rights are used in preference to those available from the license server. They are available when the machine is off the network.

3

EZCalculator Demonstration

This chapter demonstrates several key capabilities of trusted storage using the EZCalculator example files, which are provided in the FlexNet Publisher toolkit and installed on a Windows machine. The usage scenarios in this chapter include:

- Installing a Trial License—A trial license is installed on the machine containing the FlexEnabled application, without interaction with a license server or a publisher's activation server.

  ***Note** • The tutorials in this chapter refer to the machine containing the FlexEnabled application more simply as the* client machine *or the* FlexEnabled client*, depending on the context.*

- Adding License Rights to a License Server—Trusted storage on the license server is loaded with license rights from an activation specification request (ASR).
- Checking Out a Hybrid License—Trusted storage on the client machine accesses the license server's trusted storage to check out a concurrent license.
- Activating a Local License—Trusted storage on the client machine loads license rights from a license server.
- Returning a Local License—License rights on the client machine are returned to the license server.
- Repairing Local Trusted Storage—A compromised segment in the client machine's trusted storage is repaired.
- Using Activation Policies for Virtualization—Activation policies for virtual environments are defined and put into effect via sample ASRs.

Before Starting the Tutorials

Before proceeding with the tutorials in this chapter, complete the following tasks:

- Be familiar with the FlexNet Publisher Licensing Toolkit structure. See Locating the Toolkit Directories for further information. All operations take place in the `<platform_dir>` directory unless otherwise noted.
- Configure your machine to use trusted storage (that is, initialize the trusted storage directory and install the FlexNet Publisher Licensing Service). See the next section for details.
- Review Basic Tutorial Guidelines in Chapter 1, "Getting Started with Trusted Storage Licensing."

Note • *These tutorials are designed to work with the for the default vendor daemon* `demo`*, which ships with the Licensing toolkit.*

Configuring Your Machine for Trusted Storage

Before starting the following tutorials, use the following instructions to configure your machine to use trusted storage. The type of configuration depends on your machine's platform:

- Installing the Licensing Service (on Windows)
- Initializing the Trusted Storage Directory (on UNIX)

Normally, for a licensed product, this configuration occurs as part of product installation and is performed only once on a given machine.

Installing the Licensing Service (on Windows)

On a Windows machine, configuration for trusted storage involves installing the FlexNet Publisher Licensing Service and registering the product as a user of the service. This procedure uses the example executable `installanchorservice.exe`.

Task: ***To install FlexNet Publisher Licensing Service***

1. Locate the example executable:

   `<install_dir>\<platform_dir>\installanchorservice.exe`

2. Log on as administrator.
3. Run `installanchorservice.exe` using the following command:

```
installanchorservice demo EZcalc
```

This installs the FlexNet Publisher Licensing Service and registers that it is being used by the product `EZcalc` supplied by the publisher `demo`.

Initializing the Trusted Storage Directory (on UNIX)

On any UNIX machine, configuration for trusted storage involves initializing the trusted storage directory. Additionally, for a Mac OS X and Linux machine, configuration also involves installing the FlexNet Publisher Licensing Service. This process uses the example script `installfnp.sh`.

Note • *On Mac OS X platforms, your user account must have* `sudo` *privileges to perform these steps. On all other UNIX platforms, you must have* `root` *permission to perform these steps.*

Task: ***To initialize the trusted storage directory (and install the FlexNet Publisher Licensing Service if required)***

Perform the appropriate step at a command prompt from the `<platform_dir>/publisher` directory:

- On a UNIX machine, enter the following command:

  ```
  sh install_fnp.sh
  ```

- On a Mac OS X machine, enter the following command:

  ```
  sudo sh install_fnp.sh
  ```

You receive the following confirmation message:

```
Configuration completed successfully.
```

Installing a Trial License

A trial license seamlessly and silently provides your end user with a limited evaluation license for your product. This license may have limited functionality, limited duration, or both. A license with limited functionality means that it has been implemented to include only a subset of the available features in your product; limited duration means that the license expiry is set relative to the current system date. When the evaluation license expires, the end user can obtain a full license by initiating a transaction with a server.

This tutorial takes you through a typical end-user interaction with a limited-duration trial license. You perform the following:

- Add limited-duration license rights to trusted storage for a specified product
- Using the FlexEnabled application, try accessing both licensed and unlicensed features
- Using the activation utility, view the contents of trusted storage at various junctures throughout this demo

Trial Licensing Components

In general, certain components, described in the following sections, are required to provide trial-licensing to your end user. For purposes of this tutorial, the tutorial version of these components are prebuilt and provided in the FlexNet Publisher Licensing Toolkit toolkit. Keep in mind that the tutorial presents these components as distinct entities. However, in a live scenario, you can embed some or all of your version of these components in your FlexEnabled application, your installer, or a combination of both to provide a seamless activation experience for your end user.

Application Activation Utility

An application activation utility uses the Activation API to manage the license lifecycle. The utility can be stand-alone, or its functionality can be embedded within a FlexEnabled application. This tutorial incorporates the sample stand-alone utility `appactutil.exe`, located in the `<platform_dir>` directory of the Licensing toolkit. In the tutorial, this utility performs the following:

- Identifies the location of the activation specification record (ASR) containing the license rights
- Adds the license rights to trusted storage
- Views the contents of trusted storage

FlexEnabled Application

This tutorial uses the sample application EZCalculator (`ezcalc.exe`) as the FlexEnabled application that accesses license rights in trusted storage. This sample ships with the toolkit.

Activation Specification Record

The activation specification record (ASR) contains license rights that the activation utility loads into trusted storage. The license rights in an ASR are not available for checkout until they are written into trusted storage. Typically, an ASR has an expiration date.

Activating Local License Rights

This tutorial demonstrates how to activate trial license rights locally in the client machine's trusted storage using an activation specification record (ASR). This process is referred to as *local activation*. The tutorial also shows what happens when a person attempts to remove the trusted storage and reload the ASR.

Resetting Trusted Storage

If this is your first time through this tutorial, proceed to Using EZCalculator Without License Rights. Otherwise, before proceeding with the tutorial, use the following procedure to remove any client-side (for the FlexEnabled client) and server-side (for the license server) trusted-storage files that might still exist from previous demonstration or trial-licensing activities.

Task: ***To delete trusted storage***

1. Shut down all license servers that might be running. To do so, from the `<platform_dir>` directory, type the following:

   ```
   lmdown -all
   ```

2. To delete any server-side trusted-storage files from previous demonstration or trial-licensing activities, type the following at a command prompt from the `<platform_dir>` directory:

   ```
   tsreset_svr -delete all
   ```

   The command generates output that includes the following:

   ```
   Performing reset operation, please wait...
   Trusted Storage Contents have been reset...
   ```

3. To delete the client-side trusted-storage files from previous demonstration or trial-licensing activities, type the following at a command prompt from the `<platform_dir>` directory:

   ```
   tsreset_app -delete all
   ```

   The command generates output that includes the following:

   ```
   Performing reset operation, please wait...
   Trusted Storage Contents have been reset...
   ```

Note • *The utilities* `tsreset_app` *and* `tsreset_svr` *are provided for use with the exercises in this tutorial and are not intended to be shipped to an end-user site.*

Using EZCalculator Without License Rights

Task: ***To attempt to use EZCalculator before license rights have been installed***

1. To invoke EZCalculator, enter **ezcalc** at a command prompt from the <platform_dir> directory. The below usage is displayed:

```
EZ Calculator
Current value:0

Enter operator and operand (right hand number) e.g. +5
        + to add
        - to subtract
        x to multiply
        / to divide
        ^ to raise to power
        a to average

        c to clear
        q to clear

Or, just enter a number  >:
```

2. Type the add operator **+** , and press **Enter** The Flexible License Manager error dialog is displayed, stating that the checkout has failed. The failure occurs because no license rights are available to EZCalculator yet.
3. Click **OK** to dismiss the error dialog.
4. To exit EZCalculator, type **q** at the EZCalculator prompt and press **Enter**.

Adding License Rights

Task: ***To add license rights***

1. Verify that trusted storage has no fulfillment record yet. At a command prompt from the <platform_dir> directory, enter the following:

```
appactutil -view
```

The following message is displayed:

```
No fulfillment records in trusted storage
```

2. Activate a license in trusted storage using the trial ASR located in the ..\examples\ezcalc\30day_acct directory. To do so, enter the following at a command prompt at the <platform_dir> directory:

```
appactutil -local ..\examples\ezcalc\30day_acct\30day_acct.asr.
```

The following confirmation message is displayed:

```
Loading ASR ..\examples\ezcalc\30day_acct\30day_acct.asr
Loaded AvailableASRs
```

Viewing Trusted Storage

You can view the contents of trusted storage to see exactly which licenses are activated.

Task: ***To see activated licenses***

At a command prompt from the `<platform_dir>` directory, enter the following:

```
appactutil -view
```

A response similar to the following is displayed:

```
-------------------------------------------------------------------
Trust Flags: FULLY TRUSTED
Fulfillment Type: TRIAL
Status: ENABLED
Fulfillment ID: FIL-EZCALC-A1
Entitlement ID: ENTL-EZCALC-AS1
Product ID: EZCALC-A
Suite ID: NONE
Expiration date: 12-jan-2013
Feature line(s):
INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="005F 03B6 1B41 C196 \
        9754 62AB 5F23 5500 3153 E210 952D B005 BAB4 E5C0 B333"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="00C8 A83F 8D17 \
        545A 497E 01EF 341D 1E00 76D0 4EF2 282D 9434 B9A8 F2D4 16C6"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="007A 413F 3BD4 \
        AA6D 4516 9BE1 A271 1800 4305 535F 0F7E 24E4 AB73 83D8 7A00"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0075 E4A9 80A5 \
        522C 0E68 2165 6CE8 CA00 C8B3 4610 2A99 3138 D9DD 4ACA 5E0C"

-------------------------------------------------------------------
```

Information to Keep in Mind

Note the following information:

- The expiration date is dynamically calculated when the license rights are installed according to rules in the ASR. In this case, a duration of 30 days is used when calculating the expiration date. These license rights were installed on 13 December 2012. The expiration date is 12 January 2013; this date overrides the expiration date of 31 December 2020 listed in the `INCREMENT` lines.
- The `Product ID` value for this fulfillment record is `EZCALC-A`, which is the Accountant edition of EZCalculator.
- Five features are licensed: `ADD`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`, and `AVG`.
- Feature lines include `TS_OK` to enable features to be used with remote desktop sessions on Windows.

Checking Out a License

Task: ***To check out a license***

1. To invoke EZCalculator, enter `ezcalc` at a command prompt from the `<platform_dir>` directory.
2. Type `+`, and press **Enter**. The message `ADD checked out` is displayed.
3. Type the exponential operator `^`, and press **Enter**. The Flexible License Manager error is displayed, stating that checkout has failed. This failure occurs because the operator `^` is not licensed in the Accountant edition.
4. Click **OK** to dismiss the error dialog.
5. To exit EZCalculator, enter `q` at the EZCalculator prompt, and press **Enter**.

Attempting to Extend the Trial Without Authorization

This step simulates an attempt to extend the trial without authorization. The attempt is made by deleting the trusted-storage files that contain fulfillment information for EZCalculator and then reloading the trial.

Task: ***To attempt to extend the trial***

1. Delete `C:\ProgramData\FLEXnet\demo_00000100_tsf.data` and its backup file. These files store trusted-storage fulfillment information for the EZCalculator trial.

   Note • *Consider the following:*

   - *The path used in this step assumes that you are running a Windows 7 machine. For other Windows platforms, the program data might exist in a different directory for other Windows platforms. (As a tip to developers, a common method used to locate the Windows trusted-storage folder is to call* **Win32 API SHGetFolderPath** *with the* `CSIDL_COMMON_APPDATA CSIDL` *value and then append* `FLEXnet` *to the resulting string.)*
   - *If you do not see this file in your Explorer window, ensure that the* ***Show hidden files and folders*** *option is turned on. In your Explorer window, select* ***Tools | Folder Options | View | Show hidden files and folders****.*
   - *On a UNIX machine, the* `tsf` *file is located in* `/usr/local/share/macrovision/storage/FLEXnet/`*.*

2. To attempt to reload the trial ASR, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   appactutil -local ..\examples\ezcalc\30day_acct\30day_acct.asr
   ```

   An error message similar to the following is displayed:

   ```
   ERROR: flxActCommonLicSpcAddASRs - (50021, 71561, 0)
   ```

Tip • *Look up the 50021 error in* .\machind\activation\include\FlxActError.h. *The error corresponds to* LM_TS_ASR_LOAD_ONCE, *indicating that the trial has previously been loaded. Because this trial is a "load once" trial, it cannot be installed again.*

Resetting the Trial

For purposes of this demo, reset the trial anchor in the ASR and reload trial.

Note • *Trial-resetting is performed for testing purposes only.*

Task: ***To reset and reload the trial***

1. To reset the trial ASR, enter the following at a command prompt from the <platform_dir> directory:

   ```
   appactutil -reset ..\examples\ezcalc\30day_acct_reset\30day_reset.asr
   ```

2. To reload the ASR, enter the following:

   ```
   appactutil -local ..\examples\ezcalc\30day_acct\30day_acct.asr
   ```

3. To view the fulfillment record, enter the following:

   ```
   appactutil -view
   ```

4. Open the 30day_reset.asr and note the following:

   - The PUBLISHER_TRIAL_RESET data dictionary item in the ASR is set to 1 (or a non-zero value). This non-zero value allows you to reset the trial anchor (defined by the PUBLISHER_TRIAL_ID) in the ASR so that you can reload the ASR for testing purposes. Use reset-able ASRs (that is, ASRs with PUBLISHER_TRIAL_RESET=1) in testing environments only, as releasing a reset-able ASR can create a security risk. (The 30day_acct.asr is an example of an ASR that can be generally released.)
   - The publisher should increment the PUBLISHER_TRIAL_ID data dictionary item in the ASR for each new trial released, as a trial with a specific ID can be loaded only once.

For more information about resetting the trial and these specific data dictionary items, refer to the following locations in the *Programming Reference for Trusted Storage-Based Licensing*:

- Chapter 6, "Implementing Trial Licenses and Trial Packs"
- Appendix A, "XML Schemas"

Adding License Rights to a License Server

This tutorial demonstrates the process of loading license rights from an activation specification record (ASR) to trusted storage on a FlexNet license server. The loaded rights are then available to be served to FlexEnabled clients. The ASR method for loading licenses on the license server is used for trials or for performing activations when an internet connection is not available to the user, XML files are too large to send, or activations are required in high-security environments.

For exercises that demonstrate the process of loading license rights from a publisher's activation server (such as a FlexNet Operations server) to trusted storage on a FlexNet license server, see Chapter 4, "Additional EZCalculator Exercises."

What You Should Know About the Tutorial

Prepped Trusted Configuration Used

For security purposes, FlexNet Publisher enforces the policy that ASRs loading license rights into server-side trusted storage contain no Trusted Configuration for creating the initial trusted section in trusted storage. Instead, a prepped Trusted Configuration must be implemented so that the server activation utility processing the ASR automatically creates the initial trusted section from the Trusted Configuration embedded within its code.

This tutorial shows how this policy is enforced, using a sample server activation utility that has been prepped with a Trusted Configuration and an ASR that contains no Trusted Configuration definition. For more information on prepped Trusted Configuration, see the chapter "Configuring Security for Trusted Storage" in FlexNet Publisher's *Programming Reference for Trusted Storage-Based Licensing*. Also see the current FlexNet Publisher Release Notes for the list of toolkit platforms that support prepped Trusted Configuration. For those toolkits that do not support prepped Trusted Configuration, live first-time activations continue to involve both the configuration and activation actions.

Same Machine Used for License Server and Client

This tutorial describes operations that are performed server-side (referring to the FlexNet license server) and client-side (referring to the machine where the FlexEnabled application is installed). For purposes of this tutorial, the same machine is used for both types of operations. However, an actual enterprise environment typically has one or more dedicated license server machines that are distinct from the client machines.

Deleting Client-Side and Server-Side Trusted Storage

Before proceeding with the activities in this tutorial, use the following procedure to remove any client-side (for the FlexEnabled client) and server-side (for the license server) trusted-storage files that might still exist from previous demonstration or trial-licensing activities.

Task: ***To remove trusted-storage files***

1. To delete any client-side trusted-storage files from previous demonstration or trial-licensing activities, enter the following at the command prompt from the `<platform_dir>` directory:

   ```
   tsreset_app -delete all
   ```

2. To delete any server-side trusted-storage files from previous demonstration or trial-licensing activities, enter the following:

   ```
   tsreset_svr -delete all
   ```

   The output for both commands concludes with the following information:

   ```
   Performing reset operation, please wait...
   Trusted Storage Contents have been reset...
   ```

Note • *The* `tsreset_app` *and* `tsreset_svr` *utilities are for tutorial use only and are not intended to be shipped to an end-user site.*

Loading License Rights into Server Trusted Storage

The following exercises execute an ASR on the license server that loads trusted storage on the license server with ten hybrid licenses for EZCalculator's Accountant edition.

Verifying That No Server-Side Trusted Storage Information Exists

Task: ***To verify that no server-side trusted-storage information exists***

At a command prompt from the `<platform_dir>` directory, enter the following:

```
serveractutil -view
```

The following message is displayed:

```
No fulfillment records in trusted storage.
```

Loading License Rights From the ASR

In this exercise, you generate an activation request for ten hybrid licenses for EZCalculator Accountant edition. The exercise uses the `serveractutil` command to configure a trusted section (via the utility's embedded prepped Trusted Configuration) in the license server's trusted storage and to load the licenses in this section.

Task: ***To load the license rights into trusted storage***

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

   ```
   serveractutil -local ..\examples\ezcalc\30day_trial_server\30day_trial_server.asr
   ```

   The following confirmation message is displayed:

   ```
   Loaded AvailableASRs
   ```

   This command should have loaded ten hybrid licenses for EZCalculator Accountant into license-server trusted storage. Additionally, because prepped Trusted Configuration was in effect for this demonstration, the `serveractutil` activation utility, not the ASR execution, configured the initial trusted section into which the licenses were loaded.

2. To confirm the contents of trusted storage, enter the following:

   ```
   serveractutil -view
   ```

   The output should be similar to the following, showing a fulfillment record for ten hybrid licenses of the EZCalculator Accountant edition:

   ```
   ----------------------------------------------------------------------
   Trust Flags: FULLY TRUSTED
   Fulfillment Type: PUBLISHER ACTIVATION
   Status: ENABLED
   Fulfillment ID: FIL-EZCALC-LOCAL-ACCT-SERVED10
   Entitlement ID: ENTL-EZCALC-AS1
   Product ID: NAME=EZCALC-A;VERSION=1.0
   Suite ID: NONE
   Expiration date: 12-jan-2013
   ```

```
Concurrent: 0
Concurrent overdraft:0
Hybrid: 10
Hybrid overdraft: 0
Activatable: 0
Activatable overdraft: 0
Repairs: 0
Feature line(s):

        INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B1 5DCC A0E1 CF5F \
            4D2E 089D 8A3E 8D00 036D 906F C9A0 588A 2A1C 1482 2876"
        INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="00F7 AD8A BCE0 \
            7AC8 A454 FE42 ECC3 6600 20DD A746 347A 5285 8B13 B8F6 3ADE"
        INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="002F FFFC 8B6A \
            5E90 ADDA E96D 40E5 6000 1CE5 3519 B889 C9BC 3CAD 882B DC2F"
        INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="00E4 549A 9880 \
            8538 AB39 126B C608 B200 1057 0D1E 523E 9E7E FB63 A297 0291"
        INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="0022 F023 3754 4C11 \
            F24C F32A 961F C200 3F87 26A4 EB8C 7321 92E8 C2CB B472"

-------------------------------------------------------------------
```

The license rights you just loaded into trusted storage are in the hybrid group and are directly available to be checked out as concurrent licenses. Later on in this section, you will check out a license from these license rights.

Important • *Make a note of the Fulfillment IDs for use later with return or repair transactions.*

Information to Keep in Mind

Note the following information:

- The expiration date is dynamically calculated when the license rights are installed according to rules in the ASR. In this case, a duration of 30 days is used when calculating the expiration date. These license rights were installed on 13 December 2012. The expiration date is 12 January 2013; this date overrides the expiration date of 31 December 2020 listed in the `INCREMENT` lines.
- The `Product ID` value for this fulfillment record is `EZCALC-A`, which is the Accountant edition of EZCalculator.
- Five features are licensed: `ADD`, `SUBTRACT`, `MULTIPLY`, `DIVIDE`, and `AVG`.
- Feature lines include `TS_OK` to enable features to be used with remote desktop sessions on Windows.

Checking Out a Hybrid License

In this tutorial, you start the FlexNet license server and check out a concurrent license from the hybrid group for EZCalculator.

Important • *Make sure that you have performed all the steps in the previous tutorial, Adding License Rights to a License Server, before proceeding with the exercises in this tutorial. Also, review Same Machine Used for License Server and Client.*

Starting the FlexNet License Server

A hybrid license is served by a FlexNet license server. Therefore, before a license can be checked out, the `demo` vendor daemon needs to be imported and the license server started using this daemon. The following procedure describes this process.

Note • *To import a vendor daemon on a trusted-storage machine, you must use a token license file that points to this daemon. The following procedure uses* `demo.lic` *as the token file to import the* `demo` *daemon and start the license server.*

Task: ***To start the license server and import the demo vendor daemon***

1. For purposes of this demo, copy `<platform>\demo.exe` and `<platform>\demo_libFNP.dll` to the `<platform>\lmadmin\demo` directory.

   Note • *On UNIX platforms, copy* `<platform>/demo` and `<platform>/demo_libFNP.so`. *to the* `<platform>/lmadmin/demo` *directory. On AIX, copy* `demo` *and* `demo_server_libFNP_notr.so`.

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:

   ```
   lmadmin -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 demonstration, 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.

   From the Dashboard page in the `lmadmin` user interface's main window, you can monitor the available licenses.

Viewing License Server Trusted Storage Remotely

Once a license server is running, activation clients can remotely view the license details in the license server's trusted storage. To simulate the process of viewing a license server's trusted storage from a FlexEnabled client machine, follow this procedure.

Note • *FlexNet Publisher also provides functionality to view contents of a license server's trusted storage remotely from another license server (using the server activation utility). This functionality is not demonstrated in these exercises.*

Task: ***To view license-server trusted storage from the client***

Run the following command (using the client activation utility):

```
appcomptranutil -serverview 27000@localhost format=full
```

This returns details of all trusted fulfillment records for the entitlement ID ENTL-EZCALC-AS1:

```
Fulfillment Id: FIL-EZCALC-LOCAL-ACCT-SERVED10
      Fulfillment Type: 6 Publisher Activation
           Trust Flags: 0x7 Fully Trusted
                Status: Enabled
        Entitlement Id: ENTL-EZCALC-AS1
            Product Id: NAME=EZCALC-A; VERSION=1.0
       Expiration Date: 12-sep-2015
          Server chain: Server1
       Feature Line(s):

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"

INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"
            Concurrent : 0
           Activatable : 0
                Hybrid : 10
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
                Repair : 0
   Deduction count: 0

-----------------------------------------------------------------
```

This server-side trusted-storage information shown on the FlexEnabled client machine is the same output received when you run `servercomptranutil -v format=long` on the license-server machine itself.

Note • *Best practice is to use* `servercomptranutil -v format=long` *to retrieve server-side trusted storage information. The frequent use of* `appcomptranutil -serverview` *can cause client connections to experience timeouts due to the license server's being busy in processing this request.*

Checking Out a License

Task: ***To check out a license***

1. In the window in which you activated the hybrid licenses (see Adding License Rights to a License Server), invoke `lmflex`. To do so, enter `lmflex` at a command prompt from the `<platform_dir>` directory.
2. To check out a feature, type `ADD` and press **Enter**. The message `ADD checked out` is displayed.
3. On the Dashboard page in the `lmadmin` user interface's main window, click **Concurrent**. After a few seconds, the available features are displayed. Notice that the **In Use** count for the `ADD` feature is `1`.
4. In the window running `lmflex`, press **Enter** to check the license back in.
5. On the Dashboard page in the `lmadmin` user interface, refresh the page, and confirm that the **In Use** count for the `ADD` feature in now `0` (zero).
6. As another monitoring step, view a report of the check-out and check-in activity. To do so, open the vendor daemon debug log, `<platform>\lmadmin\logs\demo.log`. A typical excerpt from the log looks similar to this (if you have performed the previous steps in this exercise):

```
14:53:36 (demo) Using options file: "demo\demo.opt"
14:53:37 (demo) Server started on USER-PC for:       f1
14:53:37 (demo) f1             f3            f4
14:53:37 (demo) ADD            AVG           DIVIDE
14:53:37 (demo) MULTIPLY       SUBTRACT
14:53:37 (demo) EXTERNAL FILTERS are OFF
14:53:37 (demo) Report log started (demo/report.log).
14:54:40 (demo) TCP_NODELAY NOT enabled
15:04:00 (demo) OUT: "ADD" User1@USER-PC
15:04:00 (demo) IN: "ADD" User1@USER-PC
```

7. Keep the license server running.

Activating a Local License

This tutorial demonstrates the process of getting an activatable license from the FlexNet license server's trusted storage and activating it in the FlexEnabled client's trusted storage. This makes it permanently available and locked to the client machine.

Note • *Make sure that you have performed all the steps in the previous tutorials, Adding License Rights to a License Server and Checking Out a Hybrid License, before proceeding with the exercises in this tutorial. Also, review Same Machine Used for License Server and Client.*

Activating a License

Task: ***To activate a license***

1. To delete any client-side trusted-storage files that might still exist from a previous tutorial exercise, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   tsreset_app -delete all
   ```

2. To verify that no trusted-storage information exists for the FlexEnabled client, enter the following:

   ```
   appcomptranutil -view
   ```

   The following message is displayed:

   ```
   No fulfillment records in trusted storage
   ```

3. To activate the local license, enter the following:

   ```
   appcomptranutil -t 27000@localhost commsType=flex -activate ENTL-EZCALC-AS1
   ```

   This command does the following:

   - Creates an activation request that contains the Entitlement ID `ENTL-EZCALC-AS1`.
   - Sends the activation request to the license server at `27000@localhost`.

     Note • *For purposes of this tutorial, the default port number 27000 is used. In your production environment, best practice is not to assign a default port for the license server. Instead, for security purposes, assign a port number outside of the range 27000 through 27009.*

   - Places one fulfillment record from the license server's trusted storage into the client machine's trusted storage.

   The request and response end with the information similar to the following:

   ```
   Server 27000@localhost, comms type FLEX
   Status: 5, Request created
   Status: 7, Connected to remote server
   Status: 8, Request Sent
   Status: 11, Done
   Response processed successfully. Actions were:
   Create  fulfillment "FID-1439462671-000"
   ```

4. To confirm that the client machine's trusted storage now has license rights for the Entitlement ID, `ENTL-EZCALC-AS1`, enter the following:

```
appcomptranutil -view format=long
```

Output similar to the following is displayed:

```
Viewing application trusted storage.
----------------------------------------
        Fulfillment Id : FID-1439462671-000
      Fulfillment Type : 4 Served Activation
           Trust Flags : 0x7 Fully Trusted
                Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : BLR-P7106943|Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
 INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"
-------------------------------------------------------------------
```

5. To view the license server's trusted storage, enter the following:

```
servercomptranutil -view format=long
```

```
Viewing server trusted storage.
-------------------------------------------------------------------------------
        Fulfillment Id : FIL-EZCALC-LOCAL-ACCT-SERVED10
      Fulfillment Type : 6 Publisher Activation
           Trust Flags : 0x7 Fully Trusted
                Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"

         Concurrent : 0
```

```
          Activatable : 0
               Hybrid : 9
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
               Repair : 0
Deduction count : 1
Deduction 0 (only non-zero counts shown)
        Fulfillment ID : FID-1439462671-000
           System name : BLR-P7106943
                Expiry : 12-sep-2015
                  Type : 1
               Hybrid : 1
```

6. On the Dashboard page in the `lmadmin` user interface's main window, click **Activatable**. Notice that the **Detached** count is now 1.

7. To simulate the ability of an activation client (in this case, the FlexEnabled client) to view license details in server trusted storage remotely, run the following command, using the client activation utility:

```
appcomptranutil -serverview 27000@localhost format=full
```

Notice that, like both the `serveractutil -view` command output and the information displayed on the Dashboard page, this output confirms that the hybrid license count has been decremented.

```
Reading fulfillments from server 27000@localhost
1 fulfillments read
Reading full details for fulfillment id FIL-EZCALC-LOCAL-ACCT-SERVED10
-------------------------------------------------------------------------------
        Fulfillment Id : FIL-EZCALC-LOCAL-ACCT-SERVED10
      Fulfillment Type : 6 Publisher Activation
           Trust Flags : 0x7 Fully Trusted
                Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"

            Concurrent : 0
          Activatable : 0
               Hybrid : 9
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
```

```
                 Repair : 0

Deduction count : 1
Deduction 0 (only non-zero counts shown)
        Fulfillment ID : FID-1439462671-000
           System name : BLR-P7106943
                Expiry : 12-sep-2015
                  Type : 1
                Hybrid : 1
```

Backing Up Trusted Storage

In preparation for the next tutorial, back up the current FlexEnabled client's trusted storage to a temporary location.

Task: ***To back up the client-side trusted storage***

From the directory `C:\ProgramData\FlexNet`, copy `Demo_00000100_tsf.data` to `C:\TS_Backup`.

Note • *Consider the following:*

- *The path used in this step assumes that you are running a Windows 7 machine. For other Windows platforms, the program data might exist in a different directory. (As a tip to developers, a common method used to locate the Windows trusted-storage folder is to call* **Win32 API SHGetFolderPath** *with the* `CSIDL_COMMON_APPDATA CSIDL` *value and then append* `FLEXnet` *to the resulting string.)*
- *If you do not see this file in your Explorer window, ensure that the* ***Show hidden files and folders*** *option is turned on. In your Explorer window, select* ***Tools | Folder Options | View | Show hidden files and folders****.*
- *On a UNIX machine, the* `tsf` *file is located in* `/usr/local/share/macrovision/storage/FLEXnet/`*.*

In the next tutorial, you return the existing license rights to the FlexNet license server. You then reinstate the backed up copy of trusted storage in an attempt to surreptitiously use the rights that were returned to the license server.

Returning a Local License

Note • *Make sure that you have performed all the steps in the previous tutorials, Adding License Rights to a License Server, Checking Out a Hybrid License, and Activating a Local License, before proceeding with the exercises in this tutorial. Also, review Same Machine Used for License Server and Client.*

This tutorial demonstrates the process of returning an activated license in the client machine's trusted storage back to the license server's trusted storage.

Task: ***To return an activated license to trusted storage***

1. Determine the fulfillment ID for the license you want to return by viewing the contents of the client machine's trusted storage. To do so, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   appcomptranutil -view
   ```

   The fulfillment ID is displayed in the output:

   ```
   ------------------------------------------------------------------
   U if untrusted
   . D if disabled
   . . fulfillmentType
   . . . Expiry    Count FulfillmentId ProductId
   . . 4 12-sep-2015   1 FID-1439462671-000              NAME=EZCALC-A;VERSION=1.0
   ```

   In this example, the fulfillment ID is `FID-1439462671-000`. Your value can differ.

2. To issue the return request, enter the following:

   ```
   appcomptranutil -t 27000@localhost commsType=flex -return FID-1439462671-000
   ```

   The following message is displayed:

   ```
   Server 27000@localhost comms type FLEX
   Status: 5, Request created
   Status: 7, Connected to remote server
   Status: 8, Request Sent
   Status: 11, Done
   Response processed successfully. Actions were:
    Delete         fulfillment "FID-1439462671-000"
   ```

3. To view the client machine's trusted storage to confirm that it is empty, enter the following:

   ```
    appcomptranutil -view
   ```

   The following message is displayed:

   ```
   No fulfillment records in trusted storage.
   ```

4. To view the license server's trusted storage to confirm the return, enter the following:

   ```
   servercomptranutil -view format=long
   ```

   Notice that the number of hybrid licenses has been increased from 9 to 10, as shown in the following excerpt from the output:

   ```
   Viewing server trusted storage.
   ------------------------------------------------------------------------------
           Fulfillment Id : FIL-EZCALC-LOCAL-ACCT-SERVED10
         Fulfillment Type : 6 Publisher Activation
              Trust Flags : 0x7 Fully Trusted
                   Status : Enabled
           Entitlement Id : ENTL-EZCALC-AS1
               Product Id : NAME=EZCALC-A;VERSION=1.0
          Expiration Date : 12-sep-2015
   ```

```
          Server chain : Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"


           Concurrent : 0
          Activatable : 0
               Hybrid : 10
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
               Repair : 0

Deduction count : 0
```

Reinstating Unauthorized Local License Rights

In this exercise, you reinstate the license rights that you backed up, simulating an unauthorized reuse of license rights that have been returned. This sets the stage for the repair scenario in the next tutorial, Repairing Local Trusted Storage.

Task: ***To reinstate the backed-up trusted storage***

1. Copy `C:\TS_Backup\Demo_00000100_tsf.data` to `C:\ProgaramData\FlexNet`, and delete the trusted-storage recovery file, `demo_00000100_tsf.data_backup.001`.

2. To confirm the contents of trusted storage, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   appcomptranutil -view format=long
   ```

   This command can be abbreviated and used as `appcomptranutil -vl`

   The following output is displayed. Notice that the `Trust Flags` entry indicates trusted storage is broken.

   ```
   ------------------------------------------------------------------

   Viewing application trusted storage.
   ------------------------------------------------------------------------------
           Fulfillment Id : FID-1439463973-002
         Fulfillment Type : 4 Served Activation
   ```

```
           Trust Flags : 0x6 **BROKEN** Restore
                Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : BLR-P7106943|Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"
-------------------------------------------------------------------
```

3. Record the Fulfillment ID of the broken fulfillment record. In this example, the fulfillment ID is `FID-1439463973-002`. (Your value is most likely different.)

4. Shut down the vendor daemon (if it is running) on the FlexNet license server:

   a. From the `lmadmin` user interface's main window, click **Administration** to open the Administration page, and, if prompted, sign in with administrator credentials. (Enter **`admin`** as both the user name and the password.)

   b. Go to the Vendor Daemon Configuration tab, and click **Administer** in the entry for the `demo` vendor daemon.

   c. Click **Stop**.

   Shutting down the vendor daemon makes license rights in the server's trusted storage unavailable. Now, only license rights in the client machine's trusted storage are available.

5. To run EZCalculator, enter **`ezcalc`** at a command prompt from the `<platform_dir>` directory.

6. To attempt to check out a license, type **+** , and press **Enter**. A FlexNet License Manager error message is displayed, stating that trusted storage has been compromised.

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

8. To exit EZCalculator, type **`q`** at the EZCalculator prompt, and press **Enter**.

Repairing Local Trusted Storage

In the previous tutorial, you returned the existing license rights for the EZCalculator engineer edition. Then, to simulate an unauthorized reuse of the rights, you tried to reinstate them yourself from a backup file. The result has rendered your FlexEnabled client's trusted storage useless.

The following tutorial shows you how to repair a fulfillment record in trusted storage. This demonstration does not illustrate a real-life scenario—that is, after you issue a return of license rights, you should never run a repair on the fulfillment record. This action would give the enterprise an additional license.

Note • *Make sure that you have performed all the steps in the previous tutorials, Adding License Rights to a License Server, Checking Out a Hybrid License, Activating a Local License, and Returning a Local License, before proceeding with the exercises in this tutorial. Also, review Same Machine Used for License Server and Client.*

Task: ***To repair trusted storage that has been compromised***

1. Since the repair request is issued to the FlexNet license server, you need to start license server vendor daemon. Do so from the `lmadmin` user interface main window:
   a. If you not already on the Vendor Daemon Configuration tab, return to this tab on the Administration page.
   b. On the Vendor Daemon Configuration tab, click **Administer** in the entry for the `demo` vendor daemon.
   c. Click **Start**.
2. To issue the repair request, enter the following:

```
appcomptranutil -t 27000@localhost commsType=flex -repair  FID-1439463973-002
```

   The output ends with the following message:

```
Server 27000@localhost, comms type FLEX
Status: 5, Request created
Status: 7, Connected to remote server
Status: 8, Request Sent
Status: 11, Done
Response processed successfully. Actions were:
   Repair FulfillmentId:  FID-1439463973-002
               Reason:
              Comment:  Success but note this is a deny action
```

3. To view the contents of the client machine's trusted storage to confirm that trust has been restored, enter the following:

```
appcomptranutil -view format=long

Viewing application trusted storage.
-------------------------------------------------------------------------------
        Fulfillment Id : FID-1439463973-002
      Fulfillment Type : 4 Served Activation
          Trust Flags : 0x7 Fully Trusted
               Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : BLR-P7106943|Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
```

```
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"
```

Notice that the `Trust Flags` field reads `FULLY TRUSTED`.

4. To view the contents of the license server's trusted storage, enter the following:

```
servercomptranutil -view format=long

Viewing server trusted storage.
-------------------------------------------------------------------------------
        Fulfillment Id : FIL-EZCALC-LOCAL-ACCT-SERVED10
      Fulfillment Type : 6 Publisher Activation
           Trust Flags : 0x7 Fully Trusted
                Status : Enabled
        Entitlement Id : ENTL-EZCALC-AS1
            Product Id : NAME=EZCALC-A;VERSION=1.0
       Expiration Date : 12-sep-2015
          Server chain : Server1
       Feature Line(s) :

INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="00B9 53C8 3DA9 \
        EF50 95C6 02B9 EFC7 D800 FBE8 FF70 8288 06EE 39A3 07C0 D4A3"
INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="001B 5198 \
        893E 9D44 37AD 95DB 9735 AF00 2BF5 3404 2BE0 4267 91AE D243 \
        1CCE"
INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="00F4 D167 \
        D873 7FCE 3014 1D91 62C6 9D00 2441 17A6 DF83 5007 9600 2D4B \
        3334"
INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0057 8EE1 5676 \
        62CC 656C B0B3 8B68 A400 9B8E EE6E 71F0 DC5C 1F47 6675 B106"
INCREMENT AVG demo 1.0 31-dec-2020 1 TS_OK SIGN="00A3 DCD7 D7E8 \
        F5F8 8BC9 CDC8 811C 4400 623E 495A 2F5C 4234 FD8E 3A68 F86B"

            Concurrent : 0
           Activatable : 0
                Hybrid : 9
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
                Repair : 0

Deduction count : 1

Deduction 0 (only non-zero counts shown)
        Fulfillment ID : FID-1439463973-002
           System name : MH-Suse11-x86
                Expiry : 12-sep-2015
                  Type : 1
                Hybrid : 1Notice that the number of repairs has been appropriately decremented.
```

Using Activation Policies for Virtualization

FlexNet Publisher supports trusted-storage-based licensing in a virtual environment. This support includes the following:

- Policy that explicitly allows or disallows local activation on FlexEnabled clients running in virtual environments
- Activations, repairs, and returns performed from the publisher's activation server to FlexEnabled clients or FlexNet license servers running in virtual environments
- Within a composite request, details about the virtual machine from which the composite request originates

If your products run in virtual environments, you can set policies that define whether your trial licenses can be loaded in virtual environments only or in physical environments only. For more information about creating trials and defining activation requests and responses for virtual environments, see the *Programming Reference for Trusted Storage-Based Licensing*.

What You Should Know About the Tutorial

This tutorial uses ASRs to demonstrate the following:

- How to activate trial license rights in trusted storage on a FlexEnabled client running on a virtual platform
- How the Virtualization policy determines whether an ASR can activate trial licenses in a given environment

More About the Virtualization Policy

The data dictionary element `VIRTUALIZATION_POLICY` controls whether an ASR can be loaded on physical or virtual machines. The following three scenarios cover the most likely trial requirements for a publisher:

- **Allow trials on both physical and virtual environments**--If you do not want to distinguish between trials on virtual and non-virtual machines, simply omit the `VIRTUALIZATION_POLICY` option.
- **Allow trials only on physical environments**--Set `VIRTUALIZATION_POLICY=PHYSICAL`.
- **Allow trials only on virtual environments**--Set `VIRTUALIZATION_POLICY=VM_ONLY`.

This tutorial uses `30day_acct_virtual.asr` and `30day_acct_physical.asr` to demonstrate how this policy works.

Preparing for the Tutorial

Before running this tutorial, do the following:

- Verify if your Machine is Virtual
- Verify Your FlexNet Publisher Toolkit
- Install the Licensing Service on the Virtual Windows Machine

Verify if your Machine is Virtual

These tutorials assume that you have installed the FlexNet Publisher Toolkit on a Windows machine instance running on a supported virtual platform. The tutorial uses this machine instance to simulate the client machine (that is, the machine running the FlexEnabled client) on which you intend to activate trial licenses.

To ensure that this machine is running on a virtual platform, use this procedure.

Task: ***To check that your machine is running on a virtual platform***

At a command prompt from the `<platform_dir>` directory on the machine, type the following:

```
appactutil -virtual
```

The output would indicates the type of Machine, for example:

```
Running on Virtual Platform.
```

To view the details of the machine used, type the following:

```
appactutil -virtual -long
```

The output indicates the following:

```
Running on Virtual Platform
FAMILY=VMWARE
NAME=VMWARE
UUID=875B4D56-031C-B04C-6226-5B817EC97977
GENID=c58e2918fd732f3c:63b828fa89d67a9a
```

Verify Your FlexNet Publisher Toolkit

Ensure you are using one of the FlexNet Publisher toolkits that supports virtualization:

- i86_n3
- x64_n6
- i86_lsb
- x64_lsb

Install the Licensing Service on the Virtual Windows Machine

If you are using a virtual Windows machine for this tutorial, install FlexNet Publisher Licensing Service on the machine. See Installing the Licensing Service (on Windows) for details.

Using Activation Policies for Virtualization

This tutorial uses two sample ASRs--`30day_acct_virtual.asr` and `30day_acct_physical.asr`--available in your FlexNet Publisher toolkit to demonstrate the use of the `VIRTUALIZATION_POLICY` option in ASRs. In your own ASRs, you can use this option to control whether license installation is restricted to virtual environments only or physical environments only. For more information, see More About the Virtualization Policy.

Note • *A third sample ASR,* `30day_acct_phy_vir.asr`*, not used in this tutorial, defines a policy that allows the licenses to be loaded in any environment, virtual or physical.*

Viewing the Virtualization Policy in ASRs

In this exercise, view the sample ASRs to see how their activation policy for virtual environments are defined.

Task: ***To view virtualization policies***

1. Review the contents of `..\examples\ezcalc\30day_acct_virtual\30day_acct_virtual.asr`.
2. Locate the data dictionary element `VIRTUALIZATION_POLICY`:

   ```
   <Entry>
   <Key>VIRTUALIZATION_POLICY</Key>
   <Value>VM_ONLY</Value>
   </Entry>
   ```

   The `VM_ONLY` value restricts the loading of licenses from this ASR to virtual machines only. A later exercise in this tutorial uses this ASR to load licenses in trusted storage on your virtual client machine.
3. Review the contents of `..\examples\ezcalc\30day_acct_virtual\30day_acct_physical.asr`.
4. Locate the following data dictionary element `VIRTUALIZATION_POLICY`:

   ```
   <Entry>
   <Key>VIRTUALIZATION_POLICY</Key>
   <Value>PHYSICAL</Value>
   </Entry>
   ```

   This `PHYSYICAL` value restricts the loading of license from this ASR to physical machines only. A later exercise in this tutorial uses this ASR to attempt to load licenses in trusted storage on your virtual client machine.

Removing Client-Side Trusted Storage

If this is your first time using this tutorial, proceed to Loading Virtualization-Only License Rights on Your Virtual Machine. Otherwise, before proceeding with this tutorial, use the following procedure to remove any client-side trusted-storage files that might still exist from previous demonstration or trial-licensing activities.

Task: ***To delete any existing client-side trusted-storage files***

1. At a command prompt from the `<platform_dir>` directory, type the following:

   ```
   tsreset_app -delete all
   ```

   The output includes the following information:

   ```
   Performing reset operation, please wait...
   Trusted Storage Contents have been reset...
   ```

2. To verify that trusted storage information no longer exists, enter the following:

   ```
   appactutil -view
   ```

   The following message should display:

   ```
   No fulfillment records in trusted storage
   ```

Note • *The* `tsreset_app` *utility is for tutorial use only and is not intended to be shipped to an end-user site. An alternative to using* `tsreset_app` *is to reset your virtual machine to a snapshot taken before you first established trusted storage.*

Loading Virtualization-Only License Rights on Your Virtual Machine

This exercise installs license rights in trusted storage on your virtual client machine. They are installed using the sample ASR defined with the policy to install license rights on virtual machines only.

Task: ***To load license rights on your virtual machine***

1. To load license rights in trusted storage on the virtual client machine, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   appactutil -local ..\examples\ezcalc\30day_acct_virtual\30day_acct_virtual.asr
   ```

   The following confirmation message displays:

   ```
   Loading ASR ..\examples\ezcalc\30day_acct_virtual\30day_acct_virtual.asr
   Loaded AvailableASRs
   ```

   The ASR you used is defined with `VIRTUALIZATION_POLICY=VM_ONLY`; therefore, the licenses loaded successfully on your virtual machine.

2. To view the activated licenses, enter the following:

   ```
   appactutil -view
   ```

Attempting To Load Physical-Machine-Only Licenses on a Virtual Machine

This exercise attempts to use the sample `30day_acct_physical.asr` to install licenses on your virtual machine. This ASR is defined with the policy `VIRTUALIZATION_POLCY=PHYSICAL`, which restricts license installation to physical machines only.

Task: ***To attempt to load physical-machine-only licenses on a virtual machine***

1. To remove the sample client-side trusted-storage file created on your virtual machine in the previous exercise, enter the following at a command prompt from the `<platform_dir>` directory:

   ```
   tsreset_app -delete all
   ```

2. Verify that the trusted-storage information no longer exists:

   ```
   appactutil -view
   ```

3. Enter the following to load the ASR defined with the `VIRTUALIZATION_POLICY=PHYSICAL` on your virtual machine:

   ```
   appactutil -local ..\examples\ezcalc\30day_acct_virtual\30day_acct_physical.asr
   ```

   The `LM_TS_VIRTUALIZATION_POLICY_MISMATCH` error message is displayed because you attempted to perform a license installation restricted to physical machines only.

The Next Step

Once you have finished the tutorials in this chapter, use the `lmadmin` user interface to shut down the FlexNet license server if it is running. (Navigate to the Vendor Daemon Configuration tab on the Administration page, and click **Administer** in the `demo` entry.)

From here, you can proceed to Additional EZCalculator Exercises for more EZCalculator tutorials that demonstrate the efficiency and flexibility of composite transactions.

4

Additional EZCalculator Exercises

The tutorials in this chapter demonstrate how to use composite transactions to manage license rights on the license server. These tutorials include the following:

- Performing Multiple Activations
- Performing a Partial Upgrade
- Performing a Partial Return

What You Should Know About the Tutorial

Composite Transactions Used

This tutorial demonstrates the use of composite transactions to load licenses from the publisher's activation server. These transactions (called *V2 transactions*) combine multiple activation actions in a single interaction with the publisher's activation server. Although single-action transactions (called *V1 transactions*) are still supported for backward compatibility, composite transactions are favored for their flexibility and efficiency (for example, they allow partial upgrades and returns without losing deduction record data). For more information about composite transactions and creating the activation utilities that support them, see the *Programming Reference for Trusted Storage-Based Licensing*.

Same Machine Used for License Server and Client

These tutorials describes operations that are performed server-side (referring to the FlexNet license server). However, for purposes of these tutorials, the license server and the FlexEnabled client exist on the same machine. An actual enterprise environment typically has one or more dedicated license server machines that are distinct from the client machines.

Manual Response Generator

FlexNet Operations and FlexNet Publisher License Generator Toolkit use *response parameter files* as templates for generating responses. Both of these FlexNet products generate response parameter files automatically as part of the activation process. However, to simulate the role of FlexNet Operations, these tutorial use pre-generated response parameter files and the manual response generator, `responsegen.exe`. This response generator is a *dumb* publisher license server that generates responses based on the input response parameter file.

Preliminary Tasks

Before performing the tutorials in this chapter, you must first shut down the `demo` vendor daemon and reset client-side and server-side trusted storage. You should also review tutorial basics.

Reviewing Tutorial Basics

Do the following if you have not already:

- Be familiar with the FlexNet Publisher Licensing Toolkit structure. See Locating the Toolkit Directories in Chapter 1, Getting Started with Trusted Storage Licensing for further information. All operations take place in the `<platform_dir>` directory unless otherwise noted.
- Install the FlexNet Publisher Licensing Service. See the Installing the Licensing Service (on Windows) section in Chapter 3, EZCalculator Demonstration for details.
- Review Basic Tutorial Guidelines in Chapter 1, "Getting Started with Trusted Storage Licensing."

Note • *These tutorials are designed to work with the for the default vendor daemon* `demo`*, which ships with the Licensing toolkit.*

Shutting Down the Vendor Daemon

If you have not already shut down the vendor daemon, use the following procedure to do so.

Task: ***To shut down the vendor daemon***

1. 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 demonstration, 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.

2. From the lmadmin user interface's main window, click **Administration** to open the Administration page, and, if prompted, sign in with your administrator credentials.
3. Go to the Vendor Daemon Configuration tab, and click **Administer** in the entry for the demo vendor daemon.
4. Click **Stop**.

Removing Trusted-Storage Files

Use the following procedure to remove any client-side (for the FlexEnabled client) and server-side (for the license server) trusted-storage files created in previous demonstration or trial-licensing activities.

Task: ***To remove client-side and server-side trusted storage files***

1. To remove any existing client-side trusted storage files, enter the following at the command prompt from the <platform_dir> directory:

   ```
   tsreset_app -delete all
   ```

2. To remove any existing server-side trusted storage files, enter the following:

   ```
   tsreset_svr -delete all
   ```

The output for both commands concludes with the following information:

```
Performing reset operation, please wait...
Trusted Storage Contents have been reset...
```

Performing Multiple Activations

This tutorial uses a composite transaction to configure trusted storage on the license server and activate five concurrent ezcalc engineer licenses and five hybrid ezcalc student licenses in this trusted storage.

Creating the Request

This request specifies two entitlements:

- ENTL-EZCALC-NS1, which defines concurrent ezcalc engineer licenses
- ENTL-EZCALC-S1, which defines hybrid ezcalc student licenses

Task: ***To create the request for multiple activations***

Enter the following at the command prompt from the <platform_dir> directory:

```
servercomptranutil -new request.xml -activate ENTL-EZCALC-NS1 -activate ENTL-EZCALC-S1
```

The following message is returned:

```
Request written to request.xml
```

Generating the Response

This exercise uses the sample response parameter file `W7_EZCalcSvrSingleStepMultipleActivate` to generate the appropriate response file for activating the licenses.

Task: ***To generate the response***

Enter the following at the command prompt from the <platform_dir> directory:

```
.\publisher\responsegen.exe -i request.xml -o response.xml -k .\publisher\xml\DemoKeyout.xml -m
.\publisher\xml\Version6\W7_EZCalcSvrSingleStepMultipleActivate.xml
```

Activating the Licenses

Process the response file to activate the two groups of `ezcalc` licenses on the license server.

Task: ***To activate the licenses***

1. Enter the following at the command prompt from the <platform_dir> directory:

   ```
   servercomptranutil -process response.xml
   ```

   The following information is returned:

   ```
   Response processed successfully. Actions were:
       Configuration  TrustedId "1"
       Create         fulfillment "FIL-EZCALC-ENGR-SERVED5"
       Create         fulfillment "FIL-EZCALC-STUDENT-SERVED5"

       FLEXnet response dictionary:
               COMMENT=Single Step Multiple Activate - Load 5 concurrent licenses for
                EZCalculator Engineer Edition and 5 hybrid licenses for EZCalculator Student Edition
   ```

2. To view the licenses, enter the following command:

   ```
   servercomptranutil -v format=long
   ```

   The output shows the two activations:

```
-------------------------------------------------------------------------------
        Fulfillment Id : FIL-EZCALC-ENGR-SERVED5
      Fulfillment Type : 6 Publisher Activation
          Trust Flags : 0x7 Fully Trusted
               Status : Enabled
        Entitlement Id : ENTL-EZCALC-NS1
           Product Id : NAME=EZCALC-N;VERSION=1.0
      Expiration Date : 11-Jun-2014
         Server chain : Server1
      Feature Line(s) :

                    INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="005F 03B6 1B41 C196 \
                    9754 62AB 5F23 5500 3153 E210 952D B005 BAB4 E5C0 B333"
                    INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="00C8 A83F 8D17 \
                    545A 497E 01EF 341D 1E00 76D0 4EF2 282D 9434 B9A8 F2D4 16C6"
                    INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="007A 413F 3BD4 \
                    AA6D 4516 9BE1 A271 1800 4305 535F 0F7E 24E4 AB73 83D8 7A00"
                    INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0075 E4A9 80A5 \
                    522C 0E68 2165 6CE8 CA00 C8B3 4610 2A99 3138 D9DD 4ACA 5E0C"
                    INCREMENT POWER demo 1.0 31-dec-2020 1 TS_OK SIGN="00FF 515A 3015 \
                    4ABE 8413 02C6 1214 E100 3670 E9C2 8DFD D2EF D46E 7656 D104"

           Concurrent : 5
          Activatable : 0
               Hybrid : 0
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
               Repair : 0
-------------------------------------------------------------------------------
        Fulfillment Id : FIL-EZCALC-STUDENT-SERVED5
      Fulfillment Type : 6 Publisher Activation
          Trust Flags : 0x7 Fully Trusted
               Status : Enabled
        Entitlement Id : ENTL-EZCALC-S1
           Product Id : NAME=EZCALC-S;VERSION=1.0
      Expiration Date : 11-Jun-2014
         Server chain : Server1
      Feature Line(s) :

                    INCREMENT ADD demo 1.0 31-dec-2020 1 TS_OK SIGN="005F 03B6 1B41 C196 \
                    9754 62AB 5F23 5500 3153 E210 952D B005 BAB4 E5C0 B333"
                    INCREMENT SUBTRACT demo 1.0 31-dec-2020 1 TS_OK SIGN="00C8 A83F 8D17 \
                    545A 497E 01EF 341D 1E00 76D0 4EF2 282D 9434 B9A8 F2D4 16C6"
                    INCREMENT MULTIPLY demo 1.0 31-dec-2020 1 TS_OK SIGN="007A 413F 3BD4 \
                    AA6D 4516 9BE1 A271 1800 4305 535F 0F7E 24E4 AB73 83D8 7A00"
                    INCREMENT DIVIDE demo 1.0 31-dec-2020 1 TS_OK SIGN="0075 E4A9 80A5 \
                    522C 0E68 2165 6CE8 CA00 C8B3 4610 2A99 3138 D9DD 4ACA 5E0C"

           Concurrent : 0
          Activatable : 0
               Hybrid : 5
  Concurrent overdraft : 0
 Activatable overdraft : 0
      Hybrid overdraft : 0
               Repair : 0

Viewed 2 of 2 fulfillment records.
```

Performing a Partial Upgrade

A partial upgrade allows you to upgrade some of your licenses. This exercise upgrades two `ezcalc` student licenses to accountant licenses, reducing the number of student licenses from 5 to 3.

Note • *Complete the previous exercise Performing Multiple Activations before performing this one.*

Creating the Request

This request specifies the entitlement `ENTL-EZCALC-AS1`, which defines the accountant licenses to which two student licenses are upgrading.

Task: ***To create the request for a partial upgrade***

Enter the following at the command prompt from the `<platform_dir>` directory:

```
servercomptranutil -new request.xml -activate ENTL-EZCALC-AS1 count=2
```

The following message is returned:

```
Request written to request.xml
```

Generating the Response

This exercise uses the sample response parameter file `W7_EZCalcSvrPartialUpgrade.xml` to generate the appropriate response file for upgrading two of the five `ezcalc` student licenses to accountant licenses.

Task: ***To generate the response***

Enter the following at the command prompt from the `<platform_dir>` directory:

```
.\publisher\responsegen.exe -i request.xml -o response.xml -k .\publisher\xml\DemoKeyout.xml -m
.\publisher\xml\Version6\W7_EZCalcSvrPartialUpgrade.xml
```

Upgrading the Licenses

Process the response file to upgrade the licenses on the license server.

Task: ***To upgrade the licenses***

1. Enter the following at the command prompt from the <platform_dir> directory:

   ```
   servercomptranutil -process response.xml
   ```

   The following information is returned:

   ```
   Response processed successfully. Actions were:
       Create         fulfillment "FIL-EZCALC-ACCT-SERVED2"
       Modify         fulfillment "FIL-EZCALC-STUDENT-SERVED5"
       FLEXnet response dictionary:
               COMMENT=Partial Upgrade - 2 of 5 EZCalculator Student Edition to EZCalculator
               Accountant Edition
   ```

2. To confirm that two student licenses have been upgraded, enter the following command:

   ```
   servercomptranutil -v format=long
   ```

   The output shows that a partial upgrade maintains deduction counts of existing served licenses in server-side trusted storage. For example, the following excerpt shows that, after the upgrade, three student licenses now exist (the original count of 5 has been reduced by 2):

   ```
   Fulfillment Id:  FIL-EZCALC-STUDENT-SERVED5
   Entitlement Id: ENTL-EZCALC-S1
   Product Id: NAME=EZCALC-S;VERSION=1.0
   ...
   Concurrent: 0
   Concurrent overdraft:0
   Hybrid: 3
   Hybrid overdraft: 0
   Activatable: 0
   Activatable overdraft: 0
   Repairs: 0
   ```

   However, this output excerpt shows that two accountant licenses have been added:

   ```
   Fulfillment Id:  FIL-EZCALC-ACCT-SERVED2
   Entitlement Id: ENTL-EZCALC-AS1
   Product Id: NAME=EZCALC-A;VERSION=1.0
   Expiration Date: 11-Jun-2014
   ...
   Concurrent: 0
   Concurrent overdraft:0
   Hybrid overdraft: 0
   Hybrid: 2
   Activatable: 0
   Activatable overdraft: 0
   Repairs: 0
   ```

The five concurrent engineer licenses remain unchanged.

Performing a Partial Return

A partial return allows you to upgrade some of your server-side trusted-storage licenses. This is especially useful if some licenses have already been served to client machines, but you want to return the remaining available licenses.

This exercise returns two `ezcalc` engineer and `ezcalc` two student licenses. Currently, five engineer licenses and three student licenses exist in trusted storage on the license server.

Note • *Complete the previous exercise Performing a Partial Upgrade before performing this one.*

Creating the Request

This request specifies the two fulfillment records that are modified when you return the licenses: `FIL-EZCALC-STUDENT-SERVED5` and `FIL-EZCALC-ENGR-SERVED5`.

Task: ***To create the request for multiple returns***

Enter the following at the command prompt from the `<platform_dir>` directory:

```
servercomptranutil -new request.xml -return FIL-EZCALC-STUDENT-SERVED5 count=2
-return FIL-EZCALC-ENGR-SERVED5 count=2
```

The following message is returned:

```
Request written to request.xml
```

Generating the Response

This exercise uses the sample response parameter file `W7_EZCalcSvrPartialReturn.xml` to generate the appropriate response file for returning the two `ezcalc` student licenses and the two `ezcalc` engineering licenses.

Task: ***To generate the response***

Enter the following at the command prompt from the `<platform_dir>` directory:

```
.\publisher\responsegen.exe -i request.xml -o response.xml -k .\publisher\xml\DemoKeyout.xml -m
.\publisher\xml\Version6\W7_EZCalcSvrPartialReturn.xml
```

Returning the Licenses

Process the response file to return the licenses.

Task: ***To return the licenses***

1. Enter the following at the command prompt from the <platform_dir> directory:

   ```
   servercomptranutil -process response.xml
   ```

   The following information is returned:

   ```
   Response processed successfully. Actions were:
       Modify         fulfillment "FIL-EZCALC-ENGR-SERVED5"
       Modify         fulfillment "FIL-EZCALC-STUDENT-SERVED5"
       FLEXnet response dictionary:
               COMMENT=Partial Return - Return Two Engineer Edition Licenses and Two Student Edition
               Licenses
   ```

2. To confirm that each license type has been decremented by two licenses, enter the following command:

   ```
   servercomptranutil -v format=long
   ```

   The output shows decremented counts for each license type. For example, the following excerpt shows that the original five engineering licenses have been reduced by two:

   ```
   Fulfillment ID: FIL-EZCALC-ENGR-SERVED5
   Entitlement ID: ENTL-EZCALC-NS1
   Product Id: NAME=EZCALC-N;VERSION=1.0
   ...
   Concurrent: 3
   Concurrent overdraft:0
   Hybrid: 0
   Hybrid overdraft: 0
   Activatable: 0
   Activatable overdraft: 0
   Repairs: 0
   ```

   Additionally, the following shows that the three student licenses have been reduced by two:

   ```
   Fulfillment ID: FIL-EZCALC-STUDENT-SERVED5
   Entitlement ID: ENTL-EZCALC-S1
   Product Id:  NAME=EZCALC-S;VERSION=1.0
   ...
   Concurrent: 0
   Concurrent overdraft:0
   Activatable: 0
   Activatable overdraft: 0
   Hybrid: 1
   Hybrid overdraft: 0
   Repairs: 0
   ```

   The two hybrid accountant licenses remain unchanged.

Conclusion

The tutorials in this chapter demonstrate how composite transactions providing flexibility in delivering, upselling, or downselling licenses. Continue to the next chapter, Short-Code Transaction Scenarios, for an introduction to short-code direct transactions.

5

Short-Code Transaction Scenarios

Introduction

The term transaction is to mean a licensing life-cycle event such as activation and return that involves an exchange of information between FlexNet Publisher and FlexNet Operations server resulting in license rights being moved from FlexNet Operations to the trusted storage of a machine running FlexNet Publisher enabled applications.

There are two mechanisms

1. XML based - the information is exchanged using XML documents, ideal for online transactions.
2. Short codes - the information is encoded in simple character strings. These are for use in situations where the size of the XML documents are impractical for the channel to the server, for example if online or e-mail communication cannot be used for security or other reasons.

FlexNet Publisher has always supported short code transactions for the FlexNet Operations retail model, where the rights are transferred from an FlexNet Operations server to local trusted storage (also known as application or client trusted storage). FlexNet Publisher 2016 (11.14.0) extends this to the other FlexNet Operations license models, where the rights are transferred from an FlexNet Operations server to enterprise server trusted storage.

Overview of the short code mechanism

Licenses in trusted storage are held as fulfillments which are XML documents with a typical length of two to three thousand characters. As this is too large to be sent in the short code response the fulfillment is pre-defined in a short code ASR; these are similar to the ASRs used for local activation but contain extra data specific to short codes, the most important being the short code alias. This is an integer that uniquely defines the short code ASR.

Task: ***In a production situation the following tasks are performed***

1. FlexNet Operations creates a Transaction Key which can be downloaded to create a Trusted Configuration XML document.
2. The Trusted Configuration is used by FlexNet Publisher when prepping the product's activation application (embedding the keys needed to encrypt and verify the short codes within it)
3. FlexNet Operations assigns the alias and generates the short code ASR.
4. The ASR is downloaded from the server and deployed with the product.
5. The product's activation utility passes the ASR to the FlexNet Publisher activation API to create a short code request; this contains the short code alias plus other data to identify the machine (UMN).
6. The request is entered at the FlexNet Operations portal together with the Activation Id associated with it. The alias is extracted and used to identify the ASR in use for the transaction, enabling either an OK or Deny response to be generated and displayed.
7. The response is input to the product's activation utility and passed to the FlexNet Publisher API which, if the response is 'OK', extracts the fulfillment from the ASR and writes it to trusted storage.

Server and retail transactions

1. The request can be configured for UMN length. UMNs are 40 characters long, by default eight characters are sent in requests (to keep them short); in large machine populations more than one machine may have the same least significant eight characters so they can't be told apart. The configuration option allows a trade off between request length and risk of duplication.
2. There is an option to use stronger response signing to trade off security against response code length.
3. Request and response codes have additional error check characters to allow entry mistakes to be isolated to a few characters.
4. There is a new streamlined API that must be used by server activation applications and should be used by all new activation applications.
5. The compTranUtil sample activation utility included in the FNP kit now supports short codes; the source file `compTranUtilShortCode.c` demonstrates the new API, the executables **appcomptranutil** and **servercomptranutil** can be used to create requests and process responses.

Server transactions Only

The ASR may be used for multiple transactions to acquire licenses incrementally.

Task: ***Demonstrates short code transactions***

- For windows
- Uses server short codes.
- Uses low response signing strength.

The above mentioned points can be adapted for other platforms (refer to Other platforms section given below), local trusted storage (appcomptrantutil) and other signing strengths (refer to Table 5-1 below).

Task: ***Preparations***

1. Configure the activation application
   a. Edit the FNP kit file examples\activation\compTranUtil\servercomptrantil.xml as follows

   ```
   <trustedconfiguration path="publisher\xml\TrustedId51Output.xml" />
   ```

2. Rebuild the FNP kit.

Task: ***To generate short code response***

Use the **responsegenUI** tool available in the kit (For non-windows platforms use the command line tool **responsegen**)

1. Load the manual response generator tool from the kit - publisher\responsegenUI
2. Goto the "**Generate Short Code Response**" tab
3. Paste in the request code
4. Click "**Validate Request**" and then "**Analyze Request**"

   ```
   The request content is displayed.
   ```

5. Click "**Choose Response Data**" and selected the appropriate file for the transaction type

   ```
   W6_CreateResponseCodeParameterActivation2016.xml
   W6_CreateResponseCodeParameterReturn.xml
   W6_CreateResponseCodeParameterRepair.xml
   ```

6. For incremental activations modify the ID_SUFFIX (in the "**Server XML input Parameter**" edit box) from 1 to 2, 3 and so on.

   ```
   <Name>SCI_ID_SUFFIX</Name>
   ```

   ```
   <Value>2</Value>
   ```

7. Click "**Generate Response**"
8. Copy the response and paste to buffer or "**Save**" it to a file.

Note • ***ResponsegenUI*** *is supported for development and testing only. It loads its XML documents (ASRs and response parameter files) from the "xml" sub folder and its sub folders on startup; if any files are changed, it must be restarted. The ASR must be an identical copy of the one used to generate the request.*

The following use cases described in this chapter demonstrate the use of short-code transactions with request and response short codes:

Case 1: An activation transaction

Task: ***To execute an activation request and response transaction***

1. Open a command window and make the FNP kit binary directory the current directory. Enabling Quick Edit Mode for the window will make copying and pasting requests and responses easier.
2. Create the activation request

```
FNPkit\i86_n3>servercomptranutil -shortcode publisher\xml\ShortCode2016ServerLow.asr -activate
    New request code: 080427-697005-968894-159351-417183-193819-529649-794556
              type: ACTIVATION
```

3. Copy the request into the paste buffer and generate a suitable response (leave ID_SUFFIX at 1).
4. A response can be used:
   a. Only on the machine that generated the request
   b. Only once - whilst the request is pending
5. Process the response, for example if the response was 095939-374182-943367-122555...

```
FNPkit\i86_n3>servercomptranutil -shortcoderesponse 095939-374182-943367-122555
Processing response: 095939-374182-943367-122555
```

   Short code ACTIVATION response processed, fulfillment created

```
Entitlement Id: ENTL-EZCALC-SC2016-SVR-1
Fulfillment Id: FID-EZCALC-SC2016-SVR-1
    Product Id: NAME=EZCALC-A; VERSION=1.0
    Expiration: 24-feb-2016
```

Note • *The "-1" suffix on the Entitlement and Fulfillment Ids allows incremental activations to be performed*

6. The fulfillment can be viewed in full by "`servercomptranutil -viewlong`" (or -vl).

   For any Problems, refer Troubleshooting

Case 2: An incremental activation transaction

Task: ***Procedure for an incremental activation transaction***

1. Create another activation request

```
FNPkit\i86_n3>servercomptranutil -sc publisher\xml\ShortCode2016ServerLow.asr -a
    New request code: 468540-554185-007013-491125-732176-523816-734815-658551
                type: ACTIVATION
```

2. Copy the request into the paste buffer and generate a suitable response with ID_SUFFIX set to "2"

3. Process the response

```
FNPkit\i86_n3>servercomptranutil -scr 022116-103080-554004-058552
Processing response: 022116-103080-554004-058552
Short code ACTIVATION response processed, fulfillment created
      Entitlement Id: ENTL-EZCALC-SC2016-SVR-2
      Fulfillment Id: FID-EZCALC-SC2016-SVR-2
          Product Id: NAME=EZCALC-A;VERSION=1.0
          Expiration: 24-feb-2016
```

Note • *The "-2" suffix on the Entitlement and Fullfillment Ids.*

4. View the fulfillments

```
FNPkit\i86_n3>servercomptranutil -v
Viewing server trusted storage.
U if untrusted
. D if disabled
. . fulfillmentType
. . . Expiry    Count FulfillmentId                  ProductId
. . 6 24-feb-2016 10 FID-EZCALC-SC2016-SVR-1        NAME=EZCALC-A; VERSION=1.0
. . 6 24-feb-2016 10 FID-EZCALC-SC2016-SVR-2        NAME=EZCALC-A; VERSION=1.0
Viewed 2 of 2 fulfillment records.
```

Case 3: A return transaction

Task: ***To generate a short code return transaction***

1. Determine the fulfillment id of the fulfillment to be returned (use "servercomptramnutil -vew" as above) and create the return request

```
FNPkit\i86_n3>servercomptranutil -sc publisher\xml\ShortCode2016ServerLow.asr -return FID-
EZCALC-SC2016-SVR-1
    New request code: 579434-519788-391087-863064-121810-509876-823402-794556
                type: RETURN
      Entitlement Id: ENTL-EZCALC-SC2016-SVR-1
      Fulfillment Id: FID-EZCALC-SC2016-SVR-1
```

```
        Product Id: NAME=EZCALC-A;VERSION=1.0
        Expiration: 24-feb-2016
```

Note • *This disables the fulfillment, preventing its use for licensing.*

```
FNPkit\i86_n3>servercomptranutil -v
Viewing server trusted storage.
U if untrusted
. D if disabled
. . fulfillmentType
. . . Expiry    Count FulfillmentId                  ProductId
. D 6 24-feb-2016 10 FID-EZCALC-SC2016-SVR-1        NAME=EZCALC-A;VERSION=1.0
. . 6 24-feb-2016 10 FID-EZCALC-SC2016-SVR-2        NAME=EZCALC-A;VERSION=1.0
Viewed 2 of 2 fulfillment records.
```

2. Copy the request into the paste buffer and generate a return response.
3. Process the response

```
FNPkit\i86_n3>servercomptranutil -scr 552663-703007-799200-954555
Processing response: 552663-703007-799200-954555
```

4. Short code RETURN response processed, fulfillment returned

```
    Entitlement Id: ENTL-EZCALC-SC2016-SVR-1
    Fulfillment Id: FID-EZCALC-SC2016-SVR-1
        Product Id: NAME=EZCALC-A; VERSION=1.0
        Expiration: 24-feb-2016
```

Note • *that the fulfillment has been deleted.*

```
FNPkit\i86_n3>servercomptranutil -v
Viewing server trusted storage.
U if untrusted
. D if disabled
. . fulfillmentType
. . . Expiry    Count FulfillmentId                  ProductId
. . 6 24-feb-2016 10 FID-EZCALC-SC2016-SVR-2        NAME=EZCALC-A;VERSION=1.0
Viewed 1 of 1 fulfillment records.
```

Case 4: A repair transaction

Repairs are very similar to returns. Repair transactions are used to restore trust to fulfillments after a change to the machine environment has caused them to lose it.

If wish to try repairs you must use a different machine (or, if you are running in a VM, a different VM). Copy the trusted storage file demo_80000100_tsf.data (or demo_00000100_tsf.data for local trusted storage) to the other machine - note that this is a hidden file. For Windows this is in "`All Users\Application Data\FLEXnet`" folder.

Task: ***To generate a short code repair transaction***

1. On the new machine, check that the fulfillment is untrusted

```
FNPkit\i86_n3>servercomptranutil -v
Viewing server trusted storage.
U if untrusted
. D if disabled
. . fulfillmentType
. . . Expiry    Count FulfillmentId   ProductId
U . . 24-feb-2016 10 FID-EZCALC-SC2016-SVR-2   NAME=EZCALC-A;VERSION=1.0
Viewed 1 of 1 fulfillment records.
```

2. Create a repair request

```
FNPkit\i86_n3>servercomptranutil -sc publisher\xml\ShortCode2016ServerLow.asr -repair FID-
EZCALC-SC2016-SVR-2
    New request code: 007258-115883-850506-278816-314768-738129-496693-494568-506559
                type: REPAIR
      Entitlement Id: ENTL-EZCALC-SC2016-SVR-2
      Fulfillment Id: FID-EZCALC-SC2016-SVR-2
          Product Id: NAME=EZCALC-A;VERSION=1.0
          Expiration: 26-feb-2016
```

3. Copy the request into the paste buffer and generate a repair response.

4. Process the response

```
FNPkit\i86_n3>servercomptranutil -scr 077650-932690-120405-570550
Processing response: 077650-932690-120405-570550
Short code REPAIR response processed, fulfillment repaired
      Entitlement Id: ENTL-EZCALC-SC2016-SVR-2
      Fulfillment Id: FID-EZCALC-SC2016-SVR-2
          Product Id: NAME=EZCALC-A; VERSION=1.0
          Expiration: 26-feb-2016
Note that the fulfillment is now trusted
FNPkit\i86_n3>servercomptranutil -v
Viewing server trusted storage.
U if untrusted
. D if disabled
. . fulfillmentType
. . . Expiry    Count FulfillmentId               ProductId
. . 6 26-feb-2016 10 FID-EZCALC-SC2016-SVR-2      NAME=EZCALC-A;VERSION=1.0
Viewed 1 of 1 fulfillment records.
```

Managing pending requests.

When a request is created it is written to trusted storage together with the short code ASR and remains there pending the availability of the response. Once response processing is complete it is deleted.

1. Only one request for each short code ASR may be pending; to view it:

```
FNPkit\i86_n3>servercomptranutil -sc publisher\xml\ShortCode2016ServerLow.asr
Pending request code: 565101-354989-327644-564937-971611-739947-933867-106555
                type: ACTIVATION
```

2. Best practice is to use the pending request to generate the response and complete the transaction. However, if a response of a different type is needed, the pending request must be cancelled first.

```
FNPkit\i86_n3>servercomptranutil –sc publisher\xml\ShortCode2016ServerLow.asr -
cancelshortcoderequest
Pending request code: 565101-354989-327644-564937-971611-739947-933867-106555
                type: ACTIVATION
Cancel the pending request code? Enter "yes" or "no": yes
Request code cancelled.
```

To avoid the confirmation step you can add the "-yes" option to the command line.

Other things to try

Mistype one or more characters of a response when processing it.

Use the wrong ASR, for example. a non-server ASR with **servercomptranutil** or a local ASR

Customizing the tutorial for alphanumeric codes, UMN length and different signing strengths

Using different response signing strengths

This tutorial uses low signing strengths because, the security is not an issue and it keeps the responses short. To use medium or high use the alternate files given in the table.

Table 5-1 • File table with Signing strengths

	Low	Medium	High
Trusted Configuration	TrustedId51Output.xmlT	TrustedId55Output.xml	TrustedId56Output.xml
Short code ASR (server)	ShortCode2016ServerLow.asr	ShortCode2016ServerMedium.asr	ShortCode2016ServerHigh.asr
Short code ASR (non-server)	ShortCode2016Low.asr	ShortCode2016Medium.asr	ShortCode2016High.asr
responsegen keyfile	TrustedId51Keyfile.xml	TrustedId55Keyfile.xml	TrustedId56Keyfile.xml

Alphanumeric codes option

For shorter but more difficult to enter codes - example HCPAR3-WTFDUA-NE8NJD-QK0546-Y14UK7-DTFFZ3, When using FlexNet Operations this is specified when creating the short code ASR.

Task: ***To use FlexNet Publisher only execute the following steps***

1. Edit the short code ASR (for example: `ShortCode2016ServerLow.asr`) and change the CodeBase from DEC to ALNUM32. Element in both the `<RequestOverrides>` and `<ResponseOverrides>` directories elements.
2. Resign the ASR

   ```
   FNPkit\i86_n3>publisher\xmlsign -a activationsettings\publisher.xml -i
   publisher\xml\ShortCode2016ServerLow.asr -o publisher\xml\ShortCode2016ServerLow.asr
   ```

3. Reload responsegenUI

To include different length UMNs in the request

Again, when using FlexNet Operations this is specified when creating the short code ASR. It can be tried using FlexNet publisher only using the steps above but with the following change to the ASR XML

1. Change the `<Size>8</Size> to (say) <Size>16</Size>` (a full UMN is 40) in the `<Name>SCI_UMNE</Name>` and/or `<Name>SCI_UMN2</Name>` elements of the `<RequestOverrides>` and `<ResponseOverrides>` element.

Other platforms

When preparing the kit, edit and use the appropriate **servercomptranutil** prep XML file (`serverCompTranUtil_unix.xml` or `serverCompTranUtil_mac.xml`)

To generate responses, use the command line tool **responsegen**, specifying the correct keyfile from the File table with Signing strengths.

If you wish to try multiple activations from the same ASR you will need to make copies of the response parameter file so that you can use different suffixes.

In the `publisher\xml\version1` folder

1. Copy `W6_CreateResponseCodeParameterActivation2016.xml` to `W6_CreateResponseCodeParameterActivation2016-2.xml`
2. Edit `W6_CreateResponseCodeParameterActivation2016-2.xml` and change `<Name>SCI_ID_SUFFIX</Name><Value>1</Value>` to `<Name>SCI_ID_SUFFIX</Name><Value>2</Value>`.
3. If you wish to try more incremental activations, you will need to create similar additional files `W6_CreateResponseCodeParameterActivation2016-3.xml` and so on

Troubleshooting

The most common cause of errors during development is a mismatch between the short code ASR and the trusted configuration specified when the activation application is prepped. Errors can vary depending of the state of trusted storage but most likely is

```
FNPkit\i86_n3>servercomptranutil -shortcode publisher\xml\ShortCode2016ServerLow.asr -activate
ERROR: API function failed (50058, 61106, and 17005)
The trusted section needed by the short code ASR does not exist.
```

However, if the trusted section does exist but was created with a different configuration (with different encryption and signing keys) then errors can also occur when validating the request or processing the response.

Ensure that the steps in the Preparations section have been followed correctly (or if you are customizing the process, then check files are from the same column of the table). If this does not fix it then reset the trusted storage files ("tsreset_svr -reset") and try again.

responsegenUI

If the short code alias is not recognized check in the "**ASR Data**" tab that the short code ASR you are using is listed; these files are loaded from the xml subdirectory so the publisher directory must be the current directory when the tool is started.

If request validation or response generation fails check in the "**Key Data**" tab for the TrustedId5*Keyfile.xml key files; these are also loaded from the xml subdirectory.

responsegen

```
Error: Generate response error
Error code: 16
```

Most likely wrong key file - low/medium/high signing strengths have different key files. See the File Table for the correct ASR/keyfile pairings

Error tuple - (major, minor, system)

Short code API functions (flxActShortCodeXXX) usually set major error 50058 (LM_TS_SHORTCODE) and the system error (1nnnn) identifies the cause - see header machind\actvation\include\FlxActSystemError.h

For other major errors (5nnnn) see `FlxActError.h`

The minor error identifies a code point (not publically defined).

The short code activation API

This is defined and commented in header `FlxActShortCode.h` and reproduced here in full.

The compTranUtil sample activation utility included with the FNP kit demonstrates use of this API - also see file `compTranUtilShortCodes.c`

```
/
*****************************************************************************************
***
* NOTICE OF COPYRIGHT AND OWNERSHIP OF SOFTWARE:
*
```

```
* Copyright (c) 2016 Flexera Software LLC. All Rights Reserved.
*
*****************************************************************************************
***/
/*
 * FlxActShortCode include file
 *
 * API functions for Short Code Transactions
 * These are common to both the application and the server activation applications.
 */
#ifndef FlxActShortCode_h
#define FlxActShortCode_h 1
/*  Usage.
    Pre-requisite - an API handle obtained from flxActCommonOpen
    For request code creation and management:
    1. Obtain a short code context handle using one of
            flxActShortCodeCreate               ASR is read from a file
            flxActShortCodeCreateFromBuffer      ASR is in memory
    2a. To create a request code use one of
            flxActShortCodeGetActivationRequest
            flxActShortCodeGetReturnRequest
            flxActShortCodeGetRepairRequest
    2b. To obtain a pending request code
            flxActShortCodeGetPendingRequest
      A request is pending from the time it is created until a valid response is processed or it
is cancelled.
       Only one request may be pending for a given ASR; to create a different one, cancel the
pending one.
    2c. To cancel a pending request
         flxActShortCodeCancelRequest
Note that cancelling a return request re-enables the fulfillment making it usable again.
You need to trust that the user has not completed the return on the server, otherwise it's an
exploit.
The alternative is to delete the fulfillment with the request (user can re-install if necessary)
        see function flxActCommonLicSpcProductDelete
    3. Destroy the short code context
        flxActShortCodeDestroy

    When the response is available, process it using
        flxActShortCodeProcessResponse
No context is needed as the ASR is cached in trusted storage when the request is created and it
can be identified from the content of the response code. These functions can be called to get
more information about the response processing.
        flxActShortCodeGetDenyReason    For Deny response types
        flxActShortCodeGetWarning        On success - see flxShortCodeWarning below
        flxActShortCodeGetErrorDetail    On failure -
    Vendor data can be included in requests and responses - see  flxActShortCodeSetVendorData()
below.
*/


#include "FlxActCommon.h"
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Context for short code operations.
 */
```

```
typedef struct flxActShortCode*    FlxActShortCode;
/*
 * Create and destroy a short code context.
 */

FlxActBool flxActShortCodeCreate(
        FlxActHandle        handle,            /* As returned by flxActCommonHandleOpen    */
        const char*            pszAsrFile,        /* Path of short code ASR XML               */
        FlxActShortCode*    phShortCode);    /* For return of the short code handle        */
FlxActBool flxActShortCodeCreateFromBuffer(
        FlxActHandle        handle,            /* As returned by flxActCommonHandleOpen    */
        const char*            pszAsrBuffer,    /* Buffer containing short code ASR XML        */
        FlxActShortCode*    phShortCode);    /* For return of the short code handle        */
void flxActShortCodeDestroy(FlxActShortCode hShortCode);
FlxActBool flxActShortCodeGetActivationRequest(
        FlxActShortCode        hShortCode,
        const char**        ppszRequest);        /* For return of a pointer to the short code
request */
FlxActBool flxActShortCodeGetReturnRequest(
        FlxActShortCode        hShortCode,
      const char*         pszFulfillmentId,   /* The fulfillment to be returned
*/
        const char**        ppszRequest);        /* For return of a pointer to the short code
request    */
FlxActBool flxActShortCodeGetRepairRequest(
        FlxActShortCode        hShortCode,
      const char*         pszFulfillmentId,   /* The fulfillment to be returned
*/
        const char**        ppszRequest);        /* For return of a pointer to the short code
request */
FlxActBool flxActShortCodeGetPendingRequest(
        FlxActShortCode        hShortCode,
       const char**       ppszRequest,    /* For return of a pointer to the short code request */
        FlxShortCodeType*    pRequestType); /* For return of the request type */
FlxActBool flxActShortCodeGetPendingRequestProdSpc(
        FlxActShortCode        hShortCode,
        FlxActLicSpc        licSpc);        /* For RETURN and REPAIR, FlxActProdSpc of target
fulfillment is added */
FlxActBool flxActShortCodeCancelRequest(FlxActShortCode hShortCode);
/*
 * Process a short code response.
 * On success, if pResponseType is flxShortCodeTypeError then
 * call flxActShortCodeGetDenyReason() to get the deny reason given by the server,
 * Otherwise call flxActCommonRespProdLicSpcGet() to get details of the fulfillment just
 *    activated, returned or repaired.
 *    For activations, optionally call flxActShortCodeGetWarning to find out whether
 *    an existing fulfillment was overwritten.
 *    On failure, call flxActCommonHandleGetError() to get the error tuple.
 *    See also flxActShortCodeGetErrorDetail() below.
 */
FlxActBool flxActShortCodeProcessResponse(
        FlxActHandle        handle,            /* As returned by flxActCommonHandleOpen    */
        const char*            pszResponse,    /* The response code from the server        */
        FlxShortCodeType*    pResponseType);    /* For return of the response type            */

/*
 * Get the deny reason.
 * If the request type returned by flxActShortCodeProcessResponse() is flxShortCodeTypeError
 * then this function will return the reason given by the server for denying the request.
```

```
 * If the response was of a different type or the server did not set a reason, it will return
zero.
 * Note that the reason values are defined by the server, not FNP.
 */
uint32_t flxActShortCodeGetDenyReason(FlxActHandle handle);
/*
 * Response processing warnings - set when processing succeeds but there is something to note
 */

typedef enum
{
    flxShortCodeWarningNone                        = 0,
    flxShortCodeWarningFulfillmentOverwritten    = 1,
    /* Returned by flxActShortCodeGetWarning if handle is NULL */
    flxShortCodeWarningInvalidHandle            = -1
} flxShortCodeWarning;

flxShortCodeWarning flxActShortCodeGetWarning(FlxActHandle handle);
/*
 *    Additional error details.
 *
 * For system errors LM_TSSE_SHORTCODE_INVALID_CHAR and LM_TSSE_SHORTCODE_INVALID_CHECK the
details
 * contain a copy of the short code with potentially invalid characters replaced by a '?'
 */
const char* flxActShortCodeGetErrorDetail(FlxActHandle handle);

/*
 * Vendor Data
 *
 * Any vendor data items must be pre-defined in the ASR - name, type and length.
 * Request item values are set by calling flxActShortCodeSetVendorData() before creating the
request.
 * Response items are written to the fulfillment record and can be queried using
 * flxActCommonProdLicSpcVendorDataGetCount() and flxActCommonProdLicSpcVendorDataGetByIndex()
 */
FlxActBool flxActShortCodeSetVendorData(
        FlxActShortCode    hShortCode,
      const char*      pszItemName,      /* The name as defined in the short code ASR
*/
        const char*        pszItemValue);        /* A value valid for the type specified in the
ASR,
                                                e.g. "123" for DEC, "F12A" for HEX, "MyValue" for
ASCII96    */
#ifdef __cplusplus
}
#endif
#endif
```

6

FlexNet Publisher Implementation Methodology

Having seen a preview of FlexNet Publisher in the previous chapters via the EZCalculator example, you can now preview implementing this technology into your own products. This chapter is oriented toward developers in your organization.

Methodology Roles and Goals

Implementing FlexNet Publisher into your product requires participation from many areas of your company, including Product Management, Engineering, Release Engineering, QA, and IT. Each role can be carried out by an individual or a group of individuals; additionally, an individual or group can perform one or more roles. This chapter focuses on the following publisher roles:

Table 6-1 • FlexNet Publisher Roles

Role	Goals for Licensing	Role in this Tutorial
Product Manager	Ensures revenue growth and market penetration while maintaining customer satisfaction; specifies product features.	Specifies product features and orderables.
Application Developer	Writes software based on functional specification; inserts licensing code without having to think about the business models for selling the product.	Instruments application with FlexNet Licensing calls; unit tests licensing behavior.
QA Engineer	Performs product testing; tests as few binary variations as possible for a given product.	Tests licensing behavior using sample licenses.
Release Manager	Prepares engineering and QA-delivered bits for a product release; ensures appropriate licensing models are configured and packaged.	Packages application for distribution and installation.

While the following roles are simulated in this tutorial, you should expect participation of these roles in a real-world implementation of FlexNet Publisher.

Table 6-2 • FlexNet Publisher and End-User Roles Participation

Publisher/ End-User	Role	Description
Publisher	IT Manager	Deploys licensed software, ensuring minimum interruption so end users can gain access to licensed software, offline or online.
Publisher	Release Manager	Prepares engineering and QA-delivered bits for a product release; ensures appropriate licensing models are configured and packaged.
End-user Enterprise	License Manager	Manages and administers licensed applications, ensuring availability of licenses for mission-critical applications.
End-user Enterprise	End User	Installs and uses software whenever and wherever required.

Figure 6-1depicts the FlexNet Publisher implementation activities. These activities are grouped into four phases. This chapter addresses the tasks in the blue section, for which the software publisher is responsible. The EZCalculator sample application is used in the diagram throughout to illustrate the method.

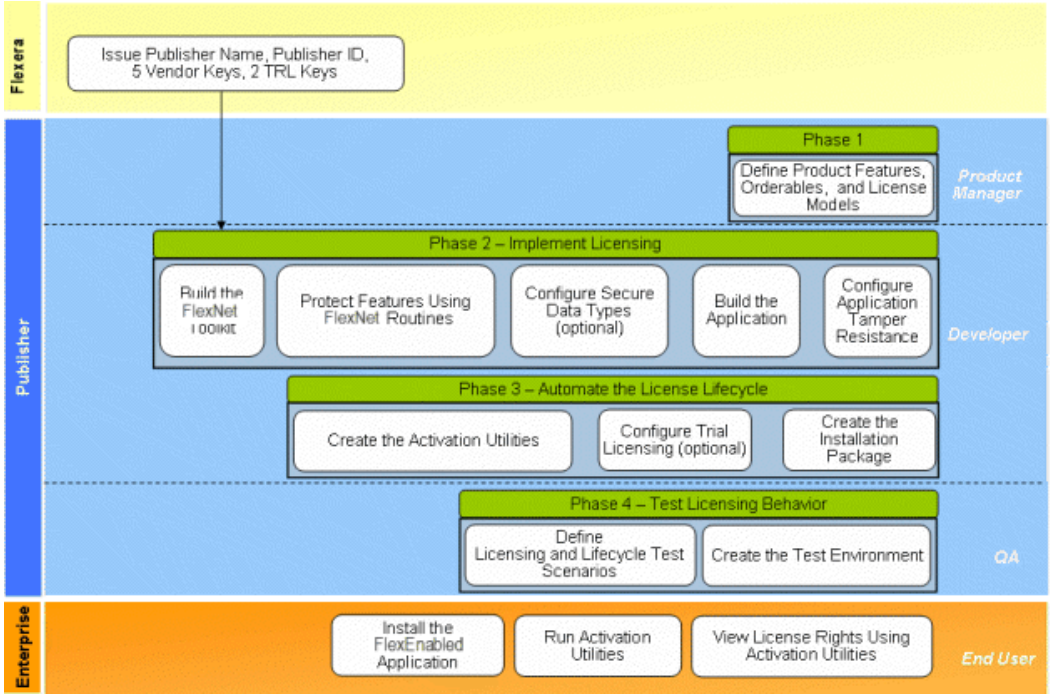

Figure 6-1: FlexNet Publisher Implementation Methodology

The methodology from the publisher perspective includes the following phases:

- Phase One: Defining Product Features, Orderables, and License Models

- Phase Two: Implementing Licensing
- Phase Three: Automating the License Lifecycle
- Phase Four: Testing Licensing Behavior

Phase One: Defining Product Features, Orderables, and License Models

Before implementing licensing into your product, features, orderables, and license models must be defined. The product manager typically performs this step.

Overview

A feature is a software capability that is licensable and is implemented by software developers in product code. A product contains one or more features. The meaning of a feature depends entirely on how it is used by a product developer or manager. For example, a feature could represent any of the following:

- A product consisting of one or more components
- A specific version of a product
- A part of a product
- A piece of data

An orderable is an individual unit that can be purchased by the end-user consumer or enterprise and is identified by one or more stock keeping units (SKU). It is the job of the product manager to define the set of orderables for a given application given their understanding of what capabilities customers are willing to buy. It is useful to think of an orderable in such terms as editions: student, engineer, accountant; or levels: gold, silver, or platinum edition of your product. Additionally, a group of individual orderables can be combined into a suite, which in itself is an orderable.

A license model defines the business terms and conditions for using an orderable. For example, a product may use locked licensing to define that an end user is restricted to using the product on a given machine. Refer to *Programming Reference for Trusted Storage-Based Licensing* for more information on available license models.

EZCalculator Orderables

As an example, Table 6-3 lists the EZCalculator orderables as defined by the product manager.

Table 6-3 • EZCalculator Orderables and Features

Product Name/ Part Number	Features	Target Customer	Entitlement ID	Product ID
Student edition/ EZ-123-STU	Add, Subtract, Multiply, Divide	Students	`ENTL-EZCALC-S1`	`NAME=EZCALC-S` `VERSION=1.0`

Table 6-3 • EZCalculator Orderables and Features

Product Name/ Part Number	Features	Target Customer	Entitlement ID	Product ID
Engineer edition/ EZ-123-ENG	Add, Subtract, Multiply, Divide, Power	Engineers	`ENTL-EZCALC-NS1`	`NAME=EZCALC-N` `VERSION=1.0`
Accountant edition/ EZ-123-ACC	Add, Subtract, Multiply, Divide, Average	Accountants	`ENTL-EZCALC-AS1`	`NAME=EZCALC-A` `VERSION=1.0`
Enterprise edition/ EZ-123-SUITE	Add, Subtract, Multiply, Divide, Power, Average This product is an upgrade for companies that have previously purchased the Engineer or Accountant edition.	Enterprise	`ENTL-EZCALC-ES1`	`NAME=EZCALC-E` `VERSION=1.0`

EZCalculator contains these features:

- Add
- Subtract
- Multiply
- Divide
- Exponent
- Average

The functionality for all features is contained in a single executable. The source for EZCalculator is included in the Licensing toolkit distribution in the `<install_dir>/examples/ezcalc` folder.

EZCalculator License Models

Table 6-4 lists the license models that align with the end-user requests for the EZCalculator product.

Table 6-4 • EZCalculator License Policies

Orderable Name	License Models Allowed
Student edition	• 1-day trial license • Permanent locked license
Engineer edition	• 15-day trial license • Concurrent license

Table 6-4 • EZCalculator License Policies

Orderable Name	License Models Allowed
Accountant edition	• 30-day trial license • Permanent locked license. Licenses must be purchased as part of a volume licensing agreement in multiples of 10 seats.
Enterprise edition	• Concurrent licenses

Phase Two: Implementing Licensing

Building the Licensing Toolkit

Building your publisher-specific production Licensing toolkit produces the following deliverables.

- Your publisher-specific FlexNet client library
- Your publisher-specific vendor daemon
- Your publisher-specific license generator
- Licensing utilities

Before you begin, make sure that you have obtained the following information from Flexera Software:

- **Vendor name**—The example for this tutorial uses `demo` for the vendor name.
- **Vendor keys**—A sample set of keys is located in `machind/lm_code.h`.
- **TRL keys**—A sample set of keys is located in `machind/lm_code.h`.
- **The signed publisher.xml file that contains your Publisher ID**—A sample file resides in `<platform_dir>/ActivationSettings/publisher.xml` for the `demo` vendor.

Protecting Product Features with Licensing Routines

You instrument your application with the Flexible API, just as you have pre-version 11.0, to protect your product features. There are an additional two routines you need to include:

- **lc_flexinit**—This initializes the FlexEnabled application to use trusted storage.
- **lc_flexinit_cleanup**—This releases FlexNet Publisher resources associated with trusted storage.

Consult the *C/C++ Function Reference* for complete details. Additionally, the EZcalculator source shows the usage of these two routines. It is located in the Licensing toolkit in the following location: `<install_dir>/examples/ezcalc/ezcalc.c`.

Configuring Secure Data Types

Secure Data Types (SDTs), which can be optionally used, allow you to create greater protection against hackers who attempt to tamper with your licensing implementation by disabling licensing calls from your application code. The main idea is to create new SDTs that are bound to particular features, and then assign some critical data values to variables of the SDTs. If licensing is disabled by a hacker, SDTs in your application contain inappropriate values, and thus cripple proper functioning of your application.

If you choose to implement SDTs into your source code, define them using macros supplied with the Licensing toolkit. See *Programming Reference for Trusted Storage-Based Licensing* for complete details.

Building Your FlexEnabled Application

Once you instrument your application with licensing calls and optionally, SDTs, you are ready to build your application. If you did not use SDTs at all, the procedure for building your application is the same as in previous versions of FlexNet Publisher. See *Development Environment Guide* for details. The next step for the majority of platforms is to modify the FlexEnabled application and create an application specific activation library component. This process securely links the FlexEnabled application and the activation library component.

On a small number of platforms this enhanced security and SDTs are not available. These platforms are identified in the Release Notes and do not require further processing of the FlexEnabled application as described in the next section.

Processing Your FlexEnabled Application

Once your application is built, you modify it and create the activation library component to use with it:

- Configure the preptool configuration XML document. A sample is provided in `examples\ezcalc\ezcalc.xml`.
- Process your executable along with the preptool configuration XML document with Flexera Software's post-processor utility, `preptool`. The *Programming Reference for Trusted Storage-Based Licensing* contains more information about this utility.

Phase Three: Automating the License Lifecycle

In the previous section, you were introduced to the steps needed to implement FlexNet Publisher into your application. This ensures your licensing terms and conditions are enforced by FlexNet Publisher at runtime.

This section introduces you to automating the license lifecycle management, including:

- Providing transparent trial licensing
- Automating the initial delivery of license rights to your end user
- Providing your end user with an automated method for rehosting and returning license rights

You need to design your license lifecycle such that your end user's experience is seamless and transparent, minimizing the number of direct user inputs required for an activation.

About Activation Utilities

Central to the license lifecycle is the activation utility, which is a crucial piece of business logic you create to automate the license lifecycle for your end user. This utility can be stand-alone or incorporated into your FlexEnabled application. The activation utility takes inputs, either from the end user directly or via other external means, and silently contacts the publisher to activate or modify license rights, thus ensuring a transparent end-user experience.

Your activation utility manages all phases of the license lifecycle:

- Initial activations, including trials and paid licenses
- Returns
- Rehosts
- Upgrades
- Renewals
- Repairs

An activation utility can reside locally to the FlexEnabled application, the application activation utility, or it can reside locally to the FlexNet license server, the server activation utility.

- **Application activation utility**—This utility is needed for activating license rights for products sold primarily to a consumer environment using a locked license model. It is also required if the FlexEnabled client machine has licenses activated from a license server. The utility is used directly by the end user on the application's machine.
- **Server activation utility**—This utility is needed for activating license rights for products sold to enterprises using a concurrent license model. The utility is used by the license administrator at the enterprise on the license server machine.

If you sell to multi-user environments, but also want end users to obtain a detached license for off-network use from a license server, you need to create both an application activation utility and a server activation utility.

Additionally, for activations from the publisher to the license server or a FlexEnabled client, best practice is to use composite transactions. For activations between license servers or from a license server to FlexEnabled clients, only single-action transactions are supported. Therefore, depending on the end user's licensing model, you might need to provide two sets of server and application activation utilities--one for composite transactions and one for single-action transactions. For more information on about composite and single-action transactions and creating the activation utilities that support them, see the *Programming Reference for Trusted Storage-Based Licensing*.

Creating the Activation Utilities

Use the Activation API to create your activation utilities for use in virtual and non-virtual environments. This API offers these categories of operations for managing and automating the license lifecycle using composite transactions:

- Activation of a license
- Return of a license to its source
- Repairing of trusted storage

More complex license lifecycle processes can be composed from these operations. For example, an upgrade of a license can be accomplished by first returning the existing license rights, followed by an activation of the new license rights.

Sample server and application activation utilities for composite and single-action transactions are included in the FlexNet Publisher toolkit distribution in the `<install_dir>/<platform_dir>` directory. The code for these utilities is in the `<install_dir>/examples/activation` directory. For instructions on creating your own server or client activation utility (that supports composite or single-action transactions), see the *Programming Reference for Trusted Storage-Based Licensing*.

Activation Utilities for EZCalculator

The product manager wants to sell EZCalculator to students as well as professional users, in single-user as well as multiuser environments, and to enable a wide range of license lifecycle processes. As a result, both an application activation utility and a server activation utility are needed--one for the FlexEnabled-client machine and one for the license-server machine.

Table 6-5 depicts the roles that these two activation utilities play for automating the EZCalculator license lifecycle processes.

Table 6-5 • Activation Utilities

Activation Utility Location	Used by...	To Automate...
Application machine	Student	• Activating of a license from the publisher • Returning of a license to the publisher
	Engineer Accountant	• Activating of a license from the enterprise license server • Returning of a license to the enterprise license server
License Server machine	IT Deployment Manager	• Charging up the license server with licenses • Returning licenses from the license server to the publisher • Transferring licenses from one license server to another • Upgrading Engineer edition to Enterprise edition

The Application Activation Utility

When the application activation utility contacts the publisher for license rights, it can prompt the end user for a serial number or other similar input, and behind the scenes, works to get the license activated. When it contacts the license server, it simply prompts the user to specify the product for which they want license rights: Accountant, Engineer, or Enterprise edition.

The Server Activation Utility

The server activation utility can prompt the license manager for the Entitlement ID and optionally, for a count, expiration and product identifier (like a Product ID or SKU). It then obtains a license from the publisher and installs it on the license server, effectively, charging up the license server.

Configuring Trial Licensing

The next step in automating your license lifecycle involves creating activation specification records (ASRs). These provide all the information necessary for trial licensing that uses *local activation*: that is, license rights activations without the need for transactions with a FlexNet license server or publisher server.

An ASR is a signed XML document that contains all the information for the activation utility to install licenses on a license server or on the machine where the FlexEnabled application resides. These specifications are packaged with your product distribution. An ASR contains information on:

- The features to enable
- Entitlement information for these features such as count, expiration, and so on
- Repeat trial rules
- A TrustedID payload to protect against piracy

See `<install_dir>/examples/ezcalc` for sample ASRs.

Creating the Installation Package

Once your activation utility or utilities and ASRs are complete, you package them along with your FlexEnabled executable into your installer for deployment at your end-user enterprise. *Development Environment Guide* gives complete details on what you need to include.

Phase Four: Testing Licensing Behavior

In this phase, you test both the licensing behavior and lifecycle processes, as they relate to a FlexNet license server, that make up the end-user experience. For the EZCalculator product, all of the necessary components are provided, pre-built, in the Licensing toolkit distribution. Although they are presented as distinct components in this demo, you can embed some or all into your FlexEnabled application, your installer, or a combination of both, to provide a seamless activation experience for your end user.

Defining Licensing and Lifecycle Test Scenarios

Given the product and feature structure for EZCalculator, the following are some suggested test scenarios that your QA department may wish to perform.

- Getting a 1-day, 15-day or 30-day trial license for Student, Engineer, or Accountant editions, respectively.
- Charging up a license server to serve concurrent licenses for the Engineer edition
- Charging up a license server to serve activatable licenses for the Accountant edition
- Checking out a concurrent license for the Engineer edition from a FlexEnabled client
- Activating a local license for the Accountant edition from a FlexEnabled client
- Upselling licenses from an Engineer edition to an Enterprise edition on the license-server machine and then checking out a concurrent license from a client machine for the Enterprise edition.
- Returning a local license for the Accountant edition from the client machine to the license server.

Creating the Testing Environment

After you have defined your licensing and lifecycle test scenarios, create a test environment in which you can run these scenarios. This may include dedicating one or more machines to simulate the end-user enterprise environment. (These machines can be physical or virtual.) In addition to the test scenarios, you should test these basic operations that take place in the end-user environment.

- **Install the application**—On a dedicated system, install your application as your end user would. Try to emulate the your end user's environment as much as possible so that you detect any internal dependencies that may inadvertently exist in your environment during the installation process.
- **Run the activation utilities**—Perform typical end-user activities as you have implemented with your activation utilities. Take into consideration your test scenarios.

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
composite transactions	Available in FlexNet Publisher 11.8, composite transactions allow multiple actions to be combined into a single transaction. Also called multi-action or V2 transactions.
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.

Term	Meaning
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.
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.
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.
license server system	See license server.

Term	Meaning
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. Also 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.
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.

Term	Meaning
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.
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.

Term	Meaning
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.
simple entitlement	An entitlement for a single, specific customer.
single-action transaction	This is the default transaction type for FlexNet Publisher versions 11.7 and earlier. Each transaction can include only a single action. Also known as legacy or V1 transactions.
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.

Term	Meaning
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.
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.

Term	Meaning
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

A

activatable license 83
activated feature line, See feature definition line
activation 83
    short code 92
activation API 83
activation application 84
activation client 84
activation request 84
activation response 84
activation server 84
activation specification record 83
activation transaction 84
anchor 84
audience 9

B

binding 84
borrowing 85
bulk entitlement 85

C

checkin 85
checkout 85
client system 85
concurrent license 85
config response 85
consumer 85
counted license 85

D

daemon, vendor 93
debug log file 86
demo license, See trial license
duplicate grouping 86

E

embedded activation, See local activation
encryption seed 86
end user 86
end-user license administrator, See license administrator
end-user machine, See client system
enterprise 86
entitlement 86, 89
entitlement ID 86
expiring license 86

F

feature 87
feature bundle 87
feature definition line 83, 87
FEATURE/INCREMENT line, See feature definition line
file
    license 88
    options 90
    report log 92
file-based activation, See manual activation
FlexEnabled application 87
FlexEnabled client 87
Flexible API 87
FlexNet Agent 87
FlexNet Licensing for Java Programmers Guide 10

FlexNet Operations 87
floating license 88
fulfill 88
fulfillment record 88

H

heartbeats 88
hop count 88
hostid 88
hostid type 88
hybrid license 88

I

intended audience 9
internet activation, See programmatic activation

K

keyboard activation, See short-code activation

L

LGAPI 89
LGT 89
license
    counted 85
    served 92
    unserved 93
License Administration Guide 10
license administrator 88
license file 88
License Generation Toolkit 89
license generator 88
License Generator API 89
license group 89
license model 89
license rights 89
    local 90
license search path 89
license server 89
license server manager 90
license server manager, definition 90
license server system 89
license sharing 90
license-file list, See license search path
licensing client 90
lmadmin 90
lmgrd 90
local activation 90
local license rights 90
log file
    debug 86
    report 92

M

machineid 90
manual activation 90
model
    license 89

N

newlink GlossaryC 85

O

options file 90

P

package 91
part number 91
permanent license 91
product 91
programmatic activation 91
publisher id 91

Q

quorum 91

R

regular entitlement, See simple entitlement
rehost 91
repair request 91
repair response 91
report log file 92
return request 92
return response 92
return transaction 92

S

served license rights 92
server
    license 89
short code 92
short-code activation 92
signature 92
simple entitlement 92
suite 92
supersede 92

T

three-server redundancy 93
triad 93
trial license 93
trusted configuration file 93
trusted storage 93
trusted storage configuration 93
type, hostid 88

U

uncounted license
    uncounted 93
unserved license rights 93
unserved licenses 93
upgrade 93
upsell 93

V

Vendor Certificate Generator 93
vendor daemon 93
vendor name 94

W

windback 94
windback detection 94
windback tolerance 94