Contents

FlexNet Embedded Local License Server Quick Start Reference ......................... 11
Installing and Running the License Server ..................................................... 12
Performing Other Service Maintenance Tasks .................................................. 13
License Server Administrator Command-line Tool: Command Summary .................. 14

1 Introduction ........................................................................................................ 25
Local License Servers vs. License Servers Hosted in the Cloud ......................... 25
What's in this Guide ............................................................................................... 26
Product Support Resources .................................................................................. 27
Contact Us ........................................................................................................... 28

2 Getting Started ................................................................................................... 29
About Getting Started with a CLS Instance ......................................................... 29
Requirements for the Local License Server ......................................................... 29
  Hardware Requirements ...................................................................................... 30
  Validated Platforms ........................................................................................... 30
  Virtual Machine Support .................................................................................. 30
  Supported Browsers .......................................................................................... 31
  Java Prerequisites ............................................................................................. 31
  License Server Manager Requirements ......................................................... 31
Overview: Administrator Experience on the Local License Server .................. 32
Configuring, Installing, and Starting the Local License Server .......................... 33
  Configure, Install, and Start on Windows ......................................................... 34
  Configure, Install, and Start on Linux .............................................................. 39
    Files Required for Installation Using systemd ............................................... 39
    Task Overview ............................................................................................... 39
    Verify systemd-Enablement on Linux System ............................................. 40
    Run the Install Script .................................................................................... 40
Contents

About the Install Script ................................................................. 40
Install and Start the Service with the Default Configuration .................. 41
Install and Start the Service with a Modified Configuration .................... 41
Configuration Values Editable from the Command Line .......................... 41
Manage the Service ..................................................................... 42
Obtain the Service Status .............................................................. 43
Stop the Service .......................................................................... 43
Start the Service ......................................................................... 43
Restart the Service ...................................................................... 44
Re-read All systemd Unit Files. ....................................................... 44
Local License Server Components .................................................... 44
Post-Installation Configuration on Windows .......................................... 48
  Edit “flexnetsls.settings” ............................................................... 48
  Edit “local-configuration.yaml” (Windows) ........................................ 49
Post-Installation Configuration on Linux .............................................. 53
  Edit Settings in “flexnetsls.conf” ................................................... 54
  Edit the DEFINES Setting ............................................................ 54
  Enable systemd Logging .............................................................. 55
  Edit “local-configuration.yaml” (Linux) ........................................... 55
Define Command-Line Options for HTTPS Configuration Files ............... 59
More About systemd Logging .......................................................... 60
View Journal Entries ..................................................................... 60
Configure the Journal ................................................................... 61
Understanding Hostids ................................................................... 61
  Hostid Types ........................................................................... 61
  Specifying a Server Hostid .......................................................... 62
  Retrieving Server Hostids ............................................................ 63
Uninstalling the Local License Server .................................................. 63
  Uninstall on Windows ................................................................ 63
  Uninstall on Linux ................................................................. 64
Preparing to Use the License Server Manager ........................................ 65
Base URL for the License Server ........................................................ 65
Managing Administrative Security on a Local License Server or CLS Instance .................................................. 67
  Security Work Flow in the Enterprise ............................................. 68
  Secured Functionality on the License Server ................................. 70
  User Roles Defining Administrative Privileges ................................. 71
  Administrative Security Policies on the License Server ..................... 71
  Credential Requirements ............................................................ 73
  HTTPS Recommended ............................................................... 74
Next Steps .................................................................................... 74
3 Using the FlexNet License Server Administrator Command-line Tool

Before You Start
   License Server URL Designation
   About the Example Commands

Using the Command-line Tool to Manage Administrative Security
   Base URL for the Secured License Server
   Reset Your Administrator Password
   Determine Administrative Security Policies
   Create and Manage Other Enterprise User Accounts
      Creating an Enterprise User Account
      Creating Another Administrator Account
      Edit a User or Administrator Account
      Deleting a User or Administrator Account
      Viewing All User Accounts
   Use Credentials to Access Operations

General Server Maintenance
   Obtain the License Server Status
   Show Version Information for the Administrator Command-line Tool
   Show General Configuration Information for the License Server
   Suspend or Resume the License Server

Activating License Rights on the Server
   Online Activation
   Offline Activation

Returning License Rights to the Server
   Online Return
   Offline Return

Viewing Features Installed on the License Server

Managing Feature Partitions
   Define a Model Definition
   Upload the Model Definition
   View the Model Definition
   Delete the Model Definition
   View Partitions

Managing Reservations
   Add Reservations
      Confirm Reservations Posting
   View Reservations
      Show a Summary of Reservation Groups
      Show Reservation Details by Group
   Back Up Reservation Information to a File
   Delete Reservations
      Delete All Reservations in a Group
      Delete All Reservation Entries in a Given Reservation

Managing License Server Policy Settings
   Review Policy Settings

Review Policy Settings
## Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Settings for Licensing Operations</td>
<td>103</td>
</tr>
<tr>
<td>Synchronization Settings</td>
<td>104</td>
</tr>
<tr>
<td>Administrative Security Policies</td>
<td>105</td>
</tr>
<tr>
<td>Log Settings</td>
<td>105</td>
</tr>
<tr>
<td>Settings for Polling the Back Office for License Updates</td>
<td>106</td>
</tr>
<tr>
<td>Failover Settings</td>
<td>106</td>
</tr>
<tr>
<td>All Settings</td>
<td>107</td>
</tr>
<tr>
<td>Write Editable Policy Settings to a File</td>
<td>107</td>
</tr>
<tr>
<td>Override Policy Settings</td>
<td>108</td>
</tr>
<tr>
<td>Override Individual Settings</td>
<td>108</td>
</tr>
<tr>
<td>From an Input File</td>
<td>109</td>
</tr>
<tr>
<td>Reset Policy Settings</td>
<td>109</td>
</tr>
<tr>
<td>Monitoring License Distribution to Clients</td>
<td>110</td>
</tr>
<tr>
<td>View a Summary of Features and Active Clients</td>
<td>110</td>
</tr>
<tr>
<td>View License Details</td>
<td>110</td>
</tr>
<tr>
<td>Viewing Current Binding Status</td>
<td>112</td>
</tr>
<tr>
<td>Summary of flexnetlsadmin Commands</td>
<td>113</td>
</tr>
<tr>
<td><strong>4 Using the FlexNet License Server Manager</strong></td>
<td>125</td>
</tr>
<tr>
<td>Setting Up the FlexNet License Server Manager</td>
<td>125</td>
</tr>
<tr>
<td>Providing Credentials on a Secured License Server</td>
<td>127</td>
</tr>
<tr>
<td>Page Navigation for Views in the License Server Manager</td>
<td>128</td>
</tr>
<tr>
<td>Devices View</td>
<td>128</td>
</tr>
<tr>
<td>Reservations View</td>
<td>129</td>
</tr>
<tr>
<td>Feature Usage View</td>
<td>133</td>
</tr>
<tr>
<td>Offline Server Updates View</td>
<td>134</td>
</tr>
<tr>
<td>Client Key Operations View</td>
<td>135</td>
</tr>
<tr>
<td>Properties View</td>
<td>135</td>
</tr>
<tr>
<td>License Server Manager Information and Settings</td>
<td>136</td>
</tr>
<tr>
<td><strong>5 More About License Server Functionality</strong></td>
<td>139</td>
</tr>
<tr>
<td>General Information</td>
<td>139</td>
</tr>
<tr>
<td>License Borrowing</td>
<td>140</td>
</tr>
<tr>
<td>Determining the Effective Borrow Interval</td>
<td>141</td>
</tr>
<tr>
<td>Setting the Admin Borrow Interval</td>
<td>141</td>
</tr>
<tr>
<td>Setting the Admin Borrow Interval Using flexnetlsadmin</td>
<td>142</td>
</tr>
<tr>
<td>Setting the Admin Borrow Interval Using the FlexNet License Server Manager</td>
<td>142</td>
</tr>
<tr>
<td>Borrow Granularity</td>
<td>142</td>
</tr>
<tr>
<td>Renew Interval</td>
<td>143</td>
</tr>
<tr>
<td><strong>Feature Partitions</strong></td>
<td>143</td>
</tr>
<tr>
<td>Defining Partitions Using a Model Definition</td>
<td>144</td>
</tr>
<tr>
<td>Model Definition Components</td>
<td>144</td>
</tr>
<tr>
<td>Model</td>
<td>145</td>
</tr>
<tr>
<td>Partitions</td>
<td>146</td>
</tr>
</tbody>
</table>
## Contents

Rules and Conditions .................................................................................................................. 146
   Rule Syntax .............................................................................................................................. 147
   Condition Types ...................................................................................................................... 150
Server Behavior When Distributing Feature Counts to Partitions .............................................. 152
   When a New Model Definition Is Uploaded ........................................................................... 152
   When the Feature Count Changes on the License Server ...................................................... 153
   When Clients Return or Renew Counts to the Server ............................................................ 153
   Basic Redistribution Rules. .................................................................................................. 153
   Order of Redistributing Counts ............................................................................................ 154
Server Behavior when Assigning Features to Clients ................................................................. 154
Partitions vs. Reservations .......................................................................................................... 156
   Comparison of Partitions and Reservations ........................................................................ 157
   Impact of Partitions Functionality on Using Reservations ................................................... 158
   Migrating Reservations to Partitions .................................................................................... 158
   Returning to Reservations after Migrating to Partitions ..................................................... 159
Limitations of Partitions ............................................................................................................ 159
License Reservations .................................................................................................................. 160
   Overview of Reservation Types ............................................................................................. 160
   Reservation Hierarchy ............................................................................................................ 161
   Managing Reservations ........................................................................................................ 162
   Definitions in JSON Format .................................................................................................. 162
   Add a New Reservation Group. .............................................................................................. 163
   Add or Delete Reservations in an Existing Group. ................................................................. 164
Disabled Reservations ................................................................................................................ 164
   License Allocation on the Server ............................................................................................ 165
   When Adding a Reservation Group ....................................................................................... 165
   When Feature Counts Change on the Server ........................................................................ 166
Processing the Capability Request When Reservations Are Used .............................................. 166
   Basic Process for Granting Licenses. ..................................................................................... 166
   Example Scenarios of the Basic Process for Granting Licenses .......................................... 167
Reservation Limitations .............................................................................................................. 169
Online Synchronization to the Back Office .................................................................................. 169
   Enablement ............................................................................................................................ 169
   Data Synchronized During Online Synchronization ............................................................. 170
   Configuring the Synchronization. ......................................................................................... 170
Offline Synchronization to the Back Office .................................................................................. 170
   Configuring Synchronization Page Size ................................................................................ 171
   Synchronization Tools .......................................................................................................... 171
   Offline Synchronization Process ........................................................................................... 171
Status of Synchronization to the Back Office ............................................................................. 174
Synchronization From the Back Office ......................................................................................... 174
   Enablement ............................................................................................................................ 175
   Synchronization Process ....................................................................................................... 175
   Viewing Restored Data .......................................................................................................... 175
   Reviewing Synchronization Settings. .................................................................................... 175
Contents

License Server Failover ......................................................... 176
 Configuring Server Failover ........................................... 176
 Editing Failover Configuration Settings ...................................... 178
 (Optional) Automatic Registration of the Failover Pair ................. 178
 Additional Failover Considerations ........................................ 179
 Outgoing HTTPS ............................................................. 180
 Default Server Configuration ............................................... 180
 Server Configuration When Another Certificate Is Used .................. 180
 New Installations .......................................................... 181
 Existing Installations ....................................................... 181
 Incoming HTTPS ............................................................. 182
 Step 1: Obtain Certificate .................................................. 183
 Step 2: Enable Access to “server” Certificate .............................. 183
 New Installations .......................................................... 183
 Existing Installations ....................................................... 184
 Step 4: Define Scope of HTTPS Communications ......................... 185
 Proxy Support for Communication with the Back Office ................. 185
 Configuring the License Server for Proxy Support ....................... 186
 Proxy Configuration Basics ............................................... 186
 Supported Proxy Parameters ............................................. 186
 Parameter Format ......................................................... 186
 Obfuscating the Proxy Password ......................................... 187
 Trusted Storage Backup and Restoration .................................. 187
 Overview of the Backup Process .......................................... 187
 Running a Trusted Storage Restoration .................................... 188
 Public Key Upload ........................................................... 189

6 Managing a CLS Instance .................................................. 191
 Attributes of the CLS Instance ............................................ 191
 Searching for a CLS Instance ............................................... 192
 Working with CLS Instances ............................................... 193
 View the History of a License Server ..................................... 193
 View Served Clients of a CLS Instance ................................... 194
 Map Entitlements to a CLS Instance ....................................... 194
 Remove Licenses from a CLS Instance ................................... 194
 Move a CLS Instance ...................................................... 195
 Creating a CLS Instance .................................................... 195
 Managing Administrative Security ........................................ 196
 Set Your Administrator Password ......................................... 196
 Create and Manage Other User Accounts .................................. 196
 Manage the License Server ............................................... 197

A Reference: License Server Policy Settings .............................. 199
Log4J SocketAppender .......................................................... 246
FlexNet Embedded Local License Server Quick Start Reference

This “quick start” reference is geared towards those license server administrators who are familiar with the FlexNet Embedded local license server and need only a reference to commands and basic steps to get the server up and running.

The instructions in this “quick start” reference refer to the following components:

- The executable for the FlexNet Embedded local license server, `flexnetls`, to manage the license server service
- The FlexNet License Server Administrator command-line tool to configure and administer the server

Additionally, these instructions provide no background information about the task at hand and no details about the specific commands being used. If you need more information, refer to appropriate chapters in this book.

This “quick start” reference contains the following topics:

- Installing and Running the License Server
- Performing Other Service Maintenance Tasks
- License Server Administrator Command-line Tool: Command Summary

About “flexnetlsadmin” Commands

Refer to the following for additional assistance about entering the `flexnetlsadmin` commands listed in this “quick start” reference:

- `flexnetlsadmin` commands require the license server’s base URL (`licenseServer_baseURL`). For details about the base URL, see Base URL for the License Server in the Getting Started chapter.

- If administrative security is enabled on the license server, you might need to provide authorization credentials to perform certain `flexnetlsadmin` operations listed in next sections. For more information about providing these credentials (and about configuring and managing security), see Using the Command-line Tool to Manage Administrative Security in the Using the FlexNet License Server Administrator Command-line Tool chapter.

- For a summary `flexnetlsadmin` command usage, see License Server Administrator Command-line Tool: Command Summary.
Installing and Running the License Server

This section covers the “quick start” steps used to install and start the FlexNet Embedded local license server as a service on Windows or Linux.

On a Linux platform, you install the license server as a systemd service, as described here. Alternatively, you can install the license server as a SysV service (see SysV Alternative for Installation on Linux).

For help in using `flexnetlsadmin` commands, see About “flexnetlsadmin” Commands.

<table>
<thead>
<tr>
<th>Task</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td><strong>Configure the installation and service</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>Install the service</strong></td>
</tr>
<tr>
<td><strong>3</strong></td>
<td><strong>Start the service</strong></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td><strong>Do a “health check” on the license server</strong></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>Activate license rights</strong></td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>Verify license rights</strong></td>
</tr>
</tbody>
</table>
Performing Other Service Maintenance Tasks

The following provides a quick reference to other tasks used to maintain the license server as a service.

The tasks listed for the license server running on a Linux platform assume the service was installed using systemd. If you have installed the license server as SysV service on a Linux platform, see SysV Alternative for Installation on Linux for details about managing the service.

For help in using flexnetlsadmin commands, see About “flexnetlsadmin” Commands.

Table -2 • Additional Service Maintenance Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update service with changed options</td>
<td>1. Edit flexnetls.settings as needed&lt;br&gt;2. Run flexnetls.bat -update&lt;br&gt;3. Run flexnetls.bat -start</td>
</tr>
<tr>
<td>Note • Depending on the security policies defined on your device, a warning message might be displayed, warning you to only run scripts that you trust. If such a warning is displayed, confirm that you want to allow each script to be executed.</td>
<td></td>
</tr>
<tr>
<td>Update service settings:</td>
<td>1. Run sudo systemctl stop flexnetls-producer_name&lt;br&gt;2. Run sudo gedit /etc/systemd/system/flexnetls-producer_name.service.d/flexnetls.conf to make edits (or replace gedit with editor of choice)&lt;br&gt;3. Run sudo systemctl daemon-reload&lt;br&gt;4. Run sudo systemctl start flexnetls-producer_name</td>
</tr>
<tr>
<td>Update local settings:</td>
<td>1. Run sudo gedit /opt/flexnetls/demo/local-configuration.yaml to make edits (or replace gedit with editor of choice)&lt;br&gt;2. Run sudo systemctl restart flexnetls-producer_name</td>
</tr>
</tbody>
</table>
The following describes the command options and associated arguments used by the FlexNet License Server Administrator command-line tool:

- **Usage**
- **Additional Information**
- **Command Summary**

## Usage

The following shows general command usage for the license server administrator and an enterprise user.

### For a License Server Administrator

The following is command usage for a license server administrator performing general administrative tasks:

```
flexnetlsadmin -server LicenseServer_baseURL [-authorize adminName {adminPassword} -passwordConsoleInput] -command_option [command_option_arguments] [-command_option...]
```

The following is the command usage for a license server administrator managing user accounts when administrative security is enabled on the license server:

```
flexnetlsadmin -server LicenseServer_baseURL -authorize adminName {adminPassword} -passwordConsoleInput -users [-create|edit|delete]
```
For an Enterprise User

The command usage for an enterprise user is the following:

```
flexnetlsadmin -server licenseServer_baseURL [-authorize userName {userPassword |
    -passwordConsoleInput}] -command_option [command_option_arguments] [-command_option...]
```

Additional Information

Refer to the previous section About “flexnetlsadmin” Commands for additional assistance in providing the licenseServer_baseURL or required authorization credentials to run the flexnetlsadmin commands.

Command Summary

The follow table summarizes flexnetlsadmin command options and arguments.

**Table -3** • FlexNet License Server Administrator Command-line Tool Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help</td>
<td>(no argument)</td>
<td>Lists the syntax of all flexnetlsadmin commands.</td>
</tr>
<tr>
<td>-server</td>
<td>(no argument)</td>
<td>Sets the license server base URL required in every command as the first option. For a local license server, use the following URL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flexnetlsadmin.bat -server <a href="http://LicenseServerURL/api/1.0/instances/instanceID">http://LicenseServerURL/api/1.0/instances/instanceID</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a CLS instance, use the URL that the producer provides, similar to this:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://siteID.compliance">https://siteID.compliance</a>. flexnetoperations.com/api/1.0/instances/instanceId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternatively, set this value as the environment user variable FLEXNETLS_BASEURL on your machine so that you no longer have to include this option in the commands. See License Server URL Designation for more information.</td>
</tr>
</tbody>
</table>

**Note** • Use of the HTTPS protocol on the local license is strongly recommended when administrative security is enabled on the license server.
The `-authorize` command option and its related options and arguments are in effect only when administrative security is enabled on the license server. Position the `-authorize` option after the `-server` option and before any other options. Administrators can choose to enter their password directly in the command or, for security reasons, use the `-passwordConsoleInput` option, which then issues a prompt to enter the password as console input. The remaining options and arguments for `-authorize` are described next. See Using the Command-line Tool to Manage Administrative Security for details.

**Note** • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-authorize adminName`  | `adminName` `{adminPassword | Creates a new user account:  
  `-passwordConsoleInput}`         | • The `adminName` and `adminPassword` must meet the criteria specified in Credential Requirements.  
  • To avoid exposing the password as a command-line argument, you can use the `-passwordConsoleInput` option, which then issues a prompt to enter the password as console input.  
  • Roles are optional. If no role is specified, ROLE_READ is assigned by default. Valid roles include ROLE_READ, ROLE_RESERVATIONS, ROLE_ADMIN, ROLE_DROP_CLIENT, and ROLE_PRODUCER. Multiple roles can be combined using a plus sign (+) without any spaces in between (for example, ROLE_READ+ROLE_RESERVATIONS). See User Roles Defining Administrative Privileges for details.  
  • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password. |
| `{-users}               | Shows all current user accounts on the license server. | |
Table - 3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| `-authorize adminName adminPassword -passwordConsoleInput` (cont.) | `-users -edit userName {password|newPassword} -passwordConsoleInput [newRoles]` | Updates the password or role or both for the user account identified by `userName`.

- The password is not optional. Enter either the user’s current password or, if changing the password, the new password (which must meet the criteria listed in Credential Requirements).
- To avoid exposing the password as a command-line argument, you can use the `-passwordConsoleInput` option, which then issues a prompt to enter the password as console input.
- Assignment of new roles is optional. If one or more new roles are specified, they replace all current roles. If no roles are specified, the user’s current role assignment remains in effect. For information about roles, see User Roles Defining Administrative Privileges.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

| `-users -delete userName` | Deletes the user account specified by `userName` |
| [any options listed below] | Authorizes your access, as needed, to operations specified by any of the options listed below. |
Table 3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-authorize userName</td>
<td>[specific options listed below]</td>
<td>Authorizes an enterprise user’s access to the specific operations to which that user has privileges. Users can choose to enter their password (as defined for their account) directly in the command or, for security reasons, use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.</td>
</tr>
<tr>
<td>-status</td>
<td>[-suspend</td>
<td>-resume]</td>
</tr>
<tr>
<td>-hostid</td>
<td>(no argument)</td>
<td>Displays the license server’s hostid value used to fulfill capability requests against a back-office server. If the server has multiple hostid values, the list contains the available hardware Ethernet addresses and dongle IDs. If virtual hosts are supported, the VM UUID will also be listed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-selected</td>
<td></td>
<td>Returns the current “selected” hostid—that is, the hostid currently used by the license server.</td>
</tr>
<tr>
<td>-setactive hostIdValue hostIdType</td>
<td></td>
<td>Designates a new “selected” hostid. The new hostid must be one of the hostids returned by the -hostid command.</td>
</tr>
</tbody>
</table>
Table -3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-activate</td>
<td>-id activation-id [-count value]</td>
<td>Activates licenses on the server by installing the specified number of copies of license rights (identified by the activation ID) from the back office. You can specify the -activate option multiple times to activate license rights from different activation IDs.</td>
</tr>
<tr>
<td></td>
<td>-activate -id activation_id -count value filename</td>
<td>(Part 1 of an offline activation) Generates a capability request containing the activation ID and saves the request as a binary file. To generate the request with multiple activation IDs, repeat the -activate option for each ID.</td>
</tr>
<tr>
<td></td>
<td>-load filename</td>
<td>(Part 2 of an offline activation) Processes the offline capability-response binary file from the back office to install licenses on the license server.</td>
</tr>
<tr>
<td>-licenses</td>
<td>[-verbose]</td>
<td>Retrieves summary information or details (with -verbose) about current license distribution to client devices.</td>
</tr>
<tr>
<td>-features</td>
<td>(no argument)</td>
<td>Shows details about the features in the current license pool on the server.</td>
</tr>
</tbody>
</table>
Chapter FlexNet Embedded Local License Server Quick Start Reference
License Server Administrator Command-line Tool: Command Summary

Table -3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| -config  | [-filter none | license-policy | sync | security | log | general | capability | failover] | Lists all license-server policy settings. Use the -filter argument to list settings for only specific categories. The categories include:  
  * none: All categories  
  * license-policy: Policies for license distribution and usage  
  * sync: Policies for synchronization with back office  
  * security: The policy enabling administrative security on the license server  
  * log: Logging parameters  
  * general: Attributes identifying the license server  
  * capability: Policies for polling the back-office for license updates  
  * failover: License server failover  
You can specify multiple categories for the -filter option. For example, flexnatlsadmin -config -filter sync log shows synchronization and logging settings. (Use -config or -config -filter none to list all settings.) Those settings that you can edit are listed with an asterisk.  
- set settingName=settingValue, settingName=settingValue... Overwrites the current value for the specified policy setting. You can provide multiple value pairs separated by commas. For additional syntax formats, see Override Policy Settings.  

Note • To determine which settings are editable, use “-config” to list the settings; editable settings are marked with an asterisk. Additionally, when you use “-o” to write settings to a file, only editable settings are written. Another way to apply edits to multiple settings is to use “-filter category -o <filename>” to write the editable settings to file, edit the settings as needed, and then use “-load <filename>” to apply the edits.  
- reset settingName,settingName... Resets one or more policy settings back to their original default values set by the producer. Separate multiple setting names by commas. For additional syntax formats, see Override Policy Settings.
### Table 3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-config (cont.)</td>
<td>-o filename</td>
<td>Writes the current, editable policy settings (except those is the general category) to the specified file in JSON format. This argument can be used with the -filter none option to write all editable settings to file or with the -filter category option (except for the general category) to limit the scope of editable settings written to the file.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Applies the edits to policy settings from the specified file to the existing policy settings. The contents of the file you are loading must be in JSON format. In a typical scenario, first run -filter category -o filename to write current editable settings to a file, edit the values as needed, and then apply the edits using the -load filename option.</td>
</tr>
<tr>
<td>-model</td>
<td>(no argument)</td>
<td>Displays the model definition that is currently active, showing the partitions and rules.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Uploads the model definition to the license server. The contents of the file you are loading must be written according to the grammar in Model Definition Grammar and Syntax—EBNF. Only one model definition can be active for each license server instance. You cannot upload a model definition with the name reservations or default, because these names are reserved. <strong>Important</strong> • You can use either reservations or partitions. <strong>Uploading a model definition deletes all currently configured reservations. For more information, see Partitions vs. Reservations.</strong></td>
</tr>
<tr>
<td>-delete</td>
<td></td>
<td>Deletes the model definition that is currently applied to the license server. The default model definition is automatically applied to the license server.</td>
</tr>
<tr>
<td>-partitions</td>
<td>(no argument)</td>
<td>Displays all partitions for the license server. The output lists information about every feature and every feature slice in each partition (feature slices are portions of feature counts allocated to a partition).</td>
</tr>
</tbody>
</table>
Provides a summary of the reservation groups, including each group’s ID, currently available to the license server.

The `-group group_id` option shows details—at the reservation (hostid) and reservation-entry (feature) levels—for the specified reservation group.

The `-o filename` option writes the reservation information in JSON format (for all groups or for the specified group ID) to a file.

**Note** • To make changes to a reservation group, delete the group and recreate it with the changes to the reservation definitions.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-reservations</td>
<td>[-group group_id] [-o filename]</td>
<td>Provides a summary of the reservation groups, including each group’s ID, currently available to the license server. The <code>-group group_id</code> option shows details—at the reservation (hostid) and reservation-entry (feature) levels—for the specified reservation group. The <code>-o filename</code> option writes the reservation information in JSON format (for all groups or for the specified group ID) to a file.</td>
</tr>
<tr>
<td>-genjson filename</td>
<td></td>
<td>Generates a template file for reservations in JSON format.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Creates a reservation group and its reservation entries via a file containing the reservation definitions in JSON format.</td>
</tr>
<tr>
<td>-delete -group group_id</td>
<td></td>
<td>Deletes the entire reservation group with all its reservations and reservation entries.</td>
</tr>
<tr>
<td>-delete -group group_id -reservation reservation_id</td>
<td></td>
<td>Deletes the specified reservation within the specified reservation group.</td>
</tr>
</tbody>
</table>

**Important** • You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-uploadPublicKey</td>
<td>clientPublicKeyFileName</td>
<td>Uploads a file containing a client-side RSA public key (2048-bit DER-encoded) in octet-stream format to the license server. The producer will provide details about obtaining and uploading this key should such a key be required. License-server administrator credentials are required to use this option.</td>
</tr>
<tr>
<td>-binding</td>
<td>(no argument)</td>
<td>Shows both the current policy for a binding-break detection facility (enabled by the producer) and the current binding status as detected by this facility (ok, hard break, or soft break). If this facility is not enabled, the output indicates as such.</td>
</tr>
</tbody>
</table>
Table -3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-version</td>
<td>(no argument)</td>
<td>Shows the version and build information for the FlexNet License Server Administrator command-line tool.</td>
</tr>
<tr>
<td>-json</td>
<td>(no argument)</td>
<td>Displays output in JSON format instead of the default table layout.</td>
</tr>
</tbody>
</table>

Note • You can query the JSON output of flexnetlsadmin with an external tool like jq. For information about jq, see https://stedolan.github.io/jq/manual/v1.6/.
Introduction

The FlexNet Embedded license server provides functionality for serving and monitoring a counted pool of licenses for intelligent devices or software clients using products enabled with FlexNet Embedded licensing. The license server is designed to administer and enforce a pool of licenses within your enterprise, report license usage to the back office, and provide served-license status information in conjunction with client code that incorporates FlexNet Embedded.

This guide describes how to administer the FlexNet Embedded license server.

Local License Servers vs. License Servers Hosted in the Cloud

The FlexNet Embedded license server can be deployed as a local license server at your site or as a Cloud Licensing Service (CLS) instance, hosted in the Revenera Cloud. Both the FlexNet Embedded local license server and the CLS instance provide the same functionality.

Note that the instructions or descriptions in this book focus on the local license server, basically because the setup and deployment of the CLS instance and a good deal of its administration are performed in FlexNet Operations. However, this book directs you to the appropriate FlexNet Operations documentation to help you get started with the CLS instance. Additionally, the book includes the Managing a CLS Instance chapter to provide basic procedures for managing the CLS instance.

Finally, because both license servers share the same functional endpoints, license server administrator tools provided by the producer (for example, flexnetlsadmin) can be used to manage important features, such as license reservations, administrative security, and enterprise user accounts, for both license server types. This book notes these areas where functionality and procedures apply to both server types.
What's in this Guide

The *FlexNet Embedded License Server Administration Guide* includes the following chapters (in addition to the FlexNet Embedded Local License Server Quick Start Reference at the beginning of this book):

**Table 1-1 • Contents of the *FlexNet Embedded License Server Administration Guide***

<table>
<thead>
<tr>
<th>Topic</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Provides an overview of the book and list of conventions used in the book's contents.</td>
</tr>
<tr>
<td>FlexNet Embedded Local License Server Quick Start Reference</td>
<td>A reference to commands and basic steps to get the server up and running (geared towards license server administrators who are familiar with the FlexNet Embedded local license server).</td>
</tr>
<tr>
<td>Getting Started</td>
<td>Includes the requirements and procedures to help you configure, install, and start the FlexNet Embedded local license server and implement administrative security for both the local license server and a CLS instance.</td>
</tr>
<tr>
<td>Using the FlexNet License Server Administrator Command-line Tool</td>
<td>Describes how to use the FlexNet License Server Administrator command-line tool to manage the FlexNet Embedded local license server or a CLS instance and its license distribution. (The producer decides whether to provide this tool.)</td>
</tr>
<tr>
<td>Using the FlexNet License Server Manager</td>
<td>Describes how to use the FlexNet License Server Manager to manage the FlexNet Embedded local license server and its license distribution. (The software producer decides whether to provide this tool.)</td>
</tr>
<tr>
<td>More About License Server Functionality</td>
<td>Provides background and set-up information for types of functionality that might be enabled for your FlexNet Embedded local license server or CLS instance.</td>
</tr>
<tr>
<td>Managing a CLS Instance</td>
<td>Highlights procedures used to manage a version of the license server hosted in either the Revenera or producer’s Cloud.</td>
</tr>
<tr>
<td>Reference: License Server Policy Settings</td>
<td>Provides a reference to the license server policy settings used to control license-server operations. Some settings are editable (and are marked as such in the reference).</td>
</tr>
<tr>
<td>SysV Alternative for Installation on Linux</td>
<td>Describes how to install the local license server as a Linux service using the SysV init runlevel system.</td>
</tr>
<tr>
<td>Model Definition Grammar for Partitions</td>
<td>Describes the possible grammar structures of the language used to write a model definition for defining partitions, and provides use case examples for sharing feature counts between partitions.</td>
</tr>
</tbody>
</table>
The following resources are available to assist you with using this product:

- Revenera Product Documentation
- Revenera Community
- Revenera Learning Center
- Revenera Support

**Revenera Product Documentation**

You can find documentation for all Revenera products on the Revenera Product Documentation site:

https://docs.revenera.com

**Revenera Community**

On the Revenera Community site, you can quickly find answers to your questions by searching content from other customers, product experts, and thought leaders. You can also post questions on discussion forums for experts to answer. For each of Revenera's product solutions, you can access forums, blog posts, and knowledge base articles.

https://community.revenera.com

**Revenera Learning Center**

The Revenera Learning Center offers free, self-guided, online videos to help you quickly get the most out of your Revenera products. You can find a complete list of these training videos in the Learning Center.

https://learning.revenera.com

**Revenera Support**

For customers who have purchased a maintenance contract for their product(s), you can submit a support case or check the status of an existing case by making selections on the Get Support menu of the Revenera Community.

https://community.revenera.com
Contact Us

Revenera is headquartered in Itasca, Illinois, and has offices worldwide. To contact us or to learn more about our products, visit our website at:

http://www.revenera.com

You can also follow us on social media:

- Twitter
- Facebook
- LinkedIn
- YouTube
- Instagram
Getting Started

This chapter focuses on how to configure, install, and start the FlexNet Embedded local license server at your site but also includes important set-up instructions that apply to both the local license server and a Cloud Licensing Service (CLS) instance. The following sections are included:

- About Getting Started with a CLS Instance
- Requirements for the Local License Server
- Overview: Administrator Experience on the Local License Server
- Configuring, Installing, and Starting the Local License Server
- Editing the Local Settings Post-installation
- Uninstalling the Local License Server
- Preparing to Use the License Server Manager
- Managing Administrative Security on a Local License Server or CLS Instance
- Next Steps

About Getting Started with a CLS Instance

Depending on the agreement with the producer, either you or the producer creates the Cloud Licensing Service (CLS) instance through FlexNet Operations. For more information about managing the CLS instance, refer to the help system for the FlexNet Operations End-User Portal; or use the Managing a CLS Instance chapter in this book as a quick reference for performing basic administration tasks.

Requirements for the Local License Server

The following sections list the requirements for the FlexNet Embedded local license server:

- Hardware Requirements
Chapter 2  Getting Started
Requirements for the Local License Server

- Validated Platforms
- Virtual Machine Support
- Supported Browsers
- Java Prerequisites
- License Server Manager Requirements

**Important** Due to potential incompatibilities with trusted storage and other stored data, do not downgrade your current license server to a previous version.

## Hardware Requirements

The following are the minimum hardware requirements for the license server:

- **Disk**—500 MB
- **RAM**—4 GB
- **CPU**—2 GHz - 2 Cores

## Validated Platforms

The following table lists the validated platforms for the FlexNet Embedded license.

**Table 2-1 • Validated FlexNet Embedded License Server Platforms**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>CPU Architecture</th>
<th>Operating System Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Linux</em></td>
<td>x86-64</td>
<td>CentOS 7.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CentOS 8.0</td>
</tr>
<tr>
<td><strong>Windows</strong></td>
<td>x86-64</td>
<td>Windows 8.1 and 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Windows Server 2012 and 2012 R2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Windows Server 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Windows Server 2019</td>
</tr>
</tbody>
</table>

* The FlexNet Embedded license server has only been validated on CentOS 7 and 8, but can also be run on RedHat Enterprise Linux 7 or 8.

** .NET Framework 4.5 or later is required for Windows platforms.

## Virtual Machine Support

Virtualization functionality in the FlexNet Embedded local license server supports the following:
• VMware ESXi 6.5
• VMware Workstation 14.1.5
• Microsoft Hyper-V 6.3 on Windows Server 2016
• Citrix XenServer 7.2
• Oracle VirtualBox 6.1.18
• QEMU-KVM 2.7
• Parallels 15.1.2
• Google Compute Cloud
• Amazon EC2
• Microsoft Azure (Windows only)

### Supported Browsers

The FlexNet License Server Manager UI supports the following browsers:

• Mozilla Firefox versions 43 and later
• Google Chrome versions 47 and later
• Microsoft Internet Explorer versions 10 and 11
• Microsoft Edge version 38.1

### Java Prerequisites

The following are the Java prerequisites for the machine where the FlexNet Embedded local license server is installed:

• Oracle Java SE 8, or OpenJDK 8 or OpenJDK 11
• A 64-bit JRE for a 64-bit license server or a 32-bit JRE for a 32-bit license server (not adhering to this requirement can cause the license server to fail to start)
• The `JAVA_HOME` (or `JRE_HOME`) environment variable on your system set to the path for your default JDK (or JRE) installation.

**Note** • *The license server requires only the JRE component. If JRE is your default Java installation, set the JRE_HOME environment variable; if JDK is your default installation, set JAVA_HOME. See the “FlexNet Embedded License Server Release Notes” for details.*

### License Server Manager Requirements

The FlexNet License Server Manager uses Apache Tomcat to deploy the required component `flsm.war`. Ensure that one of these Tomcat versions is used for this deployment:
The FlexNet License Server Manager has been validated on Apache Tomcat 9.x. Revenera recommends running the latest FlexNet License Server Manager on Apache Tomcat 9.x.

Certain platforms supported by the local license server provide a version of Apache Tomcat. To determine whether the operating system on which you intend to run the local license server provides a Tomcat version supported by the License Server Manager (as listed above), see https://wiki.apache.org/tomcat/FAQ/Linux_Unix#Q5. To download an appropriate Tomcat version, go to https://tomcat.apache.org/download-70.cgi.

**Overview: Administrator Experience on the Local License Server**

The following table summarizes the basic tasks you perform as a license server administrator on the local license server. Each task description includes a link to further instructions in this book.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Configure, install, start the license server | This phase involves installing and starting up the FlexNet Embedded local license server as a service. Before installing the license server, you have the option to configure settings that define the local environment in which the license server will run. (These settings can be edited any time after installation as well.)
|       |                                    | See Configuring, Installing, and Starting the Local License Server in this chapter.               |
| 2     | Edit local settings as needed     | The settings defining the license server’s local environment at installation can be edited any time after installation as needed. 
|       |                                    | See Editing the Local Settings Post-Installation in this chapter. Also see the More About License Server Functionality chapter for instructions for configuration you might need to perform to enable certain functionality on the server. |
| 3     | Prepare to use the License Server Manager | The software producer might have provided the FlexNet License Server Manager as a tool with which to administer your license server. If so, before launching this tool, you must have an Apache Tomcat server installed and need to perform some additional configuration steps. 
|       |                                    | See Preparing to Use the License Server Manager in this chapter.                                |
| 4     | Manage administration security    | If administration security on the license server is enabled, you need to reset your password and optionally create other enterprise user accounts. See Managing Administrative Security on a Local License Server or CLS Instance for more information. |
Configuring, Installing, and Starting the Local License Server

Refer to the following sections for information and instructions needed to install and start (or uninstall) the local license server:

- Configure, Install, and Start on Windows
- Configure, Install, and Start on Linux
- Local License Server Components
Configure, Install, and Start on Windows

Use this procedure to install and start the FlexNet Embedded local license server as a service on a Windows platform.

Important • When upgrading your license server, uninstall the old license server service before installing and starting the service for the new license server. For instructions on uninstalling the license server service on Windows, see Uninstall on Windows.

Task

1. Unpack the files received from the software producer into an installation directory.

   Note • Make sure that the “producer-settings.xml” file is located in the same directory as “flexnetls.bat”.

2. Open the flexnetls.settings file (located in the installation directory) in a text editor, and update it with your local environment information; or leave the file as is to accept the default settings. (For example, you might want to change the JAVA_HOME value or uncomment and provide a value for the PORT setting.) When you update any setting, these rules apply:

   • Any setting value that uses a space must be enclosed in quotations
   • Insert no spaces before or after the equal sign (=) in the setting syntax (for example, PORT=7071).

Refer to the following for a description of each setting:

Table 2-3 • Local Settings for the License Server on Windows

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Settings</td>
<td></td>
</tr>
<tr>
<td>FLEXNETJAR</td>
<td>The Java executable file for FlexNet Embedded local license server (default is flexnetls.jar).</td>
</tr>
<tr>
<td>PUBSETTINGS</td>
<td>The license server configuration file generated by the producer (default is producer-settings.xml).</td>
</tr>
</tbody>
</table>
Chapter 2 Getting Started
Configuring, Installing, and Starting the Local License Server

The path for JDK or JRE installation that the license server should use. The flexnetls batch file uses this location to find the necessary java.exe and jvm.dll files.

By default, the license server uses the value of your JAVA_HOME (or JRE_HOME) system environment variable to determine the Java installation location, as indicated by the "%JAVA_HOME:=" or "%JRE_HOME:=" value for this local setting. However, if you want the license server to use a different Java installation on your system, edit this local setting to override the server’s use of the environment variable. (This override pertains to the license server only; your system continues to use the Java installation defined by the system environment variable.) To perform the override, replace the default value (for example, "%JAVA_HOME:=") with the path for the desired Java installation.

Note • The “flexnetls.settings” file includes both a JAVA_HOME and a JRE_HOME setting. If both values are set, JAVA_HOME is used.

Table 2-3 • Local Settings for the License Server on Windows (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA_HOME (or JRE_HOME)</td>
<td>The path for JDK or JRE installation that the license server should use. The flexnetls batch file uses this location to find the necessary java.exe and jvm.dll files. By default, the license server uses the value of your JAVA_HOME (or JRE_HOME) system environment variable to determine the Java installation location, as indicated by the &quot;%JAVA_HOME:=&quot; or &quot;%JRE_HOME:=&quot; value for this local setting. However, if you want the license server to use a different Java installation on your system, edit this local setting to override the server’s use of the environment variable. (This override pertains to the license server only; your system continues to use the Java installation defined by the system environment variable.) To perform the override, replace the default value (for example, &quot;%JAVA_HOME:=&quot;) with the path for the desired Java installation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT</td>
<td>The listening port used by the license server. (If no value is specified here, the server uses 7070 by default.) If the machine on which the license server is running uses multiple network interfaces, you can use the PORT option to specify the interface that you want the license server to use. Simply include the IP address for the interface in square brackets, as shown in this example:</td>
</tr>
</tbody>
</table>

| PORT=127.0.0.1.1443   |                                                             |
### ACTIVE_HOSTID

The hostid to use for the license server.

The hostid can only be set using a configuration file if no hostid has yet been specified for the license server. Once a hostid has been set, it can only be changed using the FlexNet License Server Administrator or the FlexNet License Server Manager.

The syntax is \texttt{value}\texttt{/type} (for example, \texttt{7200014f5df0/Ethernet}). If no value is specified for this setting and no active hostid has yet been set for the license server, the license server uses, by default, the first available Ethernet address on the machine. If using a dongle ID or a hostid other than the first available Ethernet address, specify it here. For more information about hostids, see Understanding Hostids.

**Important** • It is not recommended to change the hostid of a license server that has licenses mapped in the back office (FlexNet Operations). If the hostid is changed to a value that is different to that specified for the license server in the back office, any existing licenses mapped to the license server that is locked to the old hostid in the back office will be orphaned.

To prevent this from happening, it is best practice to return the license server in the back office. During the return operation, the producer can transfer the licenses to a different device; this can be the same machine with the desired hostid. After the transfer, wait for the license server to synchronize with the back office (server synchronization occurs based on synchronization policies or on-demand).

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE_HOSTID</td>
<td>The hostid to use for the license server.</td>
</tr>
</tbody>
</table>

### HTTPS_SERVER_CONFIG

The path to the HTTPS “server” configuration file used to support incoming HTTPS from client devices.

**Note** • The “server” configuration file is being deprecated and will be removed in a future release. Instead of the “server” configuration file, specify parameters to access the truststore file in the \texttt{https-in} setting in \texttt{local-configuration.yaml}. For more information about HTTPS setup on the license server, see Incoming HTTPS in the More About License Server Functionality chapter.
The path to the HTTPS “client” configuration file used to support outgoing HTTPS communication to FlexNet Operations.

Note • The “client” configuration file is being deprecated and will be removed in a future release. Instead of the “client” configuration file, specify parameters to access the truststore file in the https-out setting in local-configuration.yaml. For more information about HTTPS setup on the license server, see Outgoing HTTPS in the More About License Server Functionality chapter.

A user-defined name (sometimes called host name) for the license server. This name is added to server’s capability requests to the back office, where it is then saved and used to identify the license server in the FlexNet Operations Producer and End User portals.

One important use for this setting is that the alias can be included in the initial capability request sent at server registration, providing a helpful name by which users can identify the new server in the portals. If no alias is sent at registration, the server is identified by its hostid. For more information about hostids, see Understanding Hostids.

The suffix used for the Extended Host ID feature. Contact the software producer for details.

One or more system properties (each in -Dkey=value format) that are passed to the Java Runtime system. The license server depends on the Java Runtime Environment to support certain network functionality such as specifying the HTTP proxy.

For example, if you plan to have the license server communicate with the back office through an HTTP proxy, use this setting to identify the proxy parameters needed to configure the server. (For details, see Proxy Support for Communication with the Back Office in the More About License Server Functionality chapter.) The following shows example proxy parameters listed as -D system properties for this setting:

```
EXTRA_SYSPROPERTIES="-Dhttp.proxyHost=10.90.3.133
-Dhttp.proxyPort=3128 -Dhttp.proxyUser=user1a
-Dhttp.proxyPassword=user1apwd35"
```

The entire set of parameters must be enclosed in double quotations, even if you specify only a single parameter, such as

```
EXTRA_SYSPROPERTIES="-Dhttp.proxyHost=10.90.3.133"
```

### Table 2-3 • Local Settings for the License Server on Windows (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS_CLIENT_CONFIG</td>
<td>The path to the HTTPS “client” configuration file used to support outgoing HTTPS communication to FlexNet Operations.</td>
</tr>
<tr>
<td>SERVER_ALIAS</td>
<td>A user-defined name (sometimes called host name) for the license server. This name is added to server’s capability requests to the back office, where it is then saved and used to identify the license server in the FlexNet Operations Producer and End User portals. One important use for this setting is that the alias can be included in the initial capability request sent at server registration, providing a helpful name by which users can identify the new server in the portals. If no alias is sent at registration, the server is identified by its hostid. For more information about hostids, see Understanding Hostids.</td>
</tr>
<tr>
<td>EXTENDED_SUFFIX</td>
<td>The suffix used for the Extended Host ID feature. Contact the software producer for details.</td>
</tr>
<tr>
<td>EXTRA_SYSPROPERTIES</td>
<td>One or more system properties (each in -Dkey=value format) that are passed to the Java Runtime system. The license server depends on the Java Runtime Environment to support certain network functionality such as specifying the HTTP proxy. For example, if you plan to have the license server communicate with the back office through an HTTP proxy, use this setting to identify the proxy parameters needed to configure the server. (For details, see Proxy Support for Communication with the Back Office in the More About License Server Functionality chapter.) The following shows example proxy parameters listed as -D system properties for this setting: EXTRA_SYSPROPERTIES=&quot;-Dhttp.proxyHost=10.90.3.133 -Dhttp.proxyPort=3128 -Dhttp.proxyUser=user1a -Dhttp.proxyPassword=user1apwd35&quot; The entire set of parameters must be enclosed in double quotations, even if you specify only a single parameter, such as EXTRA_SYSPROPERTIES=&quot;-Dhttp.proxyHost=10.90.3.133&quot;.</td>
</tr>
</tbody>
</table>
Important • Another file, “flexnetlsw.xml” file, also contains settings used to run the Windows service. However, do not edit this file. All information in this file is generated and maintained internally and should never be updated through an external means. To update service settings, edit only the “flexnetls.settings” file.

3. Open a command window as the Administrator, and navigate to the installation directory.

4. Execute `flexnetls.bat -install` to install the license server as a service.

5. Execute `flexnetls.bat -start` to start the service.

Note • Depending on the security policies defined on your device, a warning message might be displayed, warning you to only run scripts that you trust. If such a warning is displayed, confirm that you want to allow each script to be executed.

6. Confirm that the service is running by doing one of the following:
   - Execute `flexnetls.bat -status`. (The output should be Service running.)
   - In the Windows Services window (`services.msc`), check that the service FlexNet License Server - `producer_name` has started. (Note that in the Services tab of the Task Manager, the service’s display name is `FNLS-producer_name`.)

   (If you want to stop the service, run `flexnetls.bat -stop`.)

7. To view the license server log, navigate to the server’s logging directory (by default, `C:\Windows\ServiceProfiles\NetworkService\flexnetls\producer_name\logs`), and review the contents of the appropriate .log file.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP_SERVER_HOSTID</td>
<td>The hostid of the back-up license server in a failover configuration with the current license server (as the main server), if the back-up server is “unknown”—that is, not registered—in FlexNet Operations. This setting adds the back-up server’s hostid to the capability request sent by the current license server to the back office. The back office then automatically registers the back-up server in a failover configuration with the main server. This process saves the extra step of having to manually register the failover pair in FlexNet Operations. For more information, see License Server Failover in the More About License Server Functionality chapter. Make sure the back-up server’s hostid is the same type (for example, “ethernet”) as the hostid type of the current (main) license server. You might need to add this setting manually if it is not included as an available option in the settings file. For more information about hostids, see Understanding Hostids.</td>
</tr>
</tbody>
</table>
Note • Once you install the license server as service, you cannot use any of the “flexnetls” non-service command-line options, such as “console” or “install”.

Configure, Install, and Start on Linux

The following instructions describe how to use systemd to configure, install, and run the local license server as a Linux service:

- Files Required for Installation Using systemd
- Task Overview
- Verify systemd-Enablement on Linux System
- Run the Install Script
- Manage the Service

Alternatively, you can configure, install, and run the license server as a Linux service using a SysV init runlevel system. To do so, follow the instructions in SysV Alternative for Installation on Linux instead of the instructions in this chapter.

The content in this section assumes familiarity with the use of the systemd system for managing Linux services. You can find information about systemd at the following site:


Files Required for Installation Using systemd

The following files, distributed with the license server, support the installation and running of the license server service under systemd:

- install-systemd.sh
- install-functions.sh
- producer-settings.xml

Residing in the same directory as flexnetls.jar, these files work together to generate the components required to install and start the license server service in a systemd configuration.

Optionally, the producer might provide an rpm or deb wrapper that copies these required files to the proper location on your device and then installs the license server service. If so, use the instructions provided by the producer for installing the service.

For a list of all files (aside from the systemd installer files) that can be distributed with the license server, see Local License Server Components.

Task Overview

The following is an overview of tasks used to install and run the license server as a Linux service:
**Task 1:** Verify that the Linux system on which the license server is being installed is systemd-enabled (see Verify systemd-Enablement on Linux System).

**Task 2:** Run the install-systemd.sh installer (or your own wrapper), updating certain default configuration settings in the command line as needed (see Run the Install Script).

**Task 3:** Manage the service (see Manage the Service).

**Task 4:** (Optional) Edit configuration settings post-installation as needed (see Post-installation Configuration on Linux).

### Verify systemd-Enablement on Linux System

Before proceeding with the installation, you can run the following command to determine whether the Linux system on which the license server is being installed is systemd-enabled:

```
systemctl
```

If the system is systemd-enabled, the command runs successfully and lists all active units.

### Run the Install Script

You must run install-systemd.sh to create and start the license server service using either the default configuration for the service or a configuration customized through command-line options. See the following sections for details:

- About the Install Script
- Install and Start the Service with the Default Configuration
- Install and Start the Service with a Modified Configuration
- Configuration Values Editable from the Command Line

The commands described in these next sections are run from the directory in which the systemd script files (see About the Install Script) and flexnetls.jar reside.

**Prerequisite**

You must run the install script as a root or sudo user.

### About the Install Script

When run, the install script install-systemd.sh performs the following:

- Generates the service unit file called `/etc/systemd/system/flexnetls-producer_name.service`.

- Generates the configuration unit file called `/etc/systemd/system/flexnetls-producer_name.service.d/flexnetls.conf`. This file contains the configuration settings needed to start the service. It is created with default values, but you can include options in the install-script command line to generate this file with custom values. See Install and Start the Service with a Modified Configuration.

You can also edit this configuration file post-installation as needed, as described in Post-installation Configuration on Linux.
• Generates a `local-configuration.yaml` file in the `/opt/flexnetls/produc` directory. This file contains optional settings specific to the local service environment. Generally, these settings are initially disabled; you can edit or enable them as needed once the service is installed. See Edit “local-configuration.yaml” (Linux) for details.

• By default, enables the standard Syslog process for logging on the license server service. If desired, you can enable Systemd logging to be used in place of Syslog once the service is installed. For details, see More About systemd Logging.

• Starts the license server service.

### Install and Start the Service with the Default Configuration

The following command installs and starts the license server service using the default configuration. For more about the default values used to configure the service, Configuration Values Editable from the Command Line.

<table>
<thead>
<tr>
<th>Task</th>
<th>To install and start the license server service using the default configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the following:</td>
<td><code>sudo ./install-systemd.sh</code></td>
</tr>
</tbody>
</table>

### Install and Start the Service with a Modified Configuration

The following command installs and starts the license server service using a configuration customized through one or more specified command-line options. For a description of the configuration values that can be updated from the install-script command line, see the next section, Configuration Values Editable from the Command Line.

<table>
<thead>
<tr>
<th>Task</th>
<th>To install and start the license server service with updates to the default configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the install script, specifying one or more command-line options to change specific configuration settings (for example, the <code>user</code> value, as shown here):</td>
<td><code>sudo ./install-systemd.sh --user flexnetls01</code></td>
</tr>
</tbody>
</table>

### Configuration Values Editable from the Command Line

Use the following options with the `./install-system.sh` command to edit current configuration settings for the license server service. (Running `./install-system.sh --help` also lists these command-line options.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--program-dir</code> <code>dir</code></td>
<td>The installation location of the license server service (default is <code>/opt/flexnetls/producer_name</code>).</td>
</tr>
</tbody>
</table>
### Manage the Service

Once the license server service is installed, you can manage the service using the systemd command `systemctl`. The following describes basic management functions for the service:

- **Obtain the Service Status**
Configuring, Installing, and Starting the Local License Server

For information about managing the systemd log, if enabled for the license server service, see More About systemd Logging.

Prerequisite

You must perform systemctl tasks as a root or sudo user.

Obtain the Service Status

The following command obtains the “active” or “not active” status of license server service.

Task

To obtain the status of the license server service

Run the following command:

```
sudo systemctl -l status flexnetls-producer_name
```

The -l switch disables truncation of lines in the output.

The following shows example status output. The current service status is highlighted in this excerpt:

```
flexnetls-fnedemo.service - FlexnetLS Local License Server.
Loaded: loaded (/etc/systemd/system/flexnetls-fnedemo.service; enabled; vendor preset: disabled)
Drop-In: /etc/systemd/system/flexnetls-fnedemo.service.d
└─flexnetls.conf
  Active: active (running) since Tue 2018-11-22 14:28:25 GMT; 18h ago
...
```

Stop the Service

The following command stops the license server service.

Task

To stop the license server service

Use this command:

```
sudo systemctl stop flexnetls-producer_name
```

Start the Service

The following command starts the license server service.
Configuring, Installing, and Starting the Local License Server

**Task**  
**To start the license server service**

Use this command:

```
sudo systemctl start flexnetls-producer_name
```

**Restart the Service**

The following command stops and then restarts the license server service without re-reading all the systemd unit files related to all services. This command is useful if you have made changes to the `local-configuration.yaml` file (a non-systemd file) and want the service to read this file, but not all the systemd unit files (see Edit "local-configuration.yaml" (Linux)).

**Task**  
**To restart the license server service**

Run the following command:

```
sudo systemctl restart flexnetls-producer_name
```

**Re-read All systemd Unit Files**

The following command is used to re-read all systemd unit files related to all services. This command is useful when you have made changes manually to the `flexnetls.conf` file and want to apply the changes before restarting the service (see Edit the .conf File Manually).

**Task**  
**To re-read all systemd unit files for all services**

Run the following command:

```
sudo systemctl daemon-reload
```

**Local License Server Components**

The following basic components are usually included in the FlexNet Embedded local license server installation. The location of these components is not listed, as the software producer determines their location within the server's directory structure. Additionally, the software producer might provide a customized version of certain components, using names different from the ones listed here.
In a Windows Installation

These components are included in the license server installation on Windows.

Table 2-5 • FlexNet Embedded Local License Server Components in an Installation on Windows

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flexnetls.jar</td>
<td>The Java executable file for the FlexNet Embedded local license server.</td>
</tr>
<tr>
<td>flexnetls.bat</td>
<td>The batch file used to start the FlexNet Embedded local license server.</td>
</tr>
<tr>
<td>producer-settings.xml</td>
<td>The configuration file, generated by the producer, that defines policies for license server operations. Some of these settings can be overridden using the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager.</td>
</tr>
<tr>
<td>flexnetls.settings</td>
<td>The local settings file containing information needed to run the license server in your local environment. Any of these settings are editable.</td>
</tr>
<tr>
<td>local-configuration.yaml</td>
<td>A file containing optional local settings.</td>
</tr>
</tbody>
</table>

Note • In a future release, this file will replace flexnetls.settings and the https configuration files.

- flexnetls.exe
- flexnetls.xml
- pre-install.vbs
- Set-Permission.ps1
- Remove-Permission.ps1

Files used to run the license server as a Windows service.

Truststore file provided by producer (optional)

Optional file containing root and intermediate certificates used to validate an HTTPS connection to FlexNet Operations.

If the producer is using the Revenera-hosted version of FlexNet Operations, no truststore file is required.

If the producer is using the “on-premises” version of FlexNet Operations and the server certificate has been signed by an unknown (private) certificate authority, then the truststore file is required.

regid.2009-06.com(...).swidtag

The software ID tag file for use with software asset management tools.

- backofficeofflinesynctool.bat
- serverofflinesynctool.bat

(Optional) The set of tools allowing the license server to perform offline synchronization to the back office. The software producer determines whether to provide these tools.

The tools also require the flxPublicTools.jar, EccpressoAll.jar, and commons-codec-1.9.jar files.
Table 2-6 • FlexNet Embedded Local License Server Components in an Installation on Linux

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flexnetls.jar</td>
<td>The Java executable file for the FlexNet Embedded local license server.</td>
</tr>
<tr>
<td>producer-settings.xml</td>
<td>The configuration file, generated by the producer, that defines policies for license server operations. Some of these settings can be overridden using the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager.</td>
</tr>
<tr>
<td>install-functions.sh</td>
<td>Files used to install and run the FlexNet Embedded local license server as a Linux service using systemd, as described in this chapter.</td>
</tr>
<tr>
<td>flexnetls.conf</td>
<td>A file generated during installation and containing the configuration settings needed to start the service. The file can be created with default values, or you can include options in the install-script command line to generate this file with custom values (see Configure, Install, and Start on Linux). You can also update the settings post-installation, as described in Post-Installation Configuration on Linux.</td>
</tr>
<tr>
<td>local-configuration.yaml</td>
<td>A file generated in the installation and containing optional settings specific to the local service environment (see Configure, Install, and Start on Linux). For the most part, these settings are disabled when the file is created, but you can edit or enable them as needed once the service is installed.</td>
</tr>
</tbody>
</table>
Chapter 2 Getting Started

Configuring, Installing, and Starting the Local License Server

Setting the Server Time Zone

The license server can either use its default time zone or Coordinated Universal Time (UTC) for determining a feature’s expiry date, start date, and issue date. Your producer can configure the time zone using the licensing.defaultTimeZone setting in the producer-settings.xml file. Valid values:

- **UTC**— If UTC is set, a feature’s start date is the start of the specified day in Coordinated Universal Time (UTC). Equally, a feature will expire at the end of the day of the configured expiry date in UTC time. This is the default value.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truststore file provided by producer (optional)</td>
<td>Optional file containing root and intermediate certificates used to validate an HTTPS connection to FlexNet Operations. If the producer is using the Revenera-hosted version of FlexNet Operations, no truststore file is required. If the producer is using the “on-premises” version of FlexNet Operations and the server certificate has been signed by an unknown (private) certificate authority, then the truststore file is required.</td>
</tr>
<tr>
<td>regid.2009-06.com(...).swidtag</td>
<td>The software ID tag file for use with software asset management tools.</td>
</tr>
<tr>
<td>flexnetsadmin.sh</td>
<td>(Optional) Files used to run the FlexNet License Server Administrator command-line tool. The software producer determines whether to provide this tool.</td>
</tr>
<tr>
<td>flexnetsadmin.jar</td>
<td>(Optional) The file required to run the FlexNet License Server Manager. The software producer determines whether to provide this tool.</td>
</tr>
<tr>
<td>fne.war</td>
<td>(Optional) The file needed to migrate your custom legacy server administration tool to the FlexNet Embedded local license server. The software producer determines whether to provide this tool.</td>
</tr>
<tr>
<td>flsm.war</td>
<td>(Optional) The file required to run the FLicensingService, a service used to capture the VM-UUID on those Linux virtual platforms that do provide an appropriate mechanism for retrieving this ID. If you intend for your local license server to run on such a virtual Linux platform and require it to use the VM-UUID as its hostid, consult the software producer for further instructions.</td>
</tr>
<tr>
<td>bin directory</td>
<td>(Optional) Special set of files needed to run FNLicensingService, a service used to capture the VM-UUID on those Linux virtual platforms that do provide an appropriate mechanism for retrieving this ID. If you intend for your local license server to run on such a virtual Linux platform and require it to use the VM-UUID as its hostid, consult the software producer for further instructions.</td>
</tr>
<tr>
<td>install-vm-only.sh</td>
<td>(Optional) Readme.text</td>
</tr>
</tbody>
</table>
• SERVER—If SERVER is set, a feature’s start date is the start of the specified day in the server’s default time zone. Equally, a feature will expire at the end of the day of the configured expiry date in the server’s default time zone.

Changes to the time zone setting affect features as they are being provisioned onto the server. It does not affect features that are already on the server.

If you need to change your license server’s time zone setting, contact the software producer.

For a description of license server policies, see Reference: License Server Policy Settings.

Logging Functionality

The logging functionality that is available for the local license server is described in detail in Logging Functionality on the Local License Server. The appendix covers how to configure the logging style and how to integrate license server logging with external systems.

Editing the Local Settings Post-Installation

The license server installation provides (or provides a means to generate) a local settings file that configures the license server for your specific environment. You can use the file as is with its default settings, or you can modify the settings as needed to reflect your environment. The previous section Configuring, Installing, and Starting the Local License Server describes how to update these settings before server installation. Use the following instructions to edit the settings any time after installation.

Post-Installation Configuration on Windows

If you need to modify the license server’s local settings (defined in the flexnetls.settings and local-configuration.yaml files) after you have installed the server as a service on a Windows platform, use either of these steps:

• Edit “flexnetls.settings”
• Edit “local-configuration.yaml” (Windows)

Important • Never edit the “flexnetlsw.xml” file, which also contains settings used to run the Windows service. All information in this file is generated and maintained internally and should never be updated through an external means. To update service settings, edit only the “flexnetls.settings” file.

Edit “flexnetls.settings”

For a list of settings of flexnetls.settings (located in the installation directory), see Local Settings for the License Server on Windows.

Important • Note that the rate-limit setting (used to control the license server load) and loggingStyle (for specifying rollover, JSON formatting and timestamp behaviour in the server log) are available only in the local-configuration.yaml file (see Edit “local-configuration.yaml” (Windows)).
Chapter 2  Getting Started

Editing the Local Settings Post-Installation

Task  To modify local settings after installing the license server as a service on Windows

1. As an Administrator, open a command prompt window, and navigate to the directory where the license server is installed.

2. Open the flexnetls.settings file (located in the installation directory) in a text editor, and edit the settings as needed. (For example, you can uncomment settings or change existing values.) See the previous section Configure, Install, and Start on Windows for details about the editing process.

3. Save the file.

4. Execute `flexnetls.bat -update` to update the settings.

5. Execute `flexnetls.bat -start` to start the license server service, or restart it from the Windows Services window.

   Note • Depending on the security policies defined on your device, a warning message might be displayed, warning you to only run scripts that you trust. If such a warning is displayed, confirm that you want to allow each script to be executed.

Edit “local-configuration.yaml” (Windows)

Use the following procedure to update the local-configuration.yaml file (located in the same directory as flexnetls.jar). This file contains optional settings for the license server service showing the default values, but commented out.

Edit Settings in the “local-configuration.yaml” file

Use the following procedure to enable, edit, or disable settings in local-configuration.yaml file.

Task  To edit settings in the “local-configuration.yaml” file

1. As an Administrator, open a command prompt window, and navigate to the directory where local-configuration.yaml is installed.

2. In a text editor, open local-configuration.yaml. Uncomment (or comment out) lines and edit setting values as needed. For a description of the settings, see the next section.

3. Save the file.

4. Execute `flexnetls.bat -start` to start the license server service, or restart it from the Windows Services window.

   Note • Depending on the security policies defined on your device, a warning message might be displayed, warning you to only run scripts that you trust. If such as warning is displayed, confirm that you want to allow each script to be executed.
Settings in the “local-configuration.yaml” File

The following settings can be edited directly in the configuration-local.yaml file. Note that a space after the colon is required for all entries.

Table 2-7 • Settings in the “local-configuration.yaml” File

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port: port</td>
<td>The override for the listening port used by the license server, as described in Configuration Values Editable from the Command Line. You can edit this override value as needed. If no override is specified (that is, the setting is commented out), the service uses the default port 7070 (or the value defined in producer-settings.xml).</td>
</tr>
<tr>
<td>active-hostid: value/type</td>
<td>The hostid to use for the license server. The hostid can only be set using a configuration file if no hostid has yet been specified for the license server. Once a hostid has been set, it can only be changed using the FlexNet License Server Administrator or the FlexNet License Server Manager. The syntax is value/type (for example, 720014f5df0/Ethernet). If no value is specified for this setting and no active hostid has yet been set for the license server, the license server uses, by default, the first available Ethernet address on the machine. If using a dongle ID or a hostid other than the first available Ethernet address, specify it here. For more information about hostids, see Understanding Hostids.</td>
</tr>
</tbody>
</table>

Important • It is not recommended to change the hostid of a license server that has licenses mapped in the back office (FlexNet Operations). If the hostid is changed to a value that is different to that specified for the license server in the back office, any existing licenses mapped to the license server that is locked to the old hostid in the back office will be orphaned.

To prevent this from happening, it is best practice to return the license server in the back office. During the return operation, the producer can transfer the licenses to a different device; this can be the same machine with the desired hostid. After the transfer, wait for the license server to synchronize with the back office (server synchronization occurs based on synchronization policies or on-demand).

For more information about hostids, see Understanding Hostids.
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-hostid: value/type</td>
<td>The hostid of the back-up license server in a failover configuration with the current license server (as the main server), if the back-up server is “unknown”—that is, not registered—in FlexNet Operations. The syntax is value/type (for example, 7200014f5df0/Ethernet). For more information about hostids, see Understanding Hostids. This setting adds the back-up server's hostid to the capability request sent by the current license server to the back office. The back office then automatically registers the back-up server in a failover configuration with the main server. This process saves the extra step of having to manually register the failover pair in FlexNet Operations. For more information, see License Server Failover in the More About License Server Functionality. Make sure the back-up server's hostid is the same type (for example, “ethernet”) as the hostid type of the current (main) license server.</td>
</tr>
<tr>
<td>extended-suffix: suffix</td>
<td>The suffix used for the Extended Host ID feature. Contact the software producer for details.</td>
</tr>
<tr>
<td>server-alias: alias</td>
<td>A user-defined name (sometimes called host name) for the license server. This name is added to server's capability requests to the back office, where it is then saved and used to identify the license server in the FlexNet Operations Producer and End User portals. One important use for this setting is that the alias can be included in the initial capability request sent at server registration, providing a helpful name by which users can identify the new server in the portals. If no alias is sent at registration, the server is identified by its hostid.</td>
</tr>
<tr>
<td>jni-helper-directory: path</td>
<td>If the default location for the JNI shared library (/tmp) is not suitable (for example, because execute permission is required), use this setting to supply an alternative directory path. The directory has to exist and be writable by the server.</td>
</tr>
<tr>
<td>logging-threshold: level</td>
<td>The override for the lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG. You can edit this override value as needed. If no override is specified (that is, the setting is commented out), the service uses the default level INFO (or the value defined in producer-settings.xml). For more information about logging levels, see section Logging Policies in Reference: License Server Policy Settings.</td>
</tr>
</tbody>
</table>
Table 2-7 • Settings in the “local-configuration.yaml” File (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
</table>
| loggingStyle: style | The logging style determines rollover, JSON formatting and timestamp behaviour. If no logging style is specified, the default CONTINUOUS is used. *

**Logging style values:**

- **CONTINUOUS**—Rollover is externally handled, if at all. Timestamp values use the local time zone.
- **CONTINUOUS_UTC**—Same as CONTINUOUS, but with UTC timestamp.
- **DAILY_ROLLOVER**—The log file is closed at midnight local time, compressed with gzip, and a new log file is started. Timestamp values use the local time zone.
- **DAILY_ROLLOVER_UTC**—Same as DAILY_ROLLOVER, but with UTC timestamp.
- **JSON_ROLLOVER**—Logs are formatted using JSON. The log file is closed at midnight local time, compressed with gzip, and a new log file is started. Always uses ISO 8601 UTC timestamps.
- **JSON**—Logs are emitted as JSON to stdout only (suitable for Docker). Always uses ISO 8601 UTC timestamps.

*Note • The Docker-friendly JSON logging style is not supported when the local license server is run as a Windows or Linux service (because it only writes to stdout). If set, the style will be changed to DAILY_ROLLOVER.*

| https-in: parameters | Parameters used to access the “server” certificate from a certificate authority (CA) to support incoming HTTPS from client devices. For details, see Incoming HTTPS in the More About License Server Functionality chapter. These parameters replace the “server” configuration file used for HTTPS in previous FlexNet Embedded license server versions. To use a “server” configuration file from a previous license server version (instead of these parameters), specify the path to the “server” configuration file in the HTTPS_SERVER_CONFIG setting in the flexnetls.settings file. For more information, see Edit “flexnetls.settings”. |

*Note • The “server” configuration file is being deprecated and will be removed in a future release.*
Post-Installation Configuration on Linux

The following sections describe how to edit current settings in the `flexnetls.conf` and `local-configuration.yaml` files after the license server service is installed:

- Edit Settings in "flexnetls.conf"
- Edit the DEFINES Setting
- Enable systemd Logging
Edit Settings in “flexnetls.conf”

You can use either of the following methods to update configuration settings in the /etc/systemd/system/flexnetls-producer_name.service.d/flexnetls.conf unit file:

- Edit the .conf File Manually
- Re-install the Service with Command-line Options to Update Settings

Edit the .conf File Manually

Use the following procedure to manually edit the contents of the flexnetls.conf unit file.

**Task** To edit the flexnetls.conf file manually

1. Stop the license server service:
   
   ```bash
   sudo systemctl stop flexnetls-producer_name
   ```

2. As a root or sudo user, edit the /etc/systemd/system/flexnetls-producer_name.service.d/flexnetls.conf file in a text editor (for example, gedit or GNU nano).

3. Once the edits are complete, run the following command to re-read all the systemd unit files to capture the configuration changes:
   
   ```bash
   sudo systemctl daemon-reload
   ```

4. Start the license server service:
   
   ```bash
   sudo systemctl start flexnetls-producer_name
   ```

Re-install the Service with Command-line Options to Update Settings

The following command reinstalls the license server service, using command-line options to update specific settings in the flexnetls.conf file. For a description of the configuration settings that can be updated from the install-script command line, see Configuration Values Editable from the Command Line.

```bash
sudo ./install-systemd.sh --user flexnetls1 --group flexnetls1
```

If you include the --overwrite option in this command, the re-installation generates a new .conf file and a new .yaml file with the specified updates; any changes you previously made to these files are lost.

Edit the DEFINES Setting

The DEFINES setting in the flexnetls.conf unit file is the same as EXTRA_SYSPROPERTIES in the flexnetls.settings file used in the legacy (SysV init) license-server service installations. By design, this setting has no corresponding command-line option to update its value; therefore, you need to edit the setting manually in the configuration file. For instructions on manually editing this file, see Edit the .conf File Manually.
This setting is used to define one or more system properties (each in `-Dkey=value` format) that the license server needs to pass to the Java Runtime system to support certain network functionality, such as an HTTP proxy.

For example, if you plan to have the license server communicate with the back office through an HTTP proxy, use this setting to identify the proxy parameters needed to configure the server. (For details, see Proxy Support for Communication with the Back Office in the More About License Server Functionality chapter.) The following shows example proxy parameters listed as `-D` system properties for this setting:

```
DEFINES="-Dhttp.proxyHost=10.90.3.133 -Dhttp.proxyPort=3128 -Dhttp.proxyUser=user1a -Dhttp.proxyPassword=user1apwd35"
```

By default, this setting has no properties defined.

**Enable systemd Logging**

By default, the standard Syslog system for logging is used when the license server is installed as a systemd service. However, you can switch to the systemd integrated logging system, called *systemd journal* (see More About systemd Logging), once the service is installed.

To enable systemd journal, manually edit the `flexnetls.conf` unit file (using the process described in Edit the .conf File Manually) to update the `OPTIONS=` line to include `"--systemd-logging"`:

```
OPTIONS="--systemd-logging"
```

If you want to switch back to Syslog, simply remove `"--systemd-logging"` from `OPTIONS=`, as shown:

```
OPTIONS=
```

---

**Caution •** The license server should not use a logging level above INFO when systemd journal is enabled. A level above INFO can have a negative impact on license server performance.

**Edit “local-configuration.yaml” (Linux)**

Use the following procedure to update the `local-configuration.yaml` file generated in the `/opt/flexnetls/producer` directory during installation. This file contains optional settings for the license server service that are, by default, disabled.

**Edit Settings in the “local-configuration.yaml” file**

Use the following procedure to enable, edit, or disable settings in `local-configuration.yaml` file.

---

**Task To edit settings in the “local-configuration.yaml” file**

1. As a root or sudo user, edit the `local-configuration.yaml` file in an editor such as `gedit`. Uncomment (or comment out) lines and edit setting values as needed. For a description of the settings, see the next section.

2. Run the following command to stop and restart the service to apply the configuration changes:

```
sudo systemctl restart flexnetls-producer_name
```
**Settings in the “local-configuration.yaml” File**

The following settings can be edited directly in the `configuration-local.yaml` file. Note that a space after the colon is required for all entries.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>port: port</code></td>
<td>The override for the listening port used by the license server, as described in <a href="#">Configuration Values Editable from the Command Line</a>. You can edit this override value as needed. If no override is specified (that is, the setting is commented out), the service uses the default port 7070 (or the value defined in <code>producer-settings.xml</code>).</td>
</tr>
<tr>
<td><code>active-hostid: value/type</code></td>
<td>The hostid to use for the license server. The hostid can only be set using a configuration file if no hostid has yet been specified for the license server. Once a hostid has been set, it can only be changed using the FlexNet License Server Administrator or the FlexNet License Server Manager. The syntax is <code>value/type</code> (for example, <code>720014f5df0/Ethernet</code>). If no value is specified for this setting and no active hostid has yet been set for the license server, the license server uses, by default, the first available Ethernet address on the machine. If using a dongle ID or a hostid other than the first available Ethernet address, specify it here. For more information about hostids, see <a href="#">Understanding Hostids</a>.</td>
</tr>
</tbody>
</table>

**Important** • It is not recommended to change the hostid of a license server that has licenses mapped in the back office (FlexNet Operations). If the hostid is changed to a value that is different to that specified for the license server in the back office, any existing licenses mapped to the license server that is locked to the old hostid in the back office will be orphaned.

To prevent this from happening, it is best practice to return the license server in the back office. During the return operation, the producer can transfer the licenses to a different device; this can be the same machine with the desired hostid. After the transfer, wait for the license server to synchronize with the back office (server synchronization occurs based on synchronization policies or on-demand).
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-hostid: value/type</td>
<td>The hostid of the back-up license server in a failover configuration with the current license server (as the main server), if the back-up server is “unknown”—that is, not registered—in FlexNet Operations. The syntax is value/type (for example, 7200014f5df0/Ethernet). For more information about hostids, see Understanding Hostids. This setting adds the back-up server's hostid to the capability request sent by the current license server to the back office. The back office then automatically registers the back-up server in a failover configuration with the main server. This process saves the extra step of having to manually register the failover pair in FlexNet Operations. For more information, see License Server Failover in the More About License Server Functionality. Make sure the back-up server's hostid is the same type (for example, “ethernet”) as the hostid type of the current (main) license server.</td>
</tr>
<tr>
<td>extended-suffix: suffix</td>
<td>The suffix used for the Extended Host ID feature. Contact the software producer for details.</td>
</tr>
<tr>
<td>server-alias: alias</td>
<td>A user-defined name (sometimes called host name) for the license server. This name is added to server's capability requests to the back office, where it is then saved and used to identify the license server in the FlexNet Operations Producer and End User portals. One important use for this setting is that the alias can be included in the initial capability request sent at server registration, providing a helpful name by which users can identify the new server in the portals. If no alias is sent at registration, the server is identified by its hostid.</td>
</tr>
<tr>
<td>jni-helper-directory: path</td>
<td>If the default location for the JNI shared library (/tmp) is not suitable (for example, because execute permission is required), use this setting to supply an alternative directory path. The directory has to exist and be writable by the server.</td>
</tr>
<tr>
<td>logging-threshold: level</td>
<td>The override for the lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG, as described in Configuration Values Editable from the Command Line. You can edit this override value as needed. If no override is specified (that is, the setting is commented out), the service uses the default level INFO (or the value defined in producer-settings.xml). For more information about logging levels, see section Logging Policies in Reference: License Server Policy Settings.</td>
</tr>
</tbody>
</table>
Table 2-8 • Settings in the “local-configuration.yaml” File (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loggingStyle: style</td>
<td>The logging style determines rollover, JSON formatting and timestamp behaviour. If no logging style is specified, the default CONTINUOUS is used.</td>
</tr>
<tr>
<td></td>
<td><strong>Logging style values:</strong></td>
</tr>
<tr>
<td>CONTINUOUS</td>
<td>Rollover is externally handled, if at all. Timestamp values use the local time zone.</td>
</tr>
<tr>
<td>CONTINUOUS_UTC</td>
<td>Same as CONTINUOUS, but with UTC timestamp.</td>
</tr>
<tr>
<td>DAILY_ROLLOVER</td>
<td>The log file is closed at midnight local time, compressed with gzip, and a new log file is started. Timestamp values use the local time zone.</td>
</tr>
<tr>
<td>DAILY_ROLLOVER_UTC</td>
<td>Same as DAILY_ROLLOVER, but with UTC timestamp.</td>
</tr>
<tr>
<td>JSON_ROLLOVER</td>
<td>Logs are formatted using JSON. The log file is closed at midnight local time, compressed with gzip, and a new log file is started. Always uses ISO 8601 UTC timestamps.</td>
</tr>
<tr>
<td>JSON</td>
<td>Logs are emitted as JSON to stdout only (suitable for Docker). Always uses ISO 8601 UTC timestamps.</td>
</tr>
<tr>
<td></td>
<td><strong>Note •</strong> The Docker-friendly JSON logging style is not supported when the local license server is run as a Windows or Linux service (because it only writes to stdout). If set, the style will be changed to DAILY_ROLLOVER.</td>
</tr>
</tbody>
</table>

| https-in: parameters  | Parameters used to access the “server” certificate from a certificate authority (CA) to support incoming HTTPS from client devices. For details, see Incoming HTTPS in the More About License Server Functionality chapter. These parameters replace the “server” configuration file used for HTTPS in previous FlexNet Embedded license server versions. To use a “server” configuration file from a previous license server version (instead of these parameters), you need to set up a command-line option to specify the file path and name. For more information, see Define Command-Line Options for HTTPS Configuration Files. |
|                       | **Note •** The “server” configuration file is being deprecated and will be removed in a future release.                                                                                                                                                                                                                                                                                                                                                                           |
Define Command-Line Options for HTTPS Configuration Files

The `local-configuration.yaml` includes the `https-in` and `https-out` options used to define the parameters for the license server’s HTTPS communications with FlexNet Embedded clients and FlexNet Operations. These two options replace the “server” and “client” HTTPS configuration files used in previous license server versions. However, you can still use these configuration files (instead of the `https-in` and `https-out` options) by providing command-line options in the `flexnetls.conf` unit file to identify the paths to these files.
To identify the path for an HTTPS configuration file

Use the instructions in Edit the .conf File Manually to edit (or add) the Environment="OPTIONS=" line in the flexnetls.conf unit file to specify one or both command-line options:

- To specify the command-line option that identifies the path and name for the "server" configuration file, enter the following:
  Environment="OPTIONS=--https-server-configuration filePathAndName"

- To specify the command-line option that identifies the path and name for the "client" configuration file path, enter the following:
  Environment="OPTIONS=--https-client-configuration filePathAndName"

- To specify both command-line options, enter the following:
  Environment="OPTIONS=--https-server-configuration filePathAndName
  --https-client-configuration filePathAndName"

More About systemd Logging

By default, the standard Syslog system for logging is enabled for the license server service once the service is installed. However, systemd has its own integrated logging system called systemd journal. To switch to systemd journaling, you must edit the .conf file once the service installed. See Enable systemd Logging for details.

When using systemd journaling, the license server requires no external configuration as it does when using Syslog (which requires a custom Log4J configuration). Additionally, when the journal is enabled, the status output for the license server includes entries from the journal for the administrator's convenience. However, a drawback in using systemd journaling is that a logging level above INFO can have a negative impact on license server performance.

You manage the systemd journal using journalctl commands. The following sections provide an overview of how to perform basic management tasks:

- View Journal Entries
- Configure the Journal

Prerequisite

Run journalctl commands as a root or sudo user.

View Journal Entries

Use the following journalctl command variations to view the entries in the systemd journal.

Task To view the systemd journal entries for the license server service

- To view all entries in the journal, starting with the oldest entry, run this command:
  sudo journalctl -u flexnetls-producer_name.service
To filter the journal entries, use options similar to this (showing the option --since to view entries on or later than the specified time):

```bash
sudo journalctl -u flexnetls-producer_name.service --since 14:00:00
```

To show entries at a given log level or lower, use -p and the log level (such as err):

```bash
sudo journalctl -u flexnetls-producer_name.service -p err
```

For a complete list of `journalctl` options, go to this website:

https://www.freedesktop.org/software/systemd/man/journalctl.html#

### Configure the Journal

For instructions on how to configure journal behavior, go to the following website:

https://www.freedesktop.org/software/systemd/man/journald.conf.html

### Understanding Hostids

FlexNet Embedded uses system identifiers, called hostids, for identification. When the FlexNet Embedded local license server communicates with the back office, FlexNet Operations, to obtain its pool of licenses, the back office uses the server's hostid to identify that particular server. Similarly, when the client requests a license from the license server, the client includes a hostid in the request to which the license server binds the licenses sent in the response.

This section focuses on license server hostids. For more information about client hostids, refer to the user guide for the respective client kit.

### Hostid Types

FlexNet Embedded supports the following standard hostid types. If no hostid is specified, the license server looks up the hostids in the native DLL component of the local license server, in the order in which they are listed in the following table:

<table>
<thead>
<tr>
<th>Hostid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>producer-defined</td>
<td>In addition to the standard hostid types (ethernet, flexid-9, flexid-10, and vm_uuid), the software producer can optionally specify a producer-defined hostid value to identify the server to the back office. This hostid is defined in producer_settings.xml.</td>
</tr>
<tr>
<td>ethernet</td>
<td>The unique identifier assigned to each network interface controller (NIC).</td>
</tr>
<tr>
<td>flexid9</td>
<td>The hostid of the Aladdin dongle attached to the server.</td>
</tr>
<tr>
<td>flexid10</td>
<td>The hostid of the Wibu-Systems dongle attached to the server.</td>
</tr>
</tbody>
</table>
Specifying a Server Hostid

If the license server is run with back-office support, the producer must specify a hostid in the back office during license server creation. The workflow for specifying a server hostid depends on whether a local license server or a CLS instance is used. In implementations with a local license server, you as the license server administrator are required to identify the hostid and communicate it to the producer. When using a CLS instance, the server hostid is based on the server instance ID (generated when the instance is created). During synchronization from the back office (which occurs based on synchronization policies or on-demand), the back office sends the hostid along with other information to the license server.

Instead of waiting for the synchronization to occur, you can also specify the license server’s hostid manually. It is vital that the hostid that you set on the local license server matches the hostid that is specified in the back office. If you change the hostid on the license server to a value that is different to that specified in the back office, any licenses already mapped to the license server that is locked to the old hostid in the back office will be orphaned.

If you need to change the local license server’s hostid when it already has licenses mapped in the back office, it is best practice to return the license server in the back office. During the return operation, the producer can transfer the licenses to a different device; this can be the same machine with the desired hostid. After the transfer, wait for the license server to synchronize with the back office.

Selecting a Hostid

Specifying a server hostid is optional. If no hostid is set, the license server looks up the hostids in the native DLL component of the local license server, in the order in which they are listed in the table in section Understanding Hostids.

Specifying a server hostid is optional. However, it is important to know that the ethernet hostid might change, depending on the adapter that is in use. Consider the following scenario: A laptop plugged into a docking station uses the ethernet address of the docking station; however, when it is disconnected from the docking station, the ethernet address is no longer available. A wireless adapter also has an ethernet address and this address is not available when either the wireless adapter is removed from the machine or when the wireless adapter is disabled, but still physically attached to the machine.

In cases as described above, it is advisable to specify the hostid manually to ensure continuous operation.

Manually Specifying a Hostid

You can specify the server hostid (a value/type pair) in one of the following ways:

- Using the FlexNet License Server Administrator using the following option:
  
  -hostid -setactive hostIdValue hostIdType

  See Table 3-3 in Using the FlexNet License Server Administrator Command-line Tool.

- Using the FlexNet License Server Manager—see Properties View in Using the FlexNet License Server Manager.
Using the configuration files (depending on your platform):

- `flexnetls.settings`, setting `ACTIVE_HOSTID` (Windows)—see Table 2-3 in Configure, Install, and Start on Windows.
- `/etc/default/flexnetls-producer_name`, setting `ACTIVE_HOSTID` (when using SysV to run the license server as a Linux service)
- `configuration-local.yaml`, setting `active-hostid` (when using systemd on Linux)—see Edit "local-configuration.yaml" (Linux).

**Important** • You can only set the hostid using a configuration file if no hostid has yet been specified for the license server. Once a hostid has been set, it can only be changed using the FlexNet License Server Administrator or the FlexNet License Server Manager.

### Retrieving Server Hostids

**Task** • To retrieve a server hostid, do one of the following

- Call the JSON REST endpoint `http://licenseServerHostName/api/1.0/hostids` (or `/hostids/selected`

- Execute `flexnetlsadmin.bat -server licenseServer_baseURL -config` (`flexnetlsadmin.bat` is located in the enterprise directory)

**Important** • (Linux only) To retrieve the VM-UUID for local license servers that run on certain virtual Linux machines and that use the VM-UUID as their hostid, you need to run the FNLicensingService (located in the “service” directory).

### Uninstalling the Local License Server

Use the appropriate procedure to uninstall the FlexNet Embedded local license server service:

- Uninstall on Windows
- Preparing to Use the License Server Manager

### Uninstall on Windows

Use these steps to uninstall the license server service on Windows.
**Chapter 2 Getting Started**

**Uninstalling the Local License Server**

**Task**

**To uninstall the license server service on Windows**

1. As an administrator, open a command prompt and navigate to the license server’s installation directory.

2. Execute the command `flexnetls.bat -stop` to stop the service.

3. Execute the command `flexnetls.bat -uninstall` to uninstall the license server service. (If you attempt to uninstall the service before stopping it, a message appears indicating that you need to stop the service first.)

4. To ensure there are no hanging instances or services, execute `sc delete FNLS-producer_name` as an administrator.

   Either the command will fail with the message The specified service does not exist as an installed service, or it will succeed with the message [SC] DeleteService SUCCESS. Both outcomes indicate that the service is no longer present.

5. Delete the license server component files from the installation folder.

6. Optionally, delete these files (listed here with their default locations):
   - The trusted storage files in `C:\Windows\ServiceProfiles\NetworkService\flexnetls\producer_name` (the .ks, .db, and .0 files)
   - The log files in `C:\Windows\ServiceProfiles\NetworkService\flexnetls\producer_name\logs`

   Trusted storage and log locations are defined by the license server policies `server.trustedStorageDir` and `logging.directory`, respectively, the defaults for which are based on `${bases.dir}`. Depending on the values set for these policies on your server, your trusted storage and log files might be in locations different from those mentioned in this step. See Reference: License Server Policy Settings.

**Uninstall on Linux**

Use these steps to uninstall the license server service on Linux.

**Task**

**To uninstall the license server on Linux**

1. Run the following commands in this order:
   - `sudo systemctl stop flexnetls-producer` (to shut down the service)
   - `sudo systemctl disable flexnetls-producer` (to make the service ineligible to start on system reboot)
   - `sudo rm /etc/systemd/system/flexnetls-producer.service`
   - `sudo rm -r /etc/systemd/system/flexnetls-producer.service.d`

2. Optionally, remove these files (listed here with their default locations):
   - The trusted storage files in `/var/opts/flexnetls/producer_name` (the .ks, .db, and .0 files)
   - The log files in `/var/opts/flexnetls/producer_name/logs`
Preparing to Use the License Server Manager

The FlexNet Embedded license server might include one or both of the following administration tools to help you maintain the server and manage license distribution in your enterprise:

- FlexNet License Server Administrator command-line tool
- FlexNet License Server Manager (which uses a UI)

If you intend to use the FlexNet License Server Manager, you must have an Apache Tomcat server installed. Before launching the License Server Manager for the first time, you are required to perform these additional steps to integrate the license server with Tomcat.

**Task**

**To integrate the license server with Tomcat**

1. Install the Tomcat server if one does not already exist on your system. Tomcat installers are available from the Apache Tomcat website at [http://tomcat.apache.org/](http://tomcat.apache.org/).

   *Note* • On Windows systems, to install Tomcat as a Windows service, follow the instructions provided on the [tomcat.apache.org](http://tomcat.apache.org/) link. For detailed information and other approaches, see the Tomcat documentation.

2. Locate the `flsm.war` file in your license server installation.

3. Copy `flsm.war` to the `webapps` directory in the Tomcat installation.

4. If necessary, change the port on which the License Server Manager tool listens for browser requests. (By default, the License Server Manager listens on port 8080.) Use these steps:
   a. Locate the `server.xml` file in the `conf` directory of the Tomcat installation.
   b. Within `server.xml`, locate the appropriate `connector` element and modify its `port` attribute value to the desired port number.

5. Start up the Tomcat server, and then launch the FlexNet License Server Manager, using the instructions in the [Using the FlexNet License Server Manager](#) chapter.

**Base URL for the License Server**

Administrative tools, such as the FlexNet License Server Administrator command-line tool or a custom administrative tool provided by the producer, require that you input the license server’s base URL to perform operations. While the same basic structure is used for the CLS instance and the local license server URLs, some syntactical differences exist. See the following sections for more information about the base URL:

- Base URL for a CLS Instance
Chapter 2 Getting Started

Base URL for the License Server

- Base URL for a Local License Server
- With Administrative Security Enabled
- Option to Disable Certain Verifications on Server When Using HTTPS
- Exceptions to the Base URL
- “licenseServer_baseURL” as Base URL Designation

Base URL for a CLS Instance

The base URL for a CLS instance, which is provided by the producer, explicitly includes the server's instance ID. The following is the format for the base URL for a CLS instance.

```
https://siteID.compliance.flexnetoperations.com/api/1.0/instances/instanceId
```

An example might be the following (with XYZ10203040 as the instance ID):

```
flexnetlsadmin -server https://ABC.compliance.flexnetoperations.com/api/1.0/instances/XYZ10203040 ...
```

Base URL for a Local License Server

The base URL for a local license server includes the server's host name and port number and uses the tilde “~” character in place of the server's instance ID. The license server name can be either a network name (such as myserver.enterprise.com) or an IP address.

```
http://LicenseServerHostName:port/api/1.0/instances/~
```

An example might be the following:

```
flexnetlsadmin -server http://11.11.111:7070/api/1.0/instances/~ ...
```

With Administrative Security Enabled

When administrative security is enabled on the license server, the use of the HTTPS protocol on the local license server is strongly recommended to protect sensitive data being transmitted during administrative operations. (To use HTTPS, you must first enable it on your license server machine.)

The base URL for the local license server might look like the following.

```
https://11.11.1.111:1443/api/1.0/instances/~
```

The default port for a HTTPS URL is 443, unless one is supplied. The local license server defaults to 1443 to avoid privilege issues on Linux with port numbers less than 1024, but any appropriate port can be used.

To enforce the use of HTTPS when administrative security is enabled, set the security.http.auth.enabled license server policy to `false` in producer-settings.xml. See Administrative Security Policies on the License Server.

Option to Disable Certain Verifications on Server When Using HTTPS

If for some reason, you need to disable certificate checks and hostname verification on the license server when using HTTPS, you can do so using the `-noCertCheck` option in the FlexNet License Server Administrator command-line tool (flexnetlsadmin). See Summary of flexnetlsadmin Commands in the Using the FlexNet License Server Administrator Command-line Tool chapter.
Exceptions to the Base URL
The producer should inform you of any exceptions to the base URL.

“licenseServer_baseURL” as Base URL Designation
In this chapter (and throughout this book), the base URL in commands is represented by the variable licenseServer_baseURL.

Managing Administrative Security on a Local License Server or CLS Instance

FlexNet Embedded provides an administrative security facility on the license server to prevent unauthorized queries and operations on the server. When this security is enabled on your license server, anyone attempting to administer the license server will need to provide a set of authorization credentials before starting administrative operations.

The default license-server administrator account in place when you start the server gives you access to the same full range of administrative functionality that you would have without security enabled, with additional privileges to create and manage other enterprise user accounts. The accounts you create generally have limited administrative privileges, although you can create other license-server administrator accounts.

The administrative security feature is available on both a CLS instance and the local license server.

Note • A default producer account is also created when you start up the license server. This account is used to supply or delete checkout filters (these are selective license filters customizable by the producer), and change configuration values not editable by the default administrator account (for example, lfs.syncTo.enabled).

The following sections provide an overview of managing administrative security on the license server:

• Security Work Flow in the Enterprise
• Secured Functionality on the License Server
• User Roles Defining Administrative Privileges
• Administrative Security Policies on the License Server
• Credential Requirements
• HTTPS Recommended
Security Work Flow in the Enterprise

The following provides the basic work flow for managing license-server administrative security in the enterprise. This flow assumes that the producer has deployed your license server with security already enabled. It also assumes that you have access to a license-server administrator tool, such as the FlexNet License Server Administrator command-line tool, the FlexNet License Server Manager, or a custom administrator tool provided by the producer.

Table 2-10 • Security Work Flow for Administering the License Server

<table>
<thead>
<tr>
<th>Step</th>
<th>Who</th>
<th>Does what</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Producer</td>
<td>Sets up the default license server administrator account and distributes both the default credentials (if needed) and the license server's base URL.</td>
<td>For a CLS instance: The default license-server administrator account is automatically created when the CLS instance is created. For a local license server: The default license-server administrator account generated by the producer.</td>
</tr>
</tbody>
</table>
| 2    | Enterprise license-server administrator | Resets default administrator password | For a CLS instance: Uses Action > Set Password on View Server page in FlexNet Operations End-User Portal. Default credentials are not required.¹ For a local license server: Uses one of the following:  
  - `-authorize` command option (to authenticate default administrator credentials) with `-users` `-edit` options in FlexNet License Server Administrator command-line tool (`flexnetlsadmin`). Default credentials are required.²  
  - Appropriate facility in producer's custom administrator tool. Default credentials are required.⁴ |
### Table 2-10 • Security Work Flow for Administering the License Server (cont.)

<table>
<thead>
<tr>
<th>Step</th>
<th>Who</th>
<th>Does what</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Enterprise license-server administrator</td>
<td>Uses credentials to access license-server administrative functionality</td>
<td>Uses one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• -authorize command option (to authenticate administrator credentials) in FlexNet License Server Administrator command-line tool (flexnetlsadmin).²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Login view in FlexNet License Server Manager (for local license server only).³</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Appropriate facility in producer’s custom administrator tool.⁴</td>
</tr>
<tr>
<td>4</td>
<td>Enterprise license-server administrator</td>
<td>Optionally creates other enterprise user accounts to perform specific administrative tasks (and distributes credentials to those users)</td>
<td>Uses one of the following to create the accounts:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• -authorize command option (to authenticate administrator credentials) with -users -create options in the FlexNet License Server Administrator command-line tool (flexnetlsadmin).²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Appropriate facility in producer’s custom administrator tool.⁴</td>
</tr>
<tr>
<td>5</td>
<td>Enterprise user</td>
<td>Uses credentials to access the specific administrative functionality</td>
<td>Uses one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• -authorize command option (to authenticate user credentials) in the FlexNet License Server Administrator command-line tool (flexnetlsadmin).²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Login view in FlexNet License Server Manager (for local license server only).³</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Appropriate facility in producer’s custom administrator tool.⁴</td>
</tr>
</tbody>
</table>
Secured Functionality on the License Server

When administrative security is enabled on the license server, the following license server operations require credentials:

- Resetting your administrator password
- Creating and managing other enterprise user accounts
- Adding and deleting license reservations
- Setting or resetting license server policies, including administrative security policies
- Suspending and resuming the license server
- Manual initiating a binary exchange between the license server and the back office, such an online or offline activation or synchronization event

Additionally, “read” operations might also require credentials, depending on how administrative security is configured on the license server. Refer to the next section Policy Affecting “read” Security for more information.

For references to the appropriate documentation describing how you provide credentials when using a specific administrator tool, refer to the previous Security Work Flow in the Enterprise section.

Operations Exempt from Administrative Security

The following operations are always exempt from administrative security measures (that is, no credentials are ever needed to perform these operations):

- Binary exchanges between the license server and FlexNet Operations that are initiated by the license server, such capability-polling exchanges and synchronization at set intervals
- Binary exchanges between the license server and license-enabled clients
Policy Affecting “read” Security

When administrative security is enabled on the license server, it uses the `security.anonymous license-server` policy to determine whether users need to provide credentials simply to “read” information—such as current reservations, features, licenses, or status—on the license server. See Administrative Security Policies on the License Server for more information.

User Roles Defining Administrative Privileges

When administrative security is enabled, your default administrator account is assigned ROLE_ADMIN, ROLE_RESERVATIONS, ROLE_DROPCLIENT, and ROLE_READ, giving you full rights to administer the license server. Any user account that you create can be assigned one or more roles of these roles, including ROLE_ADMIN to create another administrator.

Keep in mind that any role can perform capability exchanges and synchronization operations as this functionality is exempt from security measures.

Table 2-11 • User Roles Used to Define Administrative Privileges on the License Server

<table>
<thead>
<tr>
<th>Role</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_READ</td>
<td>Privileges to perform “read” operations (for example, to query features, licenses, reservations, or server status). When no other role is assigned to a user account, ROLE_READ is assigned by default as the only role. Additionally, depending on the administration security configuration, either every account is automatically given “read” rights, or you are required to assign ROLE_READ explicitly to each account to give it “read” rights. See Policy Affecting “read” Security.</td>
</tr>
<tr>
<td>ROLE_RESERVATIONS</td>
<td>Privileges to add and delete reservations.</td>
</tr>
<tr>
<td>ROLE_DROPCLIENT</td>
<td>Privileges to delete client records on the license server.</td>
</tr>
<tr>
<td>ROLE_ADMIN</td>
<td>Administrator privileges to update license server policies (local license server only), create and manage other enterprise user accounts, and perform other administrative tasks, such as suspend or resume the license server.</td>
</tr>
<tr>
<td>ROLE_PRODUCER</td>
<td>Privileges given to a producer account (by convention, the account named “producer”). Required to supply or delete checkout filters (these are selective license filters customizable by the producer), and change configuration values not editable by the administrator account (for example, lfs.syncTo.enabled).</td>
</tr>
</tbody>
</table>

Administrative Security Policies on the License Server

As the license server administrator of a local license server, you can update the following policies that control administrative security:

- `security.enabled` to enable or disable administrative security on the license server. See Enabling Administrative Security on a Local License Server.
The following policies are in effect only when administrative security is enabled:

- `security.anonymous` to determine whether users need to provide credentials simply to “read” information—such as current reservations, features, licenses, or status—on the license server:
  - When `security.anonymous` is set to `true`, user accounts, including administrator accounts, are automatically granted “read” rights (ROLE_READ); no credentials are needed to perform “read” operations. This policy lessens your administrative burden to manage user accounts.
  - When this policy is set to `false`, a user account, including an administrator account, must be explicitly assigned ROLE_READ in order to perform “read” operations. (The exception occurs when no role is assigned to an account, in which case ROLE_READ is assigned as the only role by default.) Credentials are then required to perform any “read” operation. If an account is not authorized for ROLE_READ, no “read” access is given.

- `security.token.duration` to limit token validity. This policy sets a time limit on how long an authorization “token” is in effect before it expires, requiring a user to re-enter credentials. Note the following:
  - This policy is relevant to FlexNet License Server Manager and those custom administrator tools where credentials are entered once and then automatically applied to each subsequent operation requiring authorization. Once the token expires, the user must re-enter credentials.
  - The policy is not relevant for tools like the FlexNet License Server Administrator command-line tool, where users must manually re-enter credentials every time they perform an operation requiring authorization.

- `security.ip.whitelist` to grant to one or more secure machines (whose IP addresses you specify for this policy) unrestricted access to the license server administrative interface. This access is helpful in cases when you forget your credentials or when someone needs the convenience of accessing administrative functionality on the license server without having to provide credentials.

- `security.http.auth.enabled` to control whether HTTPS is required to access the license server to perform secured operations (that is, those requiring authorization):
  - When set to `true`, this policy allows the use of the HTTPS or HTTP protocol to perform secured operations.
  - When set to `false`, the policy enforces the use of the HTTPS protocol to perform secured operations. An error is generated when HTTP is used.

This policy has no effect on those operations exempt from license-server security measures, as described in Operations Exempt from Administrative Security.

For more information about these policies, see the following:

- For details about the policies, refer Reference: License Server Policy Settings.
- For instructions to update the policies, see either the Using the FlexNet License Server Administrator Command-line Tool or the Using the FlexNet License Server Manager chapter (or the instructions provided by the producer).

### Enabling Administrative Security on a Local License Server

If the producer deployed a local license server with administrative security disabled, you can enable it as long as the producer has provided you with administrator credentials. You can also disable security on a currently secured license server.
Managing Administrative Security on a Local License Server or CLS Instance

To enable administrative security on the local license server

1. Obtain default administrator credentials from the producer if you do not already have them.

2. Start up the license server. When administrative security is disabled, you have full privileges to administer the license server (as though you have an account with ROLE_ADMIN, ROLE_RESERVATIONS, ROLE_DROPCLIENT, and ROLE_READ).

3. Use your license server administrator tool to change the security.enabled policy to true:
   - For the FlexNet License Server Administrator command line tool (flexnetlsadmin), see Managing License Server Policy Settings.
   - For the FlexNet License Server Manager, see Properties View.
   - For the producer's custom administration tool, refer to the producer's instructions.

4. Once administrative security is enabled on the license server, reset your default administrator password, providing your default administrator credentials to authorize your access to this operation:
   - For the FlexNet License Server Administrator command-line tool, see Reset Your Administrator Password.
   - For the producer's custom administration tool, refer to the producer's instructions.

   This operation is currently not available in the FlexNet License Server Manager.

5. From this point on, provide your credentials to perform any secured operation:
   - For the FlexNet License Server Administrator command-line tool, see Use Credentials to Access Operations.
   - For the FlexNet License Server Manager, see Providing Credentials on a Secured License Server.
   - For the producer's custom administration tool, refer to the producer's instructions.

To disable administrative security on the local license server

Using your credentials, access your administrator tool to change the security.enabled policy to false.

Credential Requirements

The following lists the requirements for the user name and password needed for each enterprise user account you create.

User Name

The user name (up to 64 characters) is case-sensitive but requires no special characters.

User Password

The password for a user account is case-sensitive and must meet the following criteria:

- At least 8 characters (with a maximum of 64 characters)
• At least one digit
• At least one upper-case character
• At least one special character (for example, ^ * $ - + ? _ & = ! % { } / # @ and more)
• No whitespace

HTTPS Recommended

When administrative security is enabled on the local license server, the use of HTTPS is strongly recommended to protect license server data being transmitted during administrative operations. (To use the HTTPS protocol, you must enable it on your license server machine.) For more information, see Base URL for the License Server.

Optionally, you can use the security.http.auth.enabled license server policy to enforce the use of HTTPS when administrative security is enabled. See Administrative Security Policies on the License Server for details.

Next Steps

Once the license server is installed, configured, and running, the next steps include:

• **Step 1: Provisioning licenses on the license server.** A purchased pool of product licenses needs to be activated on the license server so that it can then distribute these licenses as needed to client devices running the product.

  If license rights are mapped to the license server in the back office, these are automatically activated on the license server when the server starts or when capability polling occurs. However, you can also activate licenses using rights IDs obtained from the producer. To send capability requests containing specific rights IDs, follow the instructions provided by the producer, or do one of the following:

  • If you are using the FlexNet License Server Administrator command-line tool, see Activating License Rights on the Server in the chapter Using the FlexNet License Server Administrator Command-line Tool.

  • If you are using the FlexNet License Server Manager, refer to Offline Server Updates View in the chapter Using the FlexNet License Server Manager to perform an offline activation.

  **Important** • *If the license server has not been provisioned with any features, it responds to client requests with a 503 error with the error message “Server instance <instanceID> is not ready”. This is expected behavior because the server is not ready in this state.*

  • **Step 2: Administering the license server in general.** Depending on the administrator tool you are using, refer the appropriate chapter for instructions on performing administrative tasks:

    • Using the FlexNet License Server Administrator Command-line Tool

    • Using the FlexNet License Server Manager
Using the FlexNet License Server Administrator Command-line Tool

This chapter describes how to use the FlexNet License Server Administrator command-line tool to manage the license server and its license distribution. The chapter includes following sections:

- Before You Start
- Using the Command-line Tool to Manage Administrative Security
- General Server Maintenance
- Activating License Rights on the Server
- Viewing Features Installed on the License Server
- Managing Feature Partitions
- Managing Reservations
- Managing License Server Policy Settings
- Monitoring License Distribution to Clients
- Viewing Current Binding Status
- Summary of flexnetlsadmin Commands

Before You Start

The FlexNet License Server Administrator command-line tool is run from the command line, using the flexnetlsadmin script or batch file (in Windows, flexnetlsadmin.bat). The general syntax for executing flexnetlsadmin commands is the following:

```
flexnetlsadmin -server licenseServer_baseURL [-authorize accountName accountPassword]
-command_option [command_option_arguments] [-command_option...]
```

The general syntax for flexnetlsadmin commands uses the license server’s base URL, indicated by `licenseServer_baseURL` in the command. For more information about this URL, see Base URL for the License Server in Getting Started.
A reference of all available `flexnetlsadmin` commands is available at the end of this chapter (see **Summary of `flexnetlsadmin` Commands**). You can also run `flexnetlsadmin` with the `-help` option to display brief descriptions of all the commands:

```
flexnetlsadmin -help
```

The following sections describe additional information you should know before using this tool:

- License Server URL Designation
- About the Example Commands

## License Server URL Designation

The license server's base URL, as described in **Base URL for the License Server** in the Getting Started chapter, is required for each `flexnetlsadmin` command. To identify this URL in the `flexnetlsadmin` commands, do one of the following:

- Use the `-server` Option
- Use an Environment Variable

### Use the `-server` Option

To specify the license server URL explicitly in each `flexnetlsadmin` command, use the `-server` option. Insert it as the first option in the command.

The following excerpt from a sample command shows the `-server` option designating a base URL for a CLS instance:

```
flexnetlsadmin -server https://ABC.compliance.flexnetoperations.com/api/1.0/instances/XYZ10203040 ...
```

This excerpt from a sample command shows the `-server` option designating a base URL command for a local license server:

```
flexnetlsadmin -server http://11.11.111:7070/api/1.0/instances/~ ...
```

### Use an Environment Variable

As an alternative to explicitly providing the license server's base URL every time you enter a `flexnetlsadmin` command, you can set the base URL as the environment variable `FLEXNETLS_BASEURL`. The following shows an example of the variable (set in this case for a local license server):

```
FLEXNETLS_BASEURL=http://11.11.111:7070/api/1.0/instances/~
```

See your Windows or Linux documentation for instructions on how to set this variable.

## About the Example Commands

For sake of consistency, the example `flexnetlsadmin` commands used in this chapter follow these conventions:

- For simplicity, the commands use the Linux format of the `flexnetlsadmin` command (for example, they use `flexnetlsadmin`, not `flexnetlsadmin.bat`).
• The -server option used to identify the license server’s base URL is included in each command (although you can set it as an environment variable, as described in License Server URL Designation, to avoid having to include it each time you issue a command). The example commands use the designation licenseServer_baseURL for the base URL, as described in Base URL for the License Server in Getting Started.

You will need to modify the example commands as needed for your environment.

Using the Command-line Tool to Manage Administrative Security

When administrative security is enabled on your license server, anyone attempting to administer the license server will need to provide a set of authorization credentials before starting administrative operations. When the license server is started, your default administrator account is assigned ROLE_ADMIN, ROLE_RESERVATIONS, ROLE_DROPCLIENT, and ROLE_READ. These roles give you a full range of privileges to administer the license server, including authorization to create and manage other enterprise user accounts that have limited administrative privileges. You can also create other license-server administrator accounts.

For an overview of the administrative security facility, including a description of the available roles, see Managing Administrative Security on a Local License Server or CLS Instance in the Getting Started chapter.

The following sections describe how to use the FlexNet License Server Administrator command-line tool to manage administrative security on the license server:

• Base URL for the Secured License Server
• Reset Your Administrator Password
• Determine Administrative Security Policies
• Create and Manage Other Enterprise User Accounts
• Use Credentials to Access Operations

Base URL for the Secured License Server

When administrative security is enabled on the license server, the use of the HTTPS protocol is strongly recommended to protect license server data being transmitted during administrative operations. For more information about the use of this protocol in the base URL for the license server (represented by licenseServer_baseURL in the example commands), see Base URL for the License Server.

Reset Your Administrator Password

The producer will provide the default credentials (for the default license-server administrator account) needed to authorize your use of administrative functionality when security is enabled on the license server. Once your license server starts up, your first task is to reset the default administrator password. Use the -authorize options (to provide your default credentials that authorize you for the task) with the -users -edit options (to change your password).
### Task

To reset your administrator password

1. Enter a command similar to this:
   ```
   flexnetlsadmin -server licenseServer_baseURL -authorize defaultAdminName {defaultAdminPassword|-passwordConsoleInput} -users -edit admin {newAdminPassword!|-passwordConsoleInput}
   ```
   
   The command requires the following:
   
   - **defaultAdminName**—The case-sensitive name provided by the producer for the default administrator account. ("admin" is typically the default.)
   
   - **defaultAdminPassword**—The case-sensitive password provided by the producer for the default administrator account. You can provide this password directly in the command; or, to avoid exposing the password as a command-line argument, you can use the `-passwordConsoleInput` option, which then issues a prompt to enter the password as console input.

   **Note** • On Linux platforms, if you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

   - **newAdminPassword!**—The new case-sensitive password that you want to define for the default administrator account. To avoid exposing the password as a command-line argument, you can use the `-passwordConsoleInput` option, which then issues a prompt to enter the password as console input. This password must meet the criteria specified in Credential Requirements.

   **Note** • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

2. Once this password is reset, use your updated credentials with the `-authorize` option to access secured administrative functionality on the license server. See Use Credentials to Access Operations.

   If you want to change your administrator name, create a new administrator account. See Creating Another Administrator Account in the next section, Create and Manage Other Enterprise User Accounts.

### Determine Administrative Security Policies

For a local license server, you can set policies that configure administrative security. For more information about the security policies, see Administrative Security Policies on the License Server in the Getting Started chapter. For instructions on updating the policies, see Override Policy Settings in this chapter.
Create and Manage Other Enterprise User Accounts

As a license server administrator, you can use your credentials to create other user accounts—either for enterprise users or for another administrator. Refer to the following sections:

- Creating an Enterprise User Account
- Creating Another Administrator Account
- Edit a User or Administrator Account
- Deleting a User or Administrator Account
- Viewing All User Accounts

Creating an Enterprise User Account

To create an enterprise user account, use the -authorize option (to provide your credentials that authorize you for the task) with the -users -create options to create the account. (Creating a license-server administrator account is described in the next section, Creating Another Administrator Account.)

Task To create an enterprise user account

1. Enter a command similar to this:

   ```
   flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword} -passwordConsoleInput -users -create userName {userPassword} -passwordConsoleInput role(s)
   ```

   The command requires the following:

   - `yourAdminName yourAdminPassword`—Your administrator credentials.

   Note that you can provide your administrator password directly in the command; or, to avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

   ```
   Note  (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.
   ```

   - `userName`—The case-sensitive name to identify the enterprise user’s account.

   - `userPassword`—The case-sensitive password that you are defining for the enterprise user’s account. This password must meet the criteria specified in Credential Requirements. To avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

   ```
   Note  (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is
Creating Another Administrator Account

To create another license-server administrator account, use the -authorize option (to provide your credentials that authorize you for the task) with the -users -create options to create the account.

Task  To create another administrator account for the license server

1. Enter a command similar to this:

   `flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword|-passwordConsoleInput} -users -create newAdminName {newAdminPassword|-passwordConsoleInput} ROLE_ADMIN[+additional_roles]`

The command requires the following:

- `yourAdminName yourAdminPassword`—Your administrator credentials.
  
  Note that you can provide your administrator password directly in the command; or, to avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

- `newAdminName`—The case-sensitive name with which you are identifying the new license-server administrator account.

- `newAdminPassword`—The case-sensitive password that you are defining for the new license-server administrator account. This password must meet the criteria specified in Credential Requirements. To avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.
Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

- ROLE_ADMIN[+additional_roles]—ROLE_ADMIN and optional other roles—such as ROLE_READ, ROLE_RESERVATIONS, or ROLE_DROPCLIENT—that define this administrator account. If multiple roles are specified, insert a plus sign (+) between each role, using no spaces (for example, ROLE_ADMIN+ROLE_READ+ROLE_RESERVATIONS). For more information about roles, see User Roles Defining Administrative Privileges in the Getting Started chapter.

2. View the currently available user accounts to verify that this account has been properly added. See Viewing All User Accounts.

3. Provide the new administrator with the credentials needed to perform administrative operations. Advise the administrator that both the user name and password are case-sensitive.

Edit a User or Administrator Account

To change the password or the roles for an account belonging to either an enterprise user or a license server administrator, use the -authorize option (to provide your credentials that authorize you for the task) with the -users -edit options to edit the account.

Task To update a user account

1. Enter a command similar to this:

   flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword|passwordConsoleInput} -users -edit accountName {currentPassword|newPassword|passwordConsoleInput} newRole(s)!

   The command requires the following:

   - yourAdminName yourAdminPassword—Your administrator credentials.

     Note that you can provide your administrator password directly in the command; or, to avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

   - accountName—The case-sensitive user name for the account.

   - currentPassword|newPassword|—The current account password; or, if you are changing the password, the new password, which must meet the criteria specified in Credential Requirements. To avoid
exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

newRole(s) — If you modifying the account’s roles, the new role or roles defining the type of administrative privileges for the account. If multiple roles are specified, insert a plus sign (+) between each role, using no spaces (for example, ROLE_READ+ROLE_RESERVATIONS). For more information about roles, see User Roles Defining Administrative Privileges in the Getting Started chapter.

If no role is specified, the accounts current role assignment remains in effect.

2. If you changed the account’s role assignment, view the current user accounts to verify that the account has been assigned the new roles. See Viewing All User Accounts.

3. Distribute new credentials (if updated) to the enterprise user so that the user can access the appropriate administrative operations.

Deleting a User or Administrator Account

To delete an account belonging to an enterprise user or an administrator, use the -authorize option (to provide your credentials that authorize you for the task) with the -users -delete options to delete the account.

Task To delete a user account

1. Enter a command similar to this:

   flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword} -passwordConsoleInput} -users -delete accountName

2. The command requires the following:

   • yourAdminName yourAdminPassword — Your administrator credentials.

   Note that you can provide your administrator password directly in the command; or, to avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

   Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

   • accountName — The case-sensitive user (or administrator) name for the account.

3. View the currently available user accounts to verify that the account has been deleted. See Viewing All User Accounts.
Viewing All User Accounts

To view all user accounts currently available on the license server, use the -authorize option (to provide your credentials that authorize you for the task) with the -users option.

Task

To view all user accounts

Enter a command similar to this, where yourAdminName and yourAdminPassword indicate your administrator credentials:

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword} -passwordConsoleInput -users
```

Note that you can provide your administrator password directly in the command; or, to avoid exposing this password as a command-line argument, you can use the -passwordConsoleInput option, which then prompts you to enter your password as console input.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

The output shows each account by user name and lists its roles.

Use Credentials to Access Operations

When administrative security is enabled on the license server, the license server administrator and enterprise users must use their credentials with the -authorize option to perform operations for which they are authorized. For example, if an enterprise user account is assigned ROLE_RESERVATIONS, the user would provide credentials in a command typical of the following:

```
flexnetlsadmin -server licenseServer_baseURL -authorize accountName {accountPassword} -passwordConsoleInput -reservations -delete...
```

The -authorize command requires these credentials:

- **accountName**—The case-sensitive name for the account attempting to perform the operation.
- **accountPassword**—The case-sensitive password for the account.

Note that users can provide their account password directly in the command; or, to avoid exposing this password as a command-line argument, they can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.
Attempts to Perform Non-Authorized Operations

If a user attempts to perform an operation that requires authorization, but has not provided credentials or is not authorized to perform the operation, an “access denied” message is displayed. The user must either enter the credentials or contact you about obtaining appropriate credentials.

General Server Maintenance

To perform general maintenance on your FlexNet Embedded local license server, use the FlexNet License Server Administrator command-line tool to perform the following:

- Obtain the License Server Status
- Show Version Information for the Administrator Command-line Tool
- Show General Configuration Information for the License Server
- Suspend or Resume the License Server

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.

Obtain the License Server Status

To confirm that the license server is running, you can use the -status command to perform a status check. This check tells you whether the server is active and lists version information. The output also displays the URLs for the back-office server and the back-up license server (if your environment is configured for license server failover).

Note • The server version information is useful should you need to report any incidents to Revenera Technical Support.

Task To check the license server status

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -status
```

Status information is displayed for your license server:

- Version : 2020.01.0
- Build Version : 262400
- Server : https://localhost:7070/api/1.0/instances/ ~
- State : Ready
- Backup Server : Not configured
- BackOffice Server : http://localhost:8080/request
- Records Pending Sync : 42
- Last Sync : 7d 4h38m28s
**Show Version Information for the Administrator Command-line Tool**

Use the `-version` command to display the version information for the currently installed FlexNet License Server Administrator command-line tool.

**Note** • Version information for the License Server Administrator command-line tool is useful should you need to report any incidents to Revenera Technical Support.

**Task**  
To display version information for the License Server Administrator command-line tool

Enter a command similar to this:

```
flexnetlsadmin -version
```

Version information for the License Server Administrator command-line tool is displayed:

```
Version                 2020.01.0
Build Version           262400
```

**Show General Configuration Information for the License Server**

Use the `-config -filter general` syntax to display the information about the license server's configuration, including the server’s IP address, host name, hostid (binding ID), and port number and the software producer’s name.

**Task**  
To display license server configuration information

Enter command syntax similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -filter general
```
The output lists current configuration properties for the license server. For each property, its corresponding producer-setting name is provided in parentheses. (For information about producer settings for your license server, see Reference: License Server Policy Settings.)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.uuid</td>
<td>Host UUID that uniquely identifies this server instance in the back office</td>
</tr>
<tr>
<td>server.instanceId</td>
<td>Instance ID that uniquely identifies this server instance in the REST API</td>
</tr>
<tr>
<td>server.tenantId</td>
<td>Tenant for which this server will serve licenses</td>
</tr>
<tr>
<td>server.enterpriseId</td>
<td>Enterprise to which this server belongs</td>
</tr>
<tr>
<td>server.siteId</td>
<td>Location of this server</td>
</tr>
<tr>
<td>server.trustedStorageDir</td>
<td>Default directory for the trusted storage ${base.dir}</td>
</tr>
<tr>
<td>server.log4jConfiguration</td>
<td>External override file for Log4J configuration</td>
</tr>
<tr>
<td>server.accessLogPattern</td>
<td>Name format for request log access_yyyy_mm_dd.request.log</td>
</tr>
<tr>
<td>server.publisherDefinedHostId.policy</td>
<td>Custom hostid for the license server DISABLED</td>
</tr>
<tr>
<td>server.extendedHostId.enabled</td>
<td>Extended hostids for the license server true</td>
</tr>
<tr>
<td>server.forceTSResetAllowed</td>
<td>Trusted storage can be reset when unsynchronized data exists false</td>
</tr>
<tr>
<td>server.backupMaintenance.interval</td>
<td>Maximum amount of time that the back-up server can serve licenses in failover 3d</td>
</tr>
<tr>
<td>server.syncCompatibility</td>
<td>Enable sync compatibility when migrating from FlexNet Embedded Server App false</td>
</tr>
</tbody>
</table>

The properties marked with * are the editable properties

**Suspend or Resume the License Server**

Use `-status -suspend` to put the license server in a temporary suspended state and then use `-status -resume` to remove the suspension.

In the suspended state, the license server remains running but does not accept client requests. For example, you might want to suspend the license server so that it can complete other operations, such as synchronization or capability exchanges with the back office.

When you resume the license server, it starts accepting client requests again.

**Task** To suspend the license server

Enter a command similar to the following:

```
flexnetlsadmin -server licenseServer_baseURL -status -suspend
```

The license temporarily stops accepting client requests until you issue the `-status -resume` command (described next).

**Task** To resume the license server

Enter command syntax similar to the following:

```
flexnetlsadmin -server licenseServer_baseURL -status -resume
```
Activating License Rights on the Server

Once the FlexNet Embedded local license server is installed and running, it needs to obtain the pool of licenses from which it then serves licenses as needed to client devices in your enterprise. To obtain this license pool, you perform an activation process, in which a capability request, containing one or more license activation IDs and their counts, used to specify the desired licenses, is sent to the back-office server (FlexNet Operations). The back office then fulfills the request by sending a response, which the license server, in turn, processes to activate the licenses. After the initial activation, you can perform additional activations as needed.

**Note** • If the software producer has preregistered your license server with the back office, licenses are automatically activated when the server is first started.

To perform an activation process, you need to obtain your activation IDs from the software producer. Each activation ID identifies a set of licenses to which you have rights. The software producer should provide you with a description of the licenses and counts defined for a given activation ID and the number of activation ID copies you are allowed to install. The traditional activation method is to use internet access to exchange information with the back office so that the license server is updated with the licenses without manual intervention. However, the FlexNet License Server Administrator command-line tool also supports an offline activation process, in which files are exchanged between the license server and back office by another means of communication.

Refer to the appropriate section:

- Online Activation
- Offline Activation

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.

**Important** • If the license server has not been provisioned with any features, it responds to client requests with a 503 error with the error message "Server instance <instanceID> is not ready". This is expected behavior because the server is not ready in this state.

### Online Activation

Online activation uses the internet to exchange capability information between your license server and the back office to activate licenses on the server. This is the traditional method for activating licenses.

**Task** To activate license rights on the license server

1. Run a command similar to the following:

   ```
   flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 2
   ```

   These options are used:

   - `-id` to specify the activation ID (in this example, `act1`) containing the license rights.
-count to specify the number of copies (in this case, 2) of the activation ID to be installed. If no count is provided, a count of 1 is assumed.

**Note** • The activation count must be greater than 0 and not lower than the current count on the server.

If you want to provide multiple activation IDs, repeat the -activate option:

```
flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 2 -activate -id act2 -count 1
```

2. To view the installed licenses, see Viewing Features Installed on the License Server.

## Offline Activation

Traditional activation relies on having internet access to perform a capability exchange between the license server and back office. However, for security reasons, your license server might have limited or no external network connections. The following offline activation process allows you to generate a capability request (specifying your desired activation IDs and counts) as a binary file, which you then send to the back office by an agreed-upon means. The back office returns the capability response as a binary file that you then process on the license server to activate your licenses.

### Task

**To perform an offline activation**

1. Run the following command to generate the capability request as a binary file:

```
flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 2 -o request.bin
```

These options are used:

- `-id` to specify the activation ID (in this example, `act1`) containing the license rights.
- `-count` to specify the number of copies (in this case, 2) of the activation ID to install. If no count is provided, one copy is assumed.

**Note** • The activation count must be greater than 0 and not lower than the current count on the server.

- `-o` to define the name of the output binary file to which to save the generated request (in this example, `request.bin`).

If you want to provide multiple activation IDs, repeat the `-activate` option (as shown in the previous procedure).

2. Upload the capability-request binary file to the back office, and download the capability-response binary file generated by the back office.

3. Run the following command to process the response generated to install the licenses on the license server:

```
flexnetlsadmin -server licenseServer_baseURL -activate -load capabilityResponse.bin
```

The `-load` option is used to specify the capability response file returned by the back office (in this example, `capabilityResponse.bin`).
4. To view the installed licenses, see Viewing Features Installed on the License Server.

# Returning License Rights to the Server

You might need to return license rights when you add a license server or want to move licenses from one license server to another. The return process is identical to the activation process, the only difference being that the license count is reduced instead of increased. For more information about the activation process, see Activating License Rights on the Server.

The FlexNet License Server Administrator command-line tool supports both an online and an offline return process.

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.

## Online Return

The online return process uses the internet to exchange capability information between your license server and the back office to return licenses to the back office. This is the traditional method for returning licenses.

The following procedure assumes that licenses are available on the license server.

<table>
<thead>
<tr>
<th>Task</th>
<th>To return license rights to the back office</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Run a command similar to the following:</td>
</tr>
<tr>
<td></td>
<td><code>flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 0</code></td>
</tr>
<tr>
<td></td>
<td>These options are used:</td>
</tr>
<tr>
<td></td>
<td>• -id to specify the activation ID (in this example, act1) containing the license rights.</td>
</tr>
<tr>
<td></td>
<td>• -count to specify the number of copies (in this case, 0) of the activation ID that should be available on the license server after licenses have been returned. If you specify 0, all licenses are returned to the back office.</td>
</tr>
<tr>
<td></td>
<td>If you want to provide multiple activation IDs, repeat the -activate option:</td>
</tr>
<tr>
<td></td>
<td><code>flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 0 -activate -id act2 -count 0</code></td>
</tr>
<tr>
<td>2.</td>
<td>To view the installed licenses, see Viewing Features Installed on the License Server.</td>
</tr>
</tbody>
</table>

## Offline Return

You perform an offline return when your license server has limited or no external network connection. The process is similar to an offline activation and involves generating a capability request (specifying your desired activation IDs and counts) as a binary file, which you then send to the back office by an agreed-upon means. You upload the binary file to the back office, which generates a capability response as a binary file, which you then process on the license server to decrease your licenses.
The following procedure assumes that licenses are available on the license server.

**Task To perform an offline return**

1. Run the following command to generate the capability-request binary file to decrease license rights:
   
   ```
   flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 0 -o request.bin
   ```
   
   These options are used:
   
   - **-id** to specify the activation ID (in this example, `act1`) containing the license rights.
   - **-count** to specify the number of copies (in this case, `0`) of the activation ID that should be available on the license server after licenses have been returned. If you specify `0`, all licenses are returned to the back office.
   - **-o** to define the name of the output binary file to which to save the generated request (in this example, `request.bin`).
   
   If you want to provide multiple activation IDs, repeat the `-activate` option (as shown in the previous procedure).

2. Upload the capability-request binary file to the back office, and download the capability-response binary file generated by the back office.

3. Run the following command to process the response generated to decrease the number of licenses on the license server:
   
   ```
   flexnetlsadmin -server licenseServer_baseURL -activate -load capabilityResponse.bin
   ```
   
   The `-load` option is used to specify the capability response file returned by the back office (in this example, `capabilityResponse.bin`).

   If the license server outputs error `<IP address:port> Process unsuccessful for capabilityResponse.bin`, this is due to FlexNet Operations requiring a confirmation capability request when reducing the number of copies of a license. The corresponding configuration setting is the Skip Confirmation check box. (For more information on this check box, including its location, consult the FlexNet Operations Producer Portal online help.) If this check box is not selected, the license count available on the license server is decreased in the back office, but has the status “Waiting for confirmation”. To confirm the decrease, perform the following additional steps:

   a. Run the following command to generate another capability-request binary file:
      
      ```
      flexnetlsadmin -server licenseServer_baseURL -activate -id act1 -count 0 -o request2.bin
      ```

   b. Upload the capability-request binary file to the back office.

4. To view the decremented license count, see Viewing Features Installed on the License Server.

**Viewing Features Installed on the License Server**

Use the `-features` command to obtain details about features currently installed on the license server.
If administrative security is enabled on the license server, you might need to provide authorization credentials to perform this operation. For more information, see Use Credentials to Access Operations.

<table>
<thead>
<tr>
<th>Task</th>
<th>To view feature details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter a command similar to this:</td>
<td></td>
</tr>
<tr>
<td>flexnetlsadmin -server licenseServer_baseURL -features</td>
<td></td>
</tr>
</tbody>
</table>

Feature information is displayed:

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Count</th>
<th>Type</th>
<th>Expiration</th>
<th>Start Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>1.0</td>
<td>10</td>
<td>CONCURRENT</td>
<td>permanent</td>
<td>2020-05-03</td>
</tr>
<tr>
<td>f2</td>
<td>1.0</td>
<td>10</td>
<td>CONCURRENT</td>
<td>permanent</td>
<td>2020-05-03</td>
</tr>
</tbody>
</table>

Total number of features : 2

Managing Feature Partitions

FlexNet Embedded License Server includes partitions as a mechanism to allocate licenses to specific client devices or users, to help ensure that these entities have access to the features they need. This functionality is available for the local license server or a Cloud Licensing Service (CLS) instance, hosted in the Revenera cloud.

**Important** • You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations.

For information about the setup and use of partitions on the license server, see Feature Partitions in the More About License Server Functionality chapter.

The following sections describe how to perform tasks to manage partitions:

- Define a Model Definition
- Upload the Model Definition
- View the Model Definition
- Delete the Model Definition
- View Partitions

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.
Define a Model Definition

Partitions are defined using a *model definition*. The model definition is configured and uploaded per license server instance or Cloud Licensing Service (CLS) instance. You can use the FlexNet License Server Administrator command-line tool to view or delete partitions and license count allocations on the license server.

A model definition usually specifies the following:

- One or more partitions
- Rules that allow access to licenses
- Rules that deny access to licenses

The model definition must be written according to the grammar described in [Model Definition Grammar and Syntax—EBNF](#). The following shows a sample model definition:

```plaintext
model "example" {
    partitions {
        partition "engineering" {
            f1 1.0 2
        }
    }

    on hostid("F01898AD8DD3/ETHERNET", "5E00A4F17201/ETHERNET") {
        use "engineering"
        accept
    }

    on any() {
        use "default"
        accept
    }
}
```

This model definition creates the *engineering* partition. It places 2 counts of feature f1, version 1.0 in the *engineering* partition. The remainder of the counts will be placed in the default partition. Client requests that include the ETHERNET hostid F01898AD8DD3 or 5E00A4F17201 are allowed access to the *engineering* partition, ensuring that they have access to these licenses. Client requests from other hostids are only allowed access to counts in the default partition.

For more information about the setup and use of partitions and model definitions on the license server, see [Feature Partitions](#) in the More About License Server Functionality chapter.

The appendix [Model Definition Grammar for Partitions](#), section [Partition Use Case Examples and Their Model Definitions](#), lists use cases that help you write your own model definition.

Upload the Model Definition

To add partitions, create a model definition that defines partitions and rules. Use the `-model -load <filename>` command to upload the model definition to the license server (requires administrator authorization).
Only one model definition can be active for each license server. You cannot upload a model definition with the name reservations or default, because these names are reserved.

**Task**  To upload the active model definition

Enter a command similar to this, using the file you created as the input file (in this example, definition.model):

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword|
-passwordConsoleInput} -model -load definition.model
```

**View the Model Definition**

To view the model definition that is active for the license server, you can use the -model command (requires administrator authorization). This command outputs the current model definition, showing the partitions and rules.

**Task**  To view the active model definition

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword|
-passwordConsoleInput} -model
```

The model definition for your license server is displayed, similar to this:

```
model "example" {
    partitions {
        partition "engineering" {
            feature "f1" 1.0 5
        }
        partition "sales" {
            feature "f1" 1.0 5
        }
    }

    on dictionary("business-unit" : "engineering") {
        use "engineering"
        accept
    }

    on dictionary("business-unit" : "sales") {
        use "sales"
        accept
    }

    on any() {
        use "default"
        accept
    }
}
```
Delete the Model Definition

Use the -model -delete command to delete the model definition that is currently applied to the license server (requires administrator authorization). The default model definition is applied to the server.

Important • You can use either reservations or partitions. Uploading a model definition deletes all currently configured reservations. For more information, see Partitions vs. Reservations.

Task To delete the active model definition

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword| -passwordConsoleInput} -model -delete
```

View Partitions

Use the -partitions command to view all partitions for the license server. The output lists information about every feature and every feature slice in each partition (feature slices are portions of feature counts allocated to a partition).

See JSON Response Details for a description of the most relevant lines in the JSON response.

Task To view all partitions

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword| -passwordConsoleInput} -partitions
```

The following shows example output (shortened for documentation purposes):

```
{
  "partitions": [
    {
      "id": 1,
      "name": "default",
      "lastModified": 1563445226020,
      "default": true,
      "activeFeatureSlices": [
        {
          "id": 1,
          "feature": {
            "id": 1,
            "type": "CONCURRENT",
            "featureName": "f_con_1",
            "featureVersion": "1.0",
            "expiry": "permanent",
            "featureCount": 100,
            "used": 96,
```
JSON Response Details

The following section explains the most relevant lines in the JSON response that is output when you use the `partitions` command to view partitions.

Table 3-1 • Elements in a JSON Response

<table>
<thead>
<tr>
<th>Parent Element</th>
<th>Child Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partitions</td>
<td>&quot;id&quot;: n</td>
<td>The partition ID is assigned to the partition by the license server after the model definition has been uploaded.</td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;name&quot;</td>
<td>The partition name. The names <code>default</code> and <code>reservations</code> are reserved and cannot be used.</td>
</tr>
<tr>
<td></td>
<td>&quot;lastModified&quot;: <code>yyyy-MM-dd'T'HH:mm:ss.SSS'Z'</code></td>
<td>Indicates when the partition was last changed (for example, the allocated feature count or rules for that partition have changed). The date is specified using the ISO 8601 date/time notation, expressed in UTC (for example, <code>2019-10-22T11:16:31.000Z</code>).</td>
</tr>
<tr>
<td></td>
<td>&quot;default&quot;: true</td>
<td>Indicates whether the partition is the default partition or not. The default partition cannot be deleted.</td>
</tr>
<tr>
<td>activeFeatureSlices</td>
<td>&quot;id&quot;: n</td>
<td>Each feature slice is assigned an ID by the license server after the model definition has been uploaded.</td>
</tr>
<tr>
<td></td>
<td>&quot;slice&quot;: n</td>
<td>Total license count for the feature that has been allocated to the partition. Typically, this would be the count specified in the model definition, assuming that sufficient counts are available on the license server. If fewer counts are available on the license server than are specified in the model definition, or if existing usage prevents allocation into the partition after a new model definition has been uploaded, &quot;slice&quot; specifies the actual number of allocated licenses.</td>
</tr>
<tr>
<td></td>
<td>&quot;used&quot;: n</td>
<td>The number of feature counts that have been served from this slice.</td>
</tr>
</tbody>
</table>
Important • You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations. For general information about partitions, see Feature Partitions.

### Table 3-1 • Elements in a JSON Response

<table>
<thead>
<tr>
<th>Parent Element</th>
<th>Child Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activeFeatureSlices</td>
<td>&quot;computedCount&quot;: n</td>
<td>The number of feature counts the slice should have, based on the model definition and the number of feature counts the server has been provisioned with.</td>
</tr>
<tr>
<td>&quot;requested&quot;: n</td>
<td></td>
<td>The license count for the feature that has been specified in the model definition, irrespective of whether sufficient counts are available on the license server.</td>
</tr>
<tr>
<td>&quot;status&quot;: &quot;value&quot;</td>
<td></td>
<td>The status of the feature slice indicates how the slices of a server are used.</td>
</tr>
</tbody>
</table>

Possible values:

- **OVER_ALLOCATED**—The slice has more feature counts than it should have, based on the model definition and the number of counts the server has been provisioned with ("slice" > "computedCount").
- **NORMAL**—The slice has the number of feature counts that it should have, based on the model definition and the number of counts the server has been provisioned with ("slice" = "computedCount").
- **UNDER_ALLOCATED**—The slice has fewer feature counts than it should have, based on the model definition and the number of counts the server has been provisioned with ("slice" < "computedCount").

---

Managing Reservations

The following sample code from a model definition requests 5 counts of feature f1 for the engineering partition:

```json
model "example1" {
    partitions {
        partition "engineering" {
            feature "f1" 1.0 5
        }
    }
}
```
The FlexNet Embedded local license server supports the use of license reservations as a way to pre-allocate licenses to client devices or users to help ensure that these entities have access to the licenses they need.

The reservations you post to the license server are ordered in a hierarchy, enabling more granularity in managing them. For example, reservation entries are the actual feature reservations that you define for a reservation, which specifies the hostid for the specific client device or user to which the feature reservations apply. One or more reservations are assigned to a reservation group, which represents a more global entity to which the client devices and users identified by the reservations belong. This group might identify a department, a specific job role assigned to employees within the company, a physical location, or any other entity applicable to your company. This hierarchy enables you to view, add, or delete reservations at the group level, the reservation (hostid) level, or the reservation entry (feature) level.

For more information about the setup and use of reservations on the license server, see License Reservations in the More About License Server Functionality chapter.

The following sections describe how to perform tasks to manage reservations:

- Add Reservations
- View Reservations
- Back Up Reservation Information to a File
- Delete Reservations

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.

## Add Reservations

To add reservations, create a file whose contents define a new reservation group, which, in turn, contains definitions for one or more reservations (each with a hostid identifying a client device or a user). Each reservation definition contains one or more reservation entries, each entry defined by a feature and its count.

For a quick reference to example reservation-file contents, see the following section Sample License Reservation File. For details about reservation hierarchy, definitions, and behavior, see License Reservations in the More About License Server Functionality chapter.

**Note** • If you want to add or edit reservations in an existing group, delete and then re-create the group, using an input file whose contents include the reservation definitions initially used to create the group, plus the edits. See Delete All Reservations in a Group for instructions on deleting a group.

### Task

**To add one or more reservations**

1. Run a command similar to the following:

   ```bash
   flexnetlsadmin -server licenseServer_baseURL -reservations -load resv-1.json
   ```

   The option `-load` points to the text file (in this example, `resv-1.json`) containing the reservation definitions.

2. Use the instructions in Confirm Reservations Posting to see the added reservations.
If the reservations were not added, see License Reservations in the More About License Server Functionality chapter for information on the types of errors or conditions that can cause reservations to be rejected.

**Sample License Reservation File**

The following shows the contents of an example reservation file that defines a reservation group, called **support**, that contains reservations for two hostids, **7A7A77AA77A** and **Joe**. The file is written in the JSON format required for defining reservations.

**Table 3-2 • Sample Contents of License Reservations File**

```
{
  "name": "support",
  "reservations": [
    {
      "hostId": { "value": "7A7A77AA77A", "type": "ETHERNET" },
      "reservationEntries": [
        { "featureName": "f1", "featureVersion": "1.0", "featureCount": 1 },
        { "featureName": "f2", "featureVersion": "1.0", "featureCount": 1 }
      ],
    },
    {
      "hostId": { "value": "Joe", "type": "USER" },
      "reservationEntries": [
        { "featureName": "f1", "featureVersion": "1.0", "featureCount": 1 },
        { "featureName": "f2", "featureVersion": "1.0", "featureCount": 1 }
      ],
    }
  ]
}
```

**Confirm Reservations Posting**

Use the following procedure to confirm that the reservations you just added were posted correctly to the license server.

**Task**

**To confirm proper posting of reservations**

1. Use the `-reservations` option to list the names and IDs of existing reservation groups posted to the license server. See Show a Summary of Reservation Groups for more information. Note the ID for the group you just added.

2. To drill down to view the details about the reservation group you just added, use the `-reservations -group group_id` syntax, where `group_id` is the group’s ID (obtained in the previous step). The details include the reservations defined in the group, and the reservation entries added to each reservation. See Show Reservation Details by Group for more information.

The output from these options shows you that the reservation group that has been successfully posted. If the posting process has encountered any errors or determined that an insufficient feature count exists for any of the reservations, the entire reservation group is rejected. See License Reservations in the More About License Server Functionality chapter for details.

**View Reservations**

These procedures allow you to view information about the reservations posted to your license server:
Show a Summary of Reservation Groups

Use the -reservations option to provide the list of existing reservation groups posted to the license server. This list provides the group IDs needed to manage reservations at the group level.

**Task**  
**To show a summary of existing reservation groups**

Run a command similar to the following:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -reservations
```

The following shows example output. Note that the ID column shows the specific ID used to manage a group.

```plaintext
===================================================================================
ID    Name          Reservations
===================================================================================
194    Acct2          56
225    Payroll        101
257    support        234
===================================================================================
```

Show Reservation Details by Group

Use the -reservations -group groupID syntax to list the details for every reservation in the specified reservation group. The details list the reservation entries belonging to each reservation and whether the individual reservation entries are enabled or disabled (see Disabled Reservations for more information).

**Task**  
**To show reservation details for a specific reservation group**

Run a command similar to the following:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -reservations -group 257
```

The -group value is the ID of reservation group whose details you want to examine (in this example, 257). You can obtain the group ID by using the -reservations option (as described in the previous section).

The following shows example output for the reservation group 257:

```plaintext
Reservation ID=289, hostid=ETHERNET/7A7A77AAAA7A
================================================================
Entry ID  State  Name  Version  Count
================================================================
290    ENABLED  f2  1.0  1
289    ENABLED  f1  1.0  1
================================================================
Reservation ID=290, hostid=USER/Joe
================================================================
Entry ID  State  Name  Version  Count
```

**Show a Summary of Reservation Groups**

Use the -reservations option to provide the list of existing reservation groups posted to the license server. This list provides the group IDs needed to manage reservations at the group level.

**Task**  
**To show a summary of existing reservation groups**

Run a command similar to the following:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -reservations
```

The following shows example output. Note that the ID column shows the specific ID used to manage a group.

```
===================================================================================
ID    Name          Reservations
===================================================================================
194    Acct2          56
225    Payroll        101
257    support        234
===================================================================================
```

**Show Reservation Details by Group**

Use the -reservations -group groupID syntax to list the details for every reservation in the specified reservation group. The details list the reservation entries belonging to each reservation and whether the individual reservation entries are enabled or disabled (see Disabled Reservations for more information).

**Task**  
**To show reservation details for a specific reservation group**

Run a command similar to the following:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -reservations -group 257
```

The -group value is the ID of reservation group whose details you want to examine (in this example, 257). You can obtain the group ID by using the -reservations option (as described in the previous section).

The following shows example output for the reservation group 257:

```
Reservation ID=289, hostid=ETHERNET/7A7A77AAAA7A
================================================================
Entry ID  State  Name  Version  Count
================================================================
290    ENABLED  f2  1.0  1
289    ENABLED  f1  1.0  1
================================================================
Reservation ID=290, hostid=USER/Joe
================================================================
Entry ID  State  Name  Version  Count
```
In this output, note the Reservation ID value, which identifies the given reservation. (For example, the reservation ID for hostid 7A7A77AA77A is 289; for hostid Joe, it is 290.) You can use this ID to manage the reservation entries in a given reservation.

**Disabled Reservations**

The “disabled” state for a reservation entry means that the entry is currently not active due to an insufficient feature count. A new reservation entry is never added with a “disabled” status. However, a reservation entry might become “disabled” when the license rights on the license server change. To change the reservation entry status back to “enabled”, you might need to delete the reservation group and re-add it once sufficient feature counts on the server are available.

**Back Up Reservation Information to a File**

Use the -o option to write current reservation information to a file in JSON format. You can back up either the summary of current reservation groups or the reservation details for specific group.

**Task** To write a summary of all current reservation groups to a file

Use a command similar to the following:

```
flexnetlsadmin -server licenseServer_baseURL -reservations -o reservations_summary.json
```

**Task** To write reservation details for a specified reservation group to a file

Use a command similar to the following:

```
flexnetlsadmin -server licenseServer_baseURL -reservations -group 257 -o reservations_289.json
```

**Delete Reservations**

Use the following commands to delete reservations at the group or reservation (hostid) level.

**Delete All Reservations in a Group**

Use the -reservations -delete -group group_id syntax to delete an entire reservation group. This action deletes all reservations in the specified group and removes the group itself from the reservations posted on the license server.
To delete a reservation group

1. Run a command similar to the following:

   ```bash
   flexnetlsadmin -server LicenseServer_baseURL -reservations -delete -group 194
   ```

   The `-group` value is the ID of the group whose reservation details you want to delete (in this example, `194`).

2. Use the `-reservations` option to retrieve the list of existing reservation groups to confirm that the group has been deleted (see Show a Summary of Reservation Groups).

Delete All Reservation Entries in a Given Reservation

Use the `-delete -group group_id -reservation reservation_id` syntax to delete an entire reservation (which comprises all reservation entries in the given reservation for a specific hostid) within a given reservation group. The `reservation_id` value is the ID of the specific reservation you want to delete. (You can obtain the reservation ID by using the `-reservations` option, as described in the previous section Show Reservation Details by Group.)

To delete an entire reservation

1. Run a command similar to the following

   ```bash
   flexnetlsadmin -server LicenseServer_baseURL -reservations -delete -group 1 -reservation 290
   ```

   These options are used:
   - `-group` to specify the ID of the group (in this example, `1`) containing the reservation you want to delete.
   - `-reservation` to specify the reservation ID (in this example, `290`) for the reservation you are deleting.

2. Use `-reservations -group group_id` to retrieve reservation details for the specified group to confirm that the reservation has been deleted (see Show Reservation Details by Group).

Managing License Server Policy Settings

The producer provides you with a set of policy settings that control your license server’s operations. As license server administrator, you can review these settings and override any setting that is editable. Refer to the following sections for more information:

- Review Policy Settings
- Write Editable Policy Settings to a File
- Override Policy Settings
- Reset Policy Settings

For a description of all the settings in this file and a list of the specific settings that you are allowed to edit, see Reference: License Server Policy Settings.

If administrative security is enabled on the license server, you might need to provide authorization credentials to perform operations described in this section. For more information, see Use Credentials to Access Operations.
Review Policy Settings

The License Server Administrator command-line tool provides the following views of the license server policy settings. You can examine these views individually or generate a compilation of all views.

Settings marked with an asterisk (*) are editable. See Override Policy Settings for instructions on editing properties.

- Policy Settings for Licensing Operations
- Synchronization Settings
- Administrative Security Policies
- Log Settings
- Settings for Polling the Back Office for License Updates
- Failover Settings
- All Settings

For an example of the output showing general license-server configuration settings (using the `-config -filter general` syntax), see the previous section Show General Configuration Information for the License Server.

To override any of the editable policy settings, see Override Policy Settings.

Policy Settings for Licensing Operations

Use the `-config -filter license-policy` options to retrieve the current policy settings that control licensing operations on the license server. The settings define behavior for license-borrowing and renewal, serving licenses in a virtual environment, and validating the license server’s hostid.

Task: To display policy settings for licensing

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -filter license-policy
```
The following shows sample output:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>licensing.responseLifetime</td>
<td>Lifetime of a served-license response on the client</td>
<td>1d</td>
</tr>
<tr>
<td>licensing.borrowInterval</td>
<td>Borrow interval for served licenses, in seconds</td>
<td>1w</td>
</tr>
<tr>
<td>licensing.renewInterval</td>
<td>Renew interval, as % of the borrow interval, for features</td>
<td>15</td>
</tr>
<tr>
<td>licensing.allowVirtualClients</td>
<td>Virtual clients are allowed to obtain licenses</td>
<td>true</td>
</tr>
<tr>
<td>licensing.allowVirtualServer</td>
<td>License server is allowed to run on a virtual host</td>
<td>true</td>
</tr>
<tr>
<td>licensing.defaultBorrowGranularity</td>
<td>Borrow granularity to use for clients that do not specify one</td>
<td>SECOND</td>
</tr>
<tr>
<td>licensing.hostIdValidationInterval</td>
<td>Frequency with which the license server validates that its host ID has not changed</td>
<td>5m</td>
</tr>
<tr>
<td>* licensing.backup.uri</td>
<td>URI of the backup server in a failover configuration</td>
<td></td>
</tr>
<tr>
<td>* licensing.main.uri</td>
<td>URI of the main server in a failover configuration</td>
<td></td>
</tr>
<tr>
<td>licensing.disableVirtualMachineCheck</td>
<td>Server should check if it is running on a virtual host</td>
<td>false</td>
</tr>
<tr>
<td>* licensing.clientExpiryTimer</td>
<td>Interval between client expiry sessions</td>
<td>2s</td>
</tr>
<tr>
<td>* licensing.borrowIntervalMax</td>
<td>Maximum borrow interval allowed for any client, in seconds by default</td>
<td>NOT_CONFIGURED</td>
</tr>
<tr>
<td>licensing.dropClientEnforcedDelay</td>
<td>A client drop is allowed</td>
<td>0m</td>
</tr>
<tr>
<td>* licensing.security.json.enabled</td>
<td>Security is applied to JSON licensing requests</td>
<td>true</td>
</tr>
</tbody>
</table>

The properties marked with * are the editable properties.

**Synchronization Settings**

Use the `-config -filter sync` options to display the current settings that control the following:

- Automatic synchronization of metered-usage and license-distribution data to the back office
- Whether the ability to recover data from the back office is enabled

For more information about synchronization with the back office, see Online Synchronization to the Back Office and Synchronization From the Back Office in the More About License Server Functionality chapter.

**Task**

*To display synchronization settings*

Enter a command similar to this:

```
flexnetlsadmin licenseServer_baseURL -config -filter sync
```
The following shows sample output:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lfs.syncTo.enabled</td>
<td>Synchronization to the back office is enabled</td>
<td>true</td>
</tr>
<tr>
<td>* lfs.syncTo.pagesize</td>
<td>Maximum number of client records to include in a synchronization message to the back office</td>
<td>50</td>
</tr>
<tr>
<td>* lfs.syncTo.repeats</td>
<td>Amount of time between synchronization sessions with the back office</td>
<td>1d</td>
</tr>
<tr>
<td>* lfs.syncTo.retryCount</td>
<td>Number of times to retry synchronization attempts if a synchronization session with the back office fails</td>
<td>4</td>
</tr>
<tr>
<td>* lfs.syncTo.retryRepeats</td>
<td>Amount of time between synchronization attempts, when synchronization with the back office fails</td>
<td>5m</td>
</tr>
<tr>
<td>* lfs.syncTo.delay</td>
<td>At license-server startup, the amount of time the server should wait before initiating a synchronization session to the back office</td>
<td>2s</td>
</tr>
<tr>
<td>lfs.syncTo.includeAll</td>
<td>Historical license-distribution data for concurrent features is collected and sent to the back office as part of the synchronization data</td>
<td>true</td>
</tr>
<tr>
<td>* lfs.syncTo.rate</td>
<td>Rate limiting of sync messaging to LFS</td>
<td>20</td>
</tr>
</tbody>
</table>

The properties marked with * are the editable properties

### Administrative Security Policies

Use the `-config -filter security` options to display whether administrative security is enabled or disabled on the license server.

**Task**  
To display the security setting

Enter a command similar to this:

```plaintext
flexnetlsadmin licenseServer_baseURL -config -filter security
```

The following shows sample output:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>* security.enabled</td>
<td>Security is applied to REST endpoints</td>
<td>true</td>
</tr>
<tr>
<td>* security.token.duration</td>
<td>REST Security tokens expire after this interval</td>
<td>1d</td>
</tr>
<tr>
<td>* security.http.auth.enabled</td>
<td>Allow use of plain HTTP for secure REST endpoints</td>
<td>true</td>
</tr>
<tr>
<td>* security.ip.whitelist</td>
<td>IP addresses that do not require REST Security</td>
<td></td>
</tr>
<tr>
<td>* security.jwt.cookies</td>
<td>If true, HTTP-Only secure cookies will be accepted as a token source</td>
<td>false</td>
</tr>
<tr>
<td>* security.anonymous</td>
<td>Set to permit anonymous read (GET) access</td>
<td>false</td>
</tr>
</tbody>
</table>

The properties marked with * are the editable properties

### Log Settings

Use the `-config -filter log` options to display the current settings that control the type of messages to log for your license server and the location of the log files.

**Task**  
To display log settings

Enter a command similar to this:
Chapter 3 Using the FlexNet License Server Administrator Command-line Tool
Managing License Server Policy Settings

```
flexnetlsadmin LicenseServer_baseURL -config -filter log
```

The following shows sample output:

```
logging.directory Directory to which the license server writes the log C:\Users\johndoe\flexnetls\demo\logs
* logging.threshold Lowest level of log-message granularity to record, fatal, error, warn, or info LICENSING
```

The properties marked with * are the editable properties.

## Settings for Polling the Back Office for License Updates

Use the `-config -filter capability` options to display the current settings that control the license server’s polling the back office periodically to update license rights on the server. (The polling process involves sending a capability request to the back office to determine whether license updates exist.)

### Task
To display settings used to control the polling process

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -filter capability
```

The following shows sample output:

```
lfs.capability.enabled Capability-request polling is enabled true
* lfs.capability.repeats Amount of time between capability-request polls 1d
* lfs.capability.retryCount Number of capability-polling attempts allowed, if polling fails 3
* lfs.capability.retryRepeats Amount of time between capability-polling attempts, if polling fails 30s
```

The properties marked with * are the editable properties.

## Failover Settings

Use the `-config -filter failover` options to retrieve settings that you have defined on the back-up and main server when license server failover is configured.

For more information about license-server failover configuration, see License Server Failover in the More About License Server Functionality chapter.

### Task
To display failover settings

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -filter failover
```
The following shows sample output:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>* fne.syncTo.enabled</code></td>
<td>To enable server to server synchronization</td>
<td><code>false</code></td>
</tr>
<tr>
<td><code>* fne.syncTo.mainUri</code></td>
<td>URI of the main server in a failover configuration</td>
<td></td>
</tr>
<tr>
<td><code>fne.syncTo.repeats</code></td>
<td>Amount of time between initiating synchronization sessions with the main server</td>
<td><code>5m</code></td>
</tr>
<tr>
<td><code>* fne.syncTo.pagesize</code></td>
<td>Maximum number of client records to include in a synchronization message to the backup server</td>
<td><code>100</code></td>
</tr>
<tr>
<td><code>* fne.syncTo.retryCount</code></td>
<td>When a synchronization from the main server fails, the number of times to retry synchronization</td>
<td><code>1</code></td>
</tr>
<tr>
<td><code>* fne.syncTo.retryRepeats</code></td>
<td>Amount of time between synchronization attempts when, synchronization from the main server fails</td>
<td><code>1m</code></td>
</tr>
</tbody>
</table>

The properties marked with * are the editable properties.

### All Settings

Use `-config` or `-config -filter none` to view a compilation of all settings—general license-server identification, licensing, synchronization, administrative security, logging, and capability polling. The compilation also includes failover settings defined on the main or back-up license server if license server failover is configured.

#### Task

**To display all settings**

Enter a command similar to either of these:

```
flexnetlsadmin -server licenseServer_baseURL -config
```

or

```
flexnetlsadmin -server licenseServer_baseURL -config -filter none
```

### Write Editable Policy Settings to a File

You can write the current editable policy settings for the local license server to a file in JSON format. This feature provides not only a backup of the settings for reference but also the basis for an input file you can use to apply multiple overrides to the settings (see the next section Override Policy Settings). You can write all editable settings or a specific category of editable settings to a file.

#### Task

**To write all editable policy settings to a file**

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -filter none -o allsettings.json
```

To write all editable settings, the value for the `-filter` option must be `none`; the value for the `-o` option is the name of the file to which to write the current settings (in this example, `allsettings.json`).
Chapter 3 Using the FlexNet License Server Administrator Command-line Tool
Managing License Server Policy Settings

Task To write a category of editable policy settings to a file

Enter a command using the desired -filter category. For example, to write policy settings related to synchronization with the back office, use the sync category:

`flexnetlsadmin -server licenseServer_baseURL -config -filter capability -o capability_settings.json`

The file contents look similar to this:

```json
{"lfs.capability.repeats":"1d"}
{"lfs.capability.retryCount":"3"}
{"lfs.syncTo.retryRepeats":"30s"}
```

See Summary of flexnetlsadmin Commands for a description of the -filter categories.

Override Policy Settings

You can override editable policy settings for the local license server as needed for your site. The License Server Administrator command-line tool allows you to override individual settings from the command line or apply multiple overrides through an input file that defines the overrides.

For a quick reference of editable settings, run use the flexnetlsadmin command with -config to review which settings are marked with an asterisk, indicating that they are editable (see Review Policy Settings.)

Refer to Reference: License Server Policy Settings for a description of all policy settings.

Override Individual Settings

Use the following command to override individual policy settings.

Task To override settings directly from the command line

1. Enter a command using this basic syntax, where settingName is name of the setting you are updating, and settingValue is its new value:

   `flexnetlsadmin -server licenseServer_baseURL -config -set settingName=settingValue`

   For example, to override a single policy setting (in this example, lfs.capability.repeats), enter a command similar to this:

   `flexnetlsadmin -server licenseServer_baseURL -config -set lfs.capability.repeats=2d`

   The following shows other syntax requirements:

   • To override more than one setting, enter a command similar to this, using commas to separate the settings:

     `flexnetlsadmin -server licenseServer_baseURL -config -set lfs.capability.repeats=2d, lfs.capability.retryCount=1`

   • To override a setting that contains multiple values (such as might be the case with security.ip.whitelist), use commas to separate values, and then enclose the entire set of values in
square brackets. Additionally, on Windows, you must also enclose the entire bracketed component in double quotes, as shown:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -config -set
security.ip.whitelist="[127.0.0.1,192.0.0.1]"
```

On Linux, you can omit the double quotes around the bracketed component:

```plaintext
flexnetlsadmin -server licenseServer_baseURL -config -set
security.ip.whitelist=[127.0.0.1,192.0.0.1]
```

- To override more than one setting when one of the settings includes multiple values (such as might be the case for `security.ip.whitelist`), enter a command similar to this (shown for Windows):

```plaintext
flexnetlsadmin -server licenseServer_baseURL -config -set
lfs.capability.repeats=2d,security.ip.whitelist="[127.0.0.1,192.0.0.1]"
```

As in the previous example, on Linux you can omit the double quotes around the bracketed component.

2. If necessary, stop and restart the license server to put the overrides into effect.

### From an Input File

The following procedure defines how to apply an input file to override multiple policy settings.

<table>
<thead>
<tr>
<th>Task</th>
<th>To update multiple producer settings using an input file</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Use one of the procedures described in Write Editable Policy Settings to a File to generate a JSON file containing the current editable settings for your license server.</td>
</tr>
<tr>
<td>2.</td>
<td>Open the file in a text editor, change the values for those settings you want to override, and then save the file. For example, to change the <code>lfs.capability.retryRepeats</code> value from 30 seconds to 20 seconds, locate the setting in the text file and change its value: <code>lfs.capability.retryRepeats=20s</code></td>
</tr>
<tr>
<td>3.</td>
<td>Enter a command similar to the following, using the JSON file you updated (in this example, <code>settings.json</code>) as the input file: <code>flexnetlsadmin -server licenseServer_baseURL -config -load settings.json</code></td>
</tr>
</tbody>
</table>

### Reset Policy Settings

You can reset one or more policy settings back to the original default values set by the producer.

For example, suppose that the original value for `lfs.capability.repeats`, as set by the producer for your enterprise, was 1d. At some point, you used `-config -set` to override the default with the new value 2d. If you later wanted set the policy back to the producer's default (that is, 1d), you would use the procedure described here.
Task To reset individual policy settings

Enter a command using this syntax, where `settingName` is the name of the policy setting you are resetting:

```
flexnetlsadmin -server licenseServer_baseURL -config -reset settingName...
```

For example, to reset a single policy (in this example, `lfs.capability.repeats`), enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -config -reset lfs.capability.repeats
```

To reset more than one setting, enter a command similar to this, using commas to separate the settings:

```
flexnetlsadmin -server licenseServer_baseURL -config -reset lfs.capability.repeats,
  lfs.capability.retryCount
```

Monitoring License Distribution to Clients

The FlexNet License Server Administrator command-line tool provides options to monitor the license server’s current license distribution at a summary and detailed level:

- View a Summary of Features and Active Clients
- View License Details

View a Summary of Features and Active Clients

Use the following command to display the total of different features and active clients on the license server.

Task To display a summary of licenses and active clients

Enter a command similar to this:

```
flexnetlsadmin -server licenseServer_baseURL -licenses
```

The following shows sample output:

```
License Server Address : 11.11.1.111:7070
Number of features    : 2
Number of clients     : 1
```

View License Details

Use the following command to display these license details:

- Each feature currently installed on the license server and its current used and available counts
- Each client device that currently has features checked out and the checked-out count for each feature
- A summary that includes the total counts of features not being used (available) and being used, along with the number of uncounted features
Task  To display license details

Enter a command similar to this:

```bash
flexnetlsadmin -server licenseServer_baseURL -licenses -verbose
```

The following shows sample output:

```
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Version</th>
<th>Count</th>
<th>Used/Available</th>
<th>Expiration</th>
<th>Start Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>1.0</td>
<td>6</td>
<td>4/2</td>
<td>permanent</td>
<td>2020-01-14</td>
</tr>
<tr>
<td>f2</td>
<td>1.0</td>
<td>3</td>
<td>2/1</td>
<td>permanent</td>
<td>2020-02-05</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Client Alias Feature / Used Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING/client-14 DeviceABC f2/2</td>
</tr>
<tr>
<td>STRING/client-1 DeviceXYZ f1/4</td>
</tr>
</tbody>
</table>
```

Total of all feature counts: 9
Number of installed features: 2
Number of "uncounted" features: 0
Total of used feature counts: 6

Note • In this display, the total number of licenses per feature or for all features is the sum of the available and used counts. For example, if feature f1 shows 4 counts used and 2 counts available, the total number of f1 licenses installed on the server is 6.

Add `-json` to display output in JSON format instead of the default table layout:

Task  To display license details in JSON format

Enter a command similar to this:

```bash
flexnetlsadmin -server licenseServer_baseURL -licenses -verbose -json
```

The following shows sample output:

```
[
  {
    "id" : 3,
    "type" : "CONCURRENT",
    "featureName" : "f1",
    "featureVersion" : "1.0",
    "featureId" : "a704e059-671e-4378-8492-f6c4ce05cc9c",
    "featureCount" : "2",
    "overdraftCount" : 0,
    "expiry" : "permanent",
    "startDate" : "",
    "used" : 0,
  }
]```
Viewing Current Binding Status

If the producer has enabled a special binding-break detection feature for the license server, you can check the server’s current binding status as reported by this feature.

**Task**

**To view the current binding status of the license server**

Enter a command similar to this:
flexnetlsadmin -server licenseServer_baseURL -binding

If the producer has not enabled the binding-detection feature, the following is displayed:

```
{
    "bindingPolicy" : "disabled"
}
```

If the binding-detection feature is enabled, the results show the current policy for binding breaks on the license server and the server’s current binding status, as shown in this example output:

```
Binding Policy :86400 seconds
Binding Status :soft break
```

The next sections provide more information about this output.

**Binding Policy**

The Binding Policy property shows the current policy that the producer has set for binding breaks for the license server. This policy uses one of the following values to define the action taken should the binding-break detection feature perceive a break:

- **hard**—An immediate hard binding break is imposed—that is, the license server can no longer serve licenses.
- **soft**—A soft binding break will be in effect, allowing the license server to continue to serve licenses despite the break.
- **n seconds**—A soft binding break with a grace period will go into effect, allowing the license server to continue to serve licenses until the grace period expires. (The n seconds value shows the length of the grace period.) When the expiration occurs, a hard binding break goes into immediate effect, and licenses can no longer be served.

**Binding Status**

The Binding Status property shows the current binding status on the license server:

- **ok**—No binding break is detected.
- **hard break**—A binding break is detected, and the license server is no longer able to serve licenses. If a grace period is defined, this status goes into effect when the grace period expires.
- **soft break**—A binding break is detected, but the license server continues to serve licenses. If a grace period is defined, the server continues to serve licenses until the grace period expires, at which time the status changes to hard.

**Repair a Binding Break**

To repair the binding break, contact the producer for the correct repair procedure. You might be required to send a capability request to the back office to initiate the repair process.

**Summary of flexnetlsadmin Commands**

The following describes the command options and associated arguments used by the FlexNet License Server Administrator command-line tool.
Usage

The following shows general command usage for the license server administrator and an enterprise user.

For a License Server Administrator

The following is command usage for a license server administrator performing general administrative tasks:

```
flexnetlsadmin -server licenseServer_baseURL [-authorize adminName {adminPassword|
passwordConsoleInput}] -command_option [command_option_arguments] [-command_option...]
```

The following is the command usage for a license server administrator managing user accounts when administrative security is enabled:

```
flexnetlsadmin -server licenseServer_baseURL -authorize adminName {adminPassword|
passwordConsoleInput} -users [-create|edit|delete]
```

For an Enterprise User

The command usage for an enterprise user is the following:

```
flexnetlsadmin -server licenseServer_baseURL [-authorize userName {userPassword|
passwordConsoleInput}] -command_option [command_option_arguments] [-command_option...]
```

Additional Information

For additional assistance, see the following:

- For the license server’s base URL (`licenseServer_baseURL`) required for each command, see License Server URL Designation.
- If administrative security is enabled on the license server, you are required to provide authorization credentials to perform certain `flexnetlsadmin` operations. For more information about providing these credentials, see Use Credentials to Access Operations.

Command Summary

The following table summarizes `flexnetlsadmin` command options and arguments.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-help</td>
<td>(no argument)</td>
<td>Lists the syntax of all <code>flexnetlsadmin</code> commands.</td>
</tr>
</tbody>
</table>
**Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-server</td>
<td>(no argument)</td>
<td>Sets the license server base URL required in every command as the first option. For a local license server, use the following URL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>flexnetlsadmin.bat -server <a href="http://licenseServerURL/api/1.0/instances/instanceID">http://licenseServerURL/api/1.0/instances/instanceID</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a CLS instance, use the URL that the producer provides, similar to this:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://siteID.compliance">https://siteID.compliance</a>. flexnetoperations.com/api/1.0/instances/instanceId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alternatively, set this value as the environment user variable FLEXNETLS_BASEURL on your machine so that you no longer have to include this option in the commands. See License Server URL Designation for more information.</td>
</tr>
</tbody>
</table>

**Note** • Use of the HTTPS protocol on the local license is strongly recommended when administrative security is enabled on the license server.
Chapter 3  Using the FlexNet License Server Administrator Command-line Tool

Summary of flexnetlsadmin Commands

The -authorize command option and its related options and arguments are in effect only when administrative security is enabled on the license server. Position the -authorize option after the -server option and before any other options. Administrators can choose to enter their password directly in the command or, for security reasons, use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input. The remaining options and arguments for -authorize are described next. See Using the Command-line Tool to Manage Administrative Security for details.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| -authorize adminName (adminPassword | The -authorize command option and its related options and arguments are in effect only when administrative security is enabled on the license server. Position the -authorize option after the -server option and before any other options. Administrators can choose to enter their password directly in the command or, for security reasons, use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input. The remaining options and arguments for -authorize are described next. See Using the Command-line Tool to Manage Administrative Security for details.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

[-users] Shows all current user accounts on the license server.

[-users -create userName {userPassword} -passwordConsoleInput} [roles]] Creates a new user account:

- The userName and userPassword must meet the criteria specified in Credential Requirements.
- To avoid exposing the password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.
- Roles are optional. If no role is specified, ROLE_READ is assigned by default. Valid roles include ROLE_READ, ROLE_RESERVATIONS, ROLE_ADMIN, ROLE_DROP_CLIENT, and ROLE_PRODUCER. Multiple roles can be combined using a plus sign (+) without any spaces in between (for example, ROLE_READ+ROLE_RESERVATIONS). See User Roles Defining Administrative Privileges for details.

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.
Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-authorize adminName</td>
<td>[users -edit userName {password</td>
<td>newPassword</td>
</tr>
<tr>
<td>(adminPassword)</td>
<td></td>
<td>• The password is not optional. Enter either the user’s current password or, if changing the password, the new password (which must meet the criteria listed in Credential Requirements).</td>
</tr>
<tr>
<td>-passwordConsoleInput</td>
<td></td>
<td>• To avoid exposing the password as a command-line argument, you can use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.</td>
</tr>
<tr>
<td>(cont.)</td>
<td></td>
<td>• Assignment of new roles is optional. If one or more new roles are specified, they replace all current roles. If no roles are specified, the user’s current role assignment remains in effect. For information about roles, see User Roles Defining Administrative Privileges.</td>
</tr>
</tbody>
</table>

Note • (Linux only) If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password.

[-users -delete userName] Deletes the user account specified by userName

[any options listed below] Authorizes your access, as needed, to operations specified by any of the options listed below.
Chapter 3  Using the FlexNet License Server Administrator Command-line Tool
Summary of flexnetlsadmin Commands

### Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| -authorize userName (userPassword
-passwordConsoleInput) | [specific options listed below]                    | Authorizes an enterprise user’s access to the specific operations to which that user has privileges. Users can choose to enter their password (as defined for their account) directly in the command or, for security reasons, use the -passwordConsoleInput option, which then issues a prompt to enter the password as console input.  

**Note**  
*(Linux only)* If you supply a password directly on the command line you may need to escape some characters with a backslash in order to prevent the shell from interpreting them. The safest approach is to surround the whole password with single quotes, and backslash any single quote that is part of the password. |
| -status         | [-suspend | -resume]                                      | Shows information about the license server, including its status (as active, inactive, or suspended), its build and version, the back-office URL, and other information.  
The -suspend option temporarily suspends the license server (that is, the license server is still running but does not accept capability requests from client devices).  
The -resume option ends the license server’s suspended state so that the server can proceed with normal operations. |
| -hostid         | (no argument)                                       | Displays the license server’s hostid value used to fulfill capability requests against a back-office server. If the server has multiple hostid values, the list contains the available hardware Ethernet addresses and dongle IDs. If virtual hosts are supported, the VM UUID will also be listed. |
| -selected       |                                                     | Returns the current “selected” hostid—that is, the hostid currently used by the license server.                                                                                                               |
| -setactive hostIdValue hostIdType |                                                     | Designates a new “selected” hostid. The new hostid must be one of the hostids returned by the -hostid command.                                                                                                                                                         |
## Table 3-3 - FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-activate</strong></td>
<td>-id activation-id [-count value]</td>
<td>Activates licenses on the server by installing the specified number of copies of license rights (identified by the activation ID) from the back office. You can specify the -activate option multiple times to activate license rights from different activation IDs.</td>
</tr>
<tr>
<td></td>
<td>-activate -id activation_id -count value -O filename</td>
<td>(Part 1 of an offline activation) Generates a capability request containing the activation ID and saves the request as a binary file. To generate the request with multiple activation IDs, repeat the -activate option for each ID.</td>
</tr>
<tr>
<td></td>
<td>-load filename</td>
<td>(Part 2 of an offline activation) Processes the offline capability-response binary file from the back office to install licenses on the license server.</td>
</tr>
<tr>
<td><strong>-licenses</strong></td>
<td>[-verbose]</td>
<td>Retrieves summary information or details (with -verbose) about current license distribution to client devices.</td>
</tr>
<tr>
<td><strong>-features</strong></td>
<td>(no argument)</td>
<td>Shows details about the features in the current license pool on the server.</td>
</tr>
</tbody>
</table>
Chapter 3  Using the FlexNet License Server Administrator Command-line Tool
Summary of flexnetlsadmin Commands

Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
</table>
| -config | [-filter none | license-policy | sync | security | log | general | capability | failover] | Lists all license-server policy settings. Use the -filter argument to list settings for only specific categories. The categories include:  
  - none: All categories  
  - license-policy: Policies for license distribution and usage  
  - sync: Policies for synchronization with back office  
  - security: The policy enabling administrative security on the license server  
  - log: Logging parameters  
  - general: Attributes identifying the license server  
  - capability: Policies for polling the back-office for license updates  
  - failover: License server failover  
  You can specify multiple categories for the -filter option. For example, `flexnatlsadmin -config -filter sync log` shows synchronization and logging settings. (Use -config or -config -filter none to list all settings.)  
  Those settings that you can edit are listed with an asterisk. |

120  Company Confidential  FNE-2021.07-LAG01  FlexNet Embedded 2021.07 License Server Administration Guide
Table 3-3 · FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-config (cont.)</td>
<td>-set settingName=settingValue, settingName=settingValue...</td>
<td>Overwrites the current value for the specified policy setting. You can provide multiple value pairs separated by commas. For additional syntax formats, see Override Policy Settings.</td>
</tr>
<tr>
<td></td>
<td>-reset settingName,settingName...</td>
<td>Resets one or more policy settings back to their original default values set by the producer. Separate multiple setting names by commas. For additional syntax formats, see Override Policy Settings.</td>
</tr>
<tr>
<td>-o filename</td>
<td></td>
<td>Writes the current, editable policy settings (except those is the general category) to the specified file in JSON format. This argument can be used with the -filter none option to write all editable settings to file or with the -filter category option (except for the general category) to limit the scope of editable settings written to the file.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Applies the edits to policy settings from the specified file to the existing policy settings. The contents of the file you are loading must be in JSON format.</td>
</tr>
</tbody>
</table>

In a typical scenario, first run -filter category -o filename to write current editable settings to a file, edit the values as needed, and then apply the edits using the -load filename option.
### Summary of flexnetlsadmin Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-model</td>
<td>(no argument)</td>
<td>Displays the model definition that is currently active, showing the partitions and rules.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Uploads the model definition to the license server. The contents of the file you are loading must be written according to the grammar in Model Definition Grammar and Syntax—EBNF. Only one model definition can be active for each license server instance. You cannot upload a model definition with the name reservations or default, because these names are reserved. <strong>Important</strong> You can use either reservations or partitions. Uploading a model definition deletes all currently configured reservations. For more information, see Partitions vs. Reservations.</td>
</tr>
<tr>
<td>-delete</td>
<td></td>
<td>Deletes the model definition that is currently applied to the license server. The default model definition is automatically applied to the license server.</td>
</tr>
<tr>
<td>-partitions</td>
<td>(no argument)</td>
<td>Displays all partitions for the license server. The output lists information about every feature and every feature slice in each partition (feature slices are portions of feature counts allocated to a partition).</td>
</tr>
<tr>
<td>-reservations</td>
<td>[-group group_id] [-o filename]</td>
<td>Provides a summary of the reservation groups, including each group's ID, currently available to the license server. The -group group_id option shows details—at the reservation (hostid) and reservation-entry (feature) levels—for the specified reservation group. The -o filename option writes the reservation information in JSON format (for all groups or for the specified group ID) to a file.</td>
</tr>
<tr>
<td>-genjson filename</td>
<td></td>
<td>Generates a template file for reservations in JSON format.</td>
</tr>
</tbody>
</table>

---

**Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)**
### Table 3-3 • FlexNet License Server Administrator Command-line Tool Options (cont.)

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-reservations (cont.)</td>
<td>-load filename</td>
<td>Creates a reservation group and its reservation entries via a file containing the reservation definitions in JSON format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> To make changes to a reservation group, delete the group and recreate it with the changes to the reservation definitions.</td>
</tr>
<tr>
<td></td>
<td>-delete -group group_id</td>
<td>Deletes the entire reservation group with all its reservations and reservation entries.</td>
</tr>
<tr>
<td></td>
<td>-reservation reservation_id</td>
<td>Deletes the specified reservation within the specified reservation group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Important</strong> You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations.</td>
</tr>
<tr>
<td>-uploadPublicKey</td>
<td>clientPublicKeyFileName</td>
<td>Uploads a file containing a client-side RSA public key (2048-bit DER-encoded) in octet-stream format to the license server. The producer will provide details about obtaining and uploading this key should such a key be required. License-server administrator credentials are required to use this option.</td>
</tr>
<tr>
<td>-binding</td>
<td></td>
<td>Shows both the current policy for a binding-break detection facility (enabled by the producer) and the current binding status as detected by this facility (ok, hard break, or soft break). If this facility is not enabled, the output indicates as such.</td>
</tr>
<tr>
<td>-version</td>
<td></td>
<td>Shows the version and build information for the FlexNet License Server Administrator command-line tool.</td>
</tr>
</tbody>
</table>
## Summary of flexnetlsadmin Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-model</td>
<td>(no argument)</td>
<td>Displays the current model definition, showing the partitions and rules.</td>
</tr>
<tr>
<td>-load filename</td>
<td></td>
<td>Uploads a new model definition to the server.</td>
</tr>
<tr>
<td>-delete</td>
<td></td>
<td>Deletes the current rule model.</td>
</tr>
</tbody>
</table>

**Important** • After you delete the model that is currently applied to the license server, the default model definition is applied instead.

<table>
<thead>
<tr>
<th>Option</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-partitions</td>
<td>(no argument)</td>
<td>Displays all partitions for the license server. The output lists information about every feature and every feature slice in each partition (feature slices are portions of feature counts allocated to a partition).</td>
</tr>
<tr>
<td>-json</td>
<td>(no argument)</td>
<td>Displays output in JSON format instead of the default table layout.</td>
</tr>
</tbody>
</table>

**Note** • You can query the JSON output of flexnetlsadmin with an external tool like jq. For information about jq, see [https://stedolan.github.io/jq/manual/v1.6/](https://stedolan.github.io/jq/manual/v1.6/).
Using the FlexNet License Server Manager

The FlexNet License Server Manager is a browser-based interface used to monitor and configure the local license server. This chapter contains the following sections to describe how to use the License Server Manager:

- Setting Up the FlexNet License Server Manager
- Providing Credentials on a Secured License Server
- Devices View
- Reservations View
- Feature Usage View
- Offline Server Updates View
- Client Key Operations View
- Properties View
- License Server Manager Information and Settings

**Important** • The FlexNet License Server Manager will be deprecated in a future release. Instead of the FlexNet License Server Manager, you can use the FlexNet License Server Administrator to manage and monitor the local license server and its operations.

Setting Up the FlexNet License Server Manager

The FlexNet License Server Manager is a web application that runs on Apache Tomcat, which must be installed before you can set up the License Server Manager. See Getting Started for any version requirements for the Tomcat server.
Task To set up the FlexNet License Server Manager using the Tomcat server

1. Once the Tomcat server is installed, you need to copy the flsm.war file from the FlexNet Embedded local license server installation directory to the webapps directory on Tomcat. See Preparing to Use the License Server Manager in Getting Started for complete preparation instructions.

2. Launch the Tomcat server using the instructions provided with the Tomcat product. It might be necessary to set the JAVA_HOME or JRE_HOME environment variable before starting the Tomcat server. (On Windows, make sure that the appropriate JAVA_HOME or JRE_HOME setting is defined in the local settings file flexnetls.settings file.)

3. If the license server is not already running, start it. (The License Server Manager requires that the license server be running.)

4. Point a web browser to http://licenseServerHostName:8080/flsm/ (where licenseServerHostName is the server’s host name or IP address, and 8080 is replaced by the correct port, if necessary).

By default, the License Server Manager opens to the Devices view. However, if administrative security is enabled on the license server and you have not provided your authorization credentials, the Login view is opened instead so that you can enter these credentials before proceeding with the next steps. (See Providing Credentials on a Secured License Server for details.)

5. Click the Settings link on the left side of the interface.

6. On the Settings view, check that the following settings are defined correctly:
   - **FlexNet license server host name**—The hostname of the machine on which the license server is running.
   - **FlexNet license server listen port**—The port on which the License Server Manager will look for license server. (The default port is 7870.)

To stop the License Server Manager, close the web page and shut down the Tomcat server.
Providing Credentials on a Secured License Server

When administrative security is enabled on your license server, anyone attempting to administer the license server will need to provide a set of authorization credentials before starting administrative operations. When you provide your credentials as a license server administrator, you will have administrative privileges on the license server. Additionally, other enterprise users can have credentials to perform limited administrative privileges. Refer to Managing Administrative Security on a Local License Server or CLS Instance in Getting Started for details.

Important • Once a browser HTTP client has authenticated with an instance of Tomcat running "flsm.war", any other browser client connecting to the "flsm" URL of that instance of Tomcat will not require any further authentication.

To enter authorization credentials, use the Login view. Enter the user name for your account in the User Name field, enter your password in the Password field, and click Authorize.

Figure 4-2: Login view to enter authorization credentials

Once the credentials have been verified, you can use the License Server Manager to perform the administrative operations for which you are authorized.

If you attempt to perform an operation in the License Server Manager that requires authorization, but you have not provided credentials or are not authorized to perform the operation, you receive an “access denied” message:

Figure 4-3: Message when you have not provided authorization credentials
Page Navigation for Views in the License Server Manager

The Devices View, Reservations View, and Feature Usage View provide a navigation tool that enables you to move easily between pages in the view to locate records you want to see. (Note that the number of records per page is determined by the Page size setting on the Settings view.)

The navigation tool shows the total number of records available and allows you to select a specific page to which to navigate:

Figure 4-4: Page navigation tool

Devices View

The Devices view displays the client devices recognized by the license server as having been served licenses from the server’s shared pool or from licenses reserved for the client. Each client is identified by its device ID, the ID type, and the device type—physical, virtual, or unknown. In addition, a device alias might be displayed if this has been set on the device. (A device alias is also sometimes referred to as a device name or host name. It is a human-readable alias—in contrast to the hostid—which can optionally be included in a capability request.)

Figure 4-5: Devices view

If the list of devices is extensive, you can use the navigation tool to locate records, as described in Page Navigation for Views in the License Server Manager.

When you click the device ID for a specific client, the Device Details view opens, showing details about the client device and the list of features currently served to the client. (Click Back to exit the Device Details page.)
Figure 4-6: Device Details page

Note that the Device Expiry field in the Device Details section shows the timestamp when all currently checked-out features will expire on the device. This timestamp is determined either by the borrow expiration for the checked-out concurrent features or the final expiration date (as defined in the back office) for checked-out metered features.

The Expiry field for a feature in the Consumed Features list shows its final expiration date as defined in the back office.

You can delete individual devices if the Allow device drop option in the Settings view is set to Yes. In that case, a Delete button is displayed next to the Back button. If Allow device drop is set to No, the Delete button is not available.

Reservations View

About Reservations

A factor that can affect the feature set included in the license server’s responses is the use of reservations. A reservation is an optional mapping of a client device or user to a feature name, version, and count, which represents licenses to be reserved for that specific client devices or users in the enterprise. When a particular client sends the license server a capability request with no desired features, the license server response contains all features reserved for that client (and user, if the user has reservations). When a capability request contains desired features, the license server grants the client device these features based on reserved counts and available shared counts.

The reservations saved to the license server are ordered in a hierarchy, enabling more granularity in the managing them. For example, reservation entries are the actual feature reservations that you define for a reservation, which specifies the hostid for the specific client device or user to which the feature reservations apply. One or more reservations are assigned to a reservation group, which represents a more global entity to which the client devices and users identified by the reservations belong.

For more information about the reservation format and how the license server processes reservations, see License Reservations in the More About License Server Functionality chapter.
Important • You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations. For general information about partitions, see Feature Partitions.

The Reservations View

The Reservations view enables you to view existing reservations, add reservations to a new reservation group, delete a reservation group, as well as import and export reservation definitions in a JSON format.

![Reservations View](image)

**Figure 4-7:** Reservations view

The Reservations view displays the contents of the currently defined reservations. The notation in the Status column indicates whether enough licenses exist on the license server to satisfy the reservation. The green circle indicates that a reservation can be satisfied, and the red circle with an exclamation point indicates a reservation that cannot be satisfied.

If the list of reservations is extensive, you can use the navigation tool to locate records, as described in Page Navigation for Views in the License Server Manager.

Clicking Add displays the Add Reservation page, where you add a single reservation entry to a new reservation group. You can add multiple reservation entries to the same group by using the same group name for each entry you create. You can also create multiple groups.

Also note that, in the Device ID field, you can specify either a client device hostid to define a device-based reservation or a user hostid to define a user-based reservation. Be sure specify the correct hostid type in the ID Type field.

Note • Currently the License Server Manager only adds reservations to a new reservation group. If you want to add or delete reservations in an existing group, you must delete that group and then re-create it using the Add Reservation page or the Import function to import a JSON file with the re-created group and its reservations.
In a reservation entry, the version of the feature to be reserved must be a valid FlexNet Embedded version in numeric $a.b$ format, and the reserved count must be a valid positive numeric value.

Note that the version of a feature reservation need not exactly match an actual feature’s version: the reservation will succeed if the feature version is greater than or equal to the reserved version. For example, a reservation for version 1.0 of a feature can be satisfied by version 2.0 of that feature, but not the other way around. (Reservation versions follow normal feature-version syntax rules.)

Click Add to add the reservation entry to the pending reservation group, or click Clear to reset the current entry value without saving. Reservation entries are saved in a Reservations (Pending Submission) table below the edit fields. Click Submit to finalize the reservation list, or click Cancel to exit.
Chapter 4  Using the FlexNet License Server Manager

Reservations View

Figure 4-9: New reservations pending submission

Clicking Delete in the Reservations view enables you to delete an existing reservation group.

Figure 4-10: Deleting reservation groups

To delete a reservation group, select the check box to the left of the group, and click Delete to add it to the list of pending deletions. After you have selected the groups, click Submit to delete them from the license server.
You can also click **Import** to import a JSON file containing new groups and their reservation entries, or click **Export** to save the current reservations in a JSON file format.

## Feature Usage View

The **Feature Usage** view displays details regarding all the features installed on the license server.

![Feature Usage view](image1)

**Figure 4-11:** Feature Usage view

If the list of features is extensive, you can use the navigation tool to locate records, as described in **Page Navigation for Views in the License Server Manager**. You can also search for a specific feature by entering its name in the **Search** field and then clicking the “magnifying glass” icon. (Provide the exact and complete feature name for the search string.)

![Search tool for features](image2)

**Figure 4-12:** Search tool for features

If you have several pages of features, you can list all the features in a single, sorted page by clicking anywhere in the feature table heading:

![Mechanism to put multiple pages into a single-page list on Feature Usage view](image3)

**Figure 4-13:** Mechanism to put multiple pages into a single-page list on Feature Usage view

When you click a feature name, the **Feature Details** page is displayed, containing detailed information about the feature, including the feature start date, the total count allocated to the server, the count currently being used, the number of reserved licenses within the used count, and any feature selectors and vendor string information defined for the feature.

This page also shows a **Served Info** section, listing the client devices that currently have the feature’s counts checked out.
Offline Server Updates View

The Offline Server Updates view enables you to update the served license pool using offline operations instead of direct communication to a back-office server. In the upper half of the view, you can do either:

- Save a capability request to a binary file by clicking Download in the Save capability/confirmation request... section. This request is used to poll for any license updates for the license server.

- Enter an activation ID (in the Rights ID field) and count in the Generate offline request... section. When you click Download, the activation request is saved as a binary file. This request is used to obtain license rights for the license server through a specific activation ID that the software producer has provided you.

Once the request is downloaded as a binary file, you then communicate the binary request to the back-office server through an offline means.

When you have received a binary response file from the back office, browse for the file by clicking Browse in the lower half of the view and then click Upload. The license server then processes the response, which updates the contents of the server’s trusted storage.
Client Key Operations View

The **Client Key Operations** view provides a means of uploading a file containing a client-side RSA public key (2048-bit DER-encoded) in octet-stream format to the license server. The producer will provide details about obtaining and uploading this key should such a key be required. License-server administrator credentials are required to use this option.

To upload the file, click **Choose File** to browse for and select the appropriate file, and then click **Upload**.

Properties View

The **Properties** view displays the current policy settings used by the FlexNet Embedded local license server. Settings can be overridden in this view, with the exception of settings that could present a security vulnerability if editable by anyone but the producer—such as response lifetime, failover maintenance interval, or enablement of synchronization to the back office.

Any changes made in this view take effect when you click **Save** at the bottom of the page.
Refer to the Reference: License Server Policy Settings to determine which policy settings you are allowed to update, the internal name for each setting, and its description.

![License Server Properties](image)

**Figure 4-17:** License Server Properties view

The properties exposed on this page include:

- **Server host ID:** The license server’s hostid value used to fulfill capability requests against a back-office server. If the server has multiple hostid values, the list contains the available hardware Ethernet addresses and dongle IDs. If virtual hosts are supported, the VM UUID will also be listed. Select the value registered with your back-office system.

- **General Properties:** The license server version, device UUID, and status.

- **Secure REST API Settings:** Settings that control administrative security on the license server.

- **Server Sync Settings:** Properties for synchronizing to the back office.

- **License Generation:** Policies that help manage served features.

- **Settings for server to server sync between FNE servers:** Policies for failover synchronization.

- **Capability polling settings:** Settings that control whether and how often the license server contacts the back office for license updates and how often the server should retry communications after a failed attempt.

- **Locally deployed License Server settings:** Settings specific to your license server and environment.

- **Logging Properties:** Log locations and the lowest level of granularity for log messages captured in the log.

The license server’s producer settings are stored in the `producer-settings.xml` file. These settings are described in Reference: License Server Policy Settings.

## License Server Manager Information and Settings

Below the other commands in the FlexNet License Server Manager menu is the **License Server Manager** group, which contains the **About** and **Settings** views. The **About** view (not pictured) displays build, system, and browser information. In the **Settings** view, you specify the port number and host name (network name or IP address) for
the license server, as well as the number of records to show on a page. These configuration settings appear in a separate list because they can be modified even if the license server is inaccessible, unlike the license server policy settings and configuration properties.

The default port for the license server is 7070.

Figure 4-18: Settings view for License Server Manager

The properties available in this view include:

- **Page size**—The number of records to display per page
- **FlexNet license server host name**—The hostname of the machine the license server is running. See Preparing to Use the License Server Manager in the Getting Started chapter for more information about this file.
- **FlexNet license server listen port**—The port number for the license server. (The default port for the server is 7070.)
- **Connect using HTTPS**—The option indicating which protocol—HTTP or HTTPS—must be used for FlexNet License Server Manager communications with the license server:
  - **Yes**—Use HTTPS communications.
  - **No**—Use HTTP communications.
- **Allow device drop**—Allows/disallows the deletion of individual devices (using Delete on the Device Details page).
  - **Yes**—The Device Details page shows the Delete option.
  - **No**—On the Device Details page, the Delete option is not visible.

Note that the ability to delete a device is dependent on the license server policy setting licensing.dropClientEnforcedDelay, which ultimately specifies whether dropping a client is allowed. For more information, see Reference: License Server Policy Settings.
More About License Server Functionality

This chapter provides background information about certain functionality that can be enabled for your FlexNet Embedded license server and that requires you to perform additional set-up steps. Note that the functionality described in this chapter applies mainly to the local license server, except for functionality involving license reservations, which applies to both the local license server and a CLS (Cloud Licensing Service) instance. The following topics are discussed:

- General Information
- License Borrowing
- Feature Partitions
- License Reservations
- Online Synchronization to the Back Office
- Offline Synchronization to the Back Office
- Status of Synchronization to the Back Office
- Synchronization From the Back Office
- License Server Failover
- Outgoing HTTPS
- Incoming HTTPS
- Proxy Support for Communication with the Back Office
- Trusted Storage Backup and Restoration
- Public Key Upload

General Information

The following provides some general information about using license server functionality
About “licenseServer_baseURL” Used in Commands

Some the commands referenced in this chapter use the license server’s base URL, indicated by licenseServer_baseURL in the command. For more information about this URL, see Base URL for the License Server in the Getting Started chapter.

Authorization Credentials for Administrative Security

If administrative security is enabled for the license server, you might need to provide authorization credentials to perform certain functions described in this chapter. For more information about administrative security and setting up authorization credentials, see Managing Administrative Security on a Local License Server or CLS Instance in the Getting Started chapter.

License Borrowing

The borrow interval can be used to specify an early expiration date for a feature. In other words, the borrow interval is the maximum amount of time that a client can borrow a feature from the license server. Once the borrow interval expires, the feature is no longer available for acquisition on the client. The client must send another capability request to the license server to borrow the feature again. By setting a borrow interval, a license administrator or producer can control the use of licenses more efficiently and ensure that unused licenses are returned to the license pool so that they are available to users who need them.

A borrow interval can be set at different levels:

- **In the back office**—By default, the producer specifies the borrow interval at feature-level. This borrow interval is also referred to in this documentation as the feature borrow interval.

  Note • All current versions of FlexNet Operations require the feature borrow interval to be set.

- **In producer-settings.xml**—The property licensing.borrowInterval specifies the default borrow interval. The default interval is one week (1w). This value is only editable by the producer. This borrow interval is also referred to in this documentation as the server borrow interval.

- **In a client request**—The end user can define a borrow interval that is included in capability requests that a client sends to the license server. This borrow interval is also referred to in this documentation as the client borrow interval.

- **In a configuration parameter**—The license administrator can specify the configuration parameter licensing.borrowIntervalMax to restrict the borrow period of the clients and make more efficient use of their licenses where clients are significantly shorter lived than the borrow interval set at any of the levels mentioned in the preceding bullet points. This borrow interval is also referred to in this documentation as the admin borrow interval.

  Note • The configuration parameter licensing.borrowIntervalMax cannot be used for metered features.

A feature’s current borrow expiration can never exceed the final expiration time for that feature. Should the borrow expiration be greater than the feature’s final expiration, the borrow interval is shortened to the final expiration time (including grace period, where applicable). In addition, a borrow-interval granularity is applied to the effective borrow interval. For more information, see Borrow Granularity.
Determining the Effective Borrow Interval

The following bullet points help you determine the borrow interval that will apply to a feature:

- **If the feature borrow interval has been set in the back office**, the effective borrow interval is the lowest of the following values:
  - feature borrow interval (set in the back office)
  - client borrow interval (set in a client capability request)
  - admin borrow interval (set using the configuration parameter `licensing.borrowIntervalMax`)

- **If the feature borrow interval has not been set in the back office**, the effective borrow interval is the lowest of the following values:
  - server borrow interval (defined in `producer-settings.xml` by the property `licensing.borrowInterval`)
  - client borrow interval (set in a client capability request)
  - admin borrow interval (set using the configuration parameter `licensing.borrowIntervalMax`)

The following diagram illustrates the behavior:

![Borrow interval precedence](image)

**Figure 5-1:** Borrow interval precedence

The expiration date that is calculated through the borrow interval can never exceed the feature expiry date (including grace period, where applicable).

Setting the Admin Borrow Interval

Users require the ROLE_ADMIN role to set the admin borrow interval. The configuration parameter `licensing.borrowIntervalMax` can be set in the following ways:

- Using the FlexNet License Server Administrator command-line tool, `flexnetlsadmin`
- Using the FlexNet License Server Manager

When setting the admin borrow interval, note the following:
• Take into account that the borrow-interval granularity (if set) can affect the effective borrow interval. For more information, see Borrow Granularity.

• Consult with your software producer to ensure that their software's license renewal mechanism can adapt to borrow intervals shortened by the licensing.borrowIntervalMax parameter.

• Do not set an excessively short admin borrow period which could result in the license server being overloaded by client requests.

### Setting the Admin Borrow Interval Using flexnetlsadmin

To set the admin borrow interval in flexnetlsadmin, use the -authorize option (to provide your credentials that authorize you for the task) with the -config -set option.

**Task**

To set the admin borrow interval in flexnetlsadmin

Enter a command similar to this:

```bash
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword} -passwordConsoleInput} -config -set licensing.borrowIntervalMax value
```

where *value* is the borrow interval in seconds.

### Setting the Admin Borrow Interval Using the FlexNet License Server Manager

You set the admin borrow interval in the FlexNet License Server Manager user interface.

**Task**

To set the admin borrow interval using the FlexNet License Server Manager

1. In the FlexNet License Server Manager, navigate to the Properties view.
2. In the License Generation section, locate the Borrow Interval Maximum setting and enter the required value.

### Borrow Granularity

A borrow-interval granularity is applied to the effective borrow interval. The granularity is the time unit (day, hour, minute, or second) by which the license server rounds up the borrow interval.

For example, if the borrow interval is 60 seconds, and the borrow granularity is day, then a license issued at 5:05:01 PM expires at 11:59:59 PM—which is the borrow interval (5:06:01 PM) rounded to the end of the nearest day. Likewise, if granularity is minute, expiration is at 5:06:59 PM. If the granularity is second, expiration is 5:06:01 PM.

By default, the granularity is set by the producer on the license server, and the default is second. In addition, a capability request can specify a granularity, which will override the granularity set on the license server.
Renew Interval

The renew interval can be used to specify how often—if ever—the client may attempt to recontact the local license server. Successful contact extends the expiration based on the effective borrow interval (in other words, the timer for the effective borrow interval is restarted).

The producer can set the renew interval at feature level in the back office, or at server level using the property licensing.renewInterval in producer-settings.xml. The renew interval is set as a percentage of the effective borrow interval. The default value is 15. For information on how the effective borrow interval is calculated, see Determining the Effective Borrow Interval.

Important • This specification by itself does not lead to enforcement. The client-side APIs must extract this value from the license server capability response and take appropriate action.

Example

The following example illustrates how the renew interval might be used:

Settings: Effective borrow interval = 604800s (1 week); renew interval = 15

Suppose the device initially obtains its license on January 1. The 7-day effective borrow interval results in a license that is valid until January 8. The 15% renew interval indicates that the device should attempt to recontact the server after 90720 seconds, that is, on January 2 (15% of 604800 seconds = 90720 seconds = approx. 1 day). If successful, the license is now valid until January 9. If unsuccessful, the license remains valid until the end of the original borrow period, which is still January 8.

Feature Partitions

Important • You can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. If currently no license reservations are defined for your license server, Revena strongly recommends to use the partitions functionality to allocate licenses. The license reservations feature may be deprecated in a future release. For more information, see Partitions vs. Reservations.

The partitions functionality available with the FlexNet Embedded license server enables you to allocate licenses to a group of client devices or users to help ensure that these entities have access to the features they need.

A partition can be viewed as a group of licenses. Each license server has a default partition which contains the pool of licenses available to its client devices. The license server distributes licenses from this license pool on a first-come-first-served basis to client devices requesting them.

However, you can define additional partitions (also called named partitions) and divide your license estate between all partitions (that is, named partitions and the default partition). Each partition contains license counts for one or more features, which can be allocated to client devices or users. It is important to note that in a partition, licenses are not assigned to individual client devices or users. Instead, counts allocated to a partition can be accessed by a number of devices or users that meet certain criteria, such as, for example, be member of a particular business unit or have a certain hostid. Allocated license counts can be checked out by any device or user that fulfills the specified criteria.
As the license server administrator, you can choose to allocate the entire pool of licenses, a portion of the licenses, or none of the licenses. Any license that is not allocated remains in the default partition and is available to any device on a first-come-first-served basis.

The following sections provide information about partitions on the FlexNet Embedded license server:

- Defining Partitions Using a Model Definition
- Model Definition Components
- Server Behavior When Distributing Feature Counts to Partitions
- Server Behavior when Assigning Features to Clients
- Partitions vs. Reservations
- Limitations of Partitions

### Defining Partitions Using a Model Definition

Partitions are defined using a *model definition*. The model definition is configured and uploaded per local license server instance or Cloud Licensing Service (CLS) instance. You can use the FlexNet License Server Administrator command-line tool to view or delete partitions and license count allocations on the license server.

A model definition specifies the following:

- **One or more partitions**—Partitions are optional. If a model definition does not include any named partitions, all counts are placed in the default partition, and rules can only allow or deny access to the default partition. If a model definition contains named partitions, each partition will hold one or more features. Within the named partitions, counts are arranged in *feature slices* based on their feature ID, meaning each slice contains only counts from a single feature (which has a unique feature ID). For each feature, the partition specifies the name of the feature, its version, and its feature count. The feature count can either be specified as a number or as a percentage of the total of the available feature count.

- **Rules that allow access to licenses**—Rules allow client devices or users that fulfill certain conditions access to counts in specified partitions, in a given order. For example, you could specify that only clients of a specific host type can receive licenses from a certain partition.

- **Rules that deny access to licenses**—You can define criteria that block certain client devices or users from obtaining licenses. This means that no licenses will be allocated to entities meeting those criteria, even if sufficient licenses are available on that license server.

After the model definition has been uploaded to the license server, the definition is saved in the database. Partitions and rules defined in the model definition persist even if the license server is restarted.

For information about the format and grammar of the model definition, see Model Definition Grammar for Partitions. This section also contains some sample definitions that help you write your own definition.

For information about managing (uploading, deleting, and viewing) the model definition, see Managing Feature Partitions.

### Model Definition Components

A model definition typically contains the following entities:
Model

The top line of the model definition specifies the model name. The model name cannot be “default” or “reservations”, because these names are reserved for the default partition and the definition that would be generated automatically for existing reservations (when converting reservations to partitions). For information about the relationship between partitions and reservations, see Partitions vs. Reservations.

Example

model "test"
Partitions

You can define one or more named partitions under the optional `partitions` element. (If you do not specify a partition, all feature counts are placed in the default partition.) For each partition, you need to specify a name followed by one or more features that the partition should contain. For each feature, you need to specify the feature name, feature version, and feature count. The feature count can be expressed as a number or as a percentage of the overall feature count that is available on the license server.

Example

```plaintext
partitions {
    partition "p1" {
        "f1" 1.0 50
        "f1" 2.0 25
        "f2" 1.0 100
    }
}
```

For information about how the license server allocates feature counts to partitions, see Server Behavior When Distributing Feature Counts to Partitions.

Allocating Remaining License Counts

The `remainder` keyword can be used to allocate any remaining license counts to a specified partition. This keyword is particularly useful in preventing the situation where integer rounding would cause left-over counts to remain in the default partition when trying to configure partitions summing up to a count of 100%.

Using the `remainder` keyword is equivalent to specifying `100%`, which allocates all available license counts. Any partition later in the model definition than the one specifying the `remainder` keyword (or `100%`) in the model definition (including the default partition) receives zero counts for that feature. For an example demonstrating its use, see the Use Case: Partition Receiving Entire Remaining Feature Count.

Example

```plaintext
partitions {
    partition "p1" {
        "f1" 1.0 33%
    }
    partition "p2" {
        "f1" 1.0 33%
    }
    partition "p3" {
        "f1" 1.0 remainder
    }
}
```

Rules and Conditions

Rules define criteria that allow or block certain client devices or users from obtaining licenses from specified partitions. This section covers the following topics:

- Rule Syntax
Rule Syntax

Rules are defined using the following simplified syntax:

```plaintext
on <condition> {
    [use "<partition-name>"],
    [accept|deny|continue]
}
```

where `<condition>` specifies one or more conditions that must be matched.

Example

If the condition `dictionary("business-unit" : "engineering")` evaluates to true—that is, the capability request's vendor dictionary contains the `business-unit:engineering` key-value pair—access to the engineering partition is granted:

```plaintext
on dictionary("business-unit" : "engineering") {
    use "engineering"
    accept
}
```

The following subsections provide more information about configuring rules:

- **Conditions Syntax**—Describes how to configure a condition in simple terms.
- **AND, OR, and NOT Operators**—Describes how to combine conditions within a rule.
- **Keywords and their actions**—Explains the keywords `accept`, `deny`, and `continue`.
- **Letting Server Specify Counts**—Explains how to use the `without requested features` directive, to let the server allocate feature counts.
- **Default Behavior**—Describes how feature counts are served when a feature request does not meet any of the conditions in the model definition.

**Conditions Syntax**

This section describes the syntax for defining conditions in simple terms. For detailed information, see Model Definition Grammar and Syntax—EBNF.

The following shows a simplified conditions syntax:

```plaintext
[NOT]<condition_type>(<parameter_list>) [AND|OR [NOT] <condition_type>(<parameter_list>)]
```

where `<condition_type>` is replaced with a `condition type` (hostid, hostname, hosttype, dictionary, or any), followed by a value or value/type pair that qualifies the condition.

Optionally, additional conditions—`<condition_type>(<parameter_list>)`—can be chained on using an AND, OR, or optional NOT operator.
AND, OR, and NOT Operators

The syntax supports the AND, OR, and NOT operators to combine conditions within a rule to create a more complex rule. The operators have synonyms that are familiar to C and Java programmers:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Synonym</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>OR</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>!</td>
</tr>
</tbody>
</table>

The syntax for combining conditions within a rule using these operators is illustrated in detail in section Model Definition Grammar and Syntax—EBNF. You may also find it helpful to review the provided sample rules.

Precedence Considerations

The AND and OR operators have left-to-right associativity; NOT has right-to-left associativity. NOT has a higher precedence than AND and OR.

Parentheses are not supported. An expression like this:

```
on hosttype("tv") and not hostname("xyz") || hostid("h1")
```

is interpreted as:

```
OR(
    AND(hosttype("tv"), NOT(hostname("xyz"))),
    hostid("h1")
)
```

Sample Rules

The following sample rules in the appendix Model Definition Grammar for Partitions illustrate the use of operators:

- Use Case: Exclusive Use of Feature Counts for Business Unit With Exception of Specific Clients
- Use Case: Exclusive Use of Feature Counts for Business Unit and Specified Clients from other Business Units
- Use Case: Assigning Features Based on Combined hosttype and hostname Properties

Keywords and their actions

The following table explains the keywords and their actions.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>If the conditions of the rule are met and the rule ends with accept, the feature request is granted. No further rules are evaluated.</td>
</tr>
</tbody>
</table>
Letting Server Specify Counts

There might be scenarios when you want the license server to specify the features that should be served to a client. The server cannot override a capability request containing desired features coming from the client. However, if a capability request lists no desired features, the `without requested features` directive lets the server allocate feature counts.

You can use the `without requested features` directive to do the following:

- **Allocate specific feature counts.** To achieve this, list the feature counts that should be served. The following example shows the syntax:

  ```
  on hostname("myhost") {
    use "p1", "default"
    without requested features {
      "f1" 1.0 1
    }
    accept
  }
  ```

  For a use case example, see Use Case: Letting Server Specify Counts.

- **Allocate the entire feature count from a partition.** To achieve this, combine `without requested features` with the `all from "partitionName"` instruction. Replace `partitionName` with the name of the partition whose entire list of features, as specified in the model definition, should be allocated. Note that if the specified partition does not have sufficient counts to satisfy the request, counts are allocated from subsequent partitions listed in the rules (provided that the client has access to those partitions). The following example shows the syntax:

  ```
  on hostname("myhost") {
    use "p1", "default"
    without requested features {
      all from "p1"
    }
    accept
  }
  ```

This is the default behavior if no other keywords are present.

The lack of an `accept` or `deny` action also means that the `on any` clause (see Default Behavior, below) will be in scope to be matched.

### Table 5-2 • Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>continue</td>
<td>The <code>continue</code> keyword causes rules on subsequent lines to be evaluated, allowing multiple rules that match a single request. The <code>continue</code> keyword therefore allows clients to accumulate counts from more than one partition. For a use case example, see Use Case: Allocating Counts from Multiple Partitions (“Continue” Action). This is the default behavior if no other keywords are present. The lack of an <code>accept</code> or <code>deny</code> action also means that the <code>on any</code> clause (see Default Behavior, below) will be in scope to be matched.</td>
</tr>
<tr>
<td>deny</td>
<td>If the conditions of the rule are met and the rule ends with <code>deny</code>, the feature request is refused. No counts are acquired and any previously held counts are returned to the server. No further rules are evaluated. If the conditions of the rule are not met and the rule ends with <code>deny</code>, the next rule is evaluated.</td>
</tr>
</tbody>
</table>

### Letting Server Specify Counts

There might be scenarios when you want the license server to specify the features that should be served to a client. The server cannot override a capability request containing desired features coming from the client. However, if a capability request lists no desired features, the `without requested features` directive lets the server allocate feature counts.

You can use the `without requested features` directive to do the following:

- **Allocate specific feature counts.** To achieve this, list the feature counts that should be served. The following example shows the syntax:

  ```
  on hostname("myhost") {
    use "p1", "default"
    without requested features {
      "f1" 1.0 1
    }
    accept
  }
  ```

  For a use case example, see Use Case: Letting Server Specify Counts.

- **Allocate the entire feature count from a partition.** To achieve this, combine `without requested features` with the `all from "partitionName"` instruction. Replace `partitionName` with the name of the partition whose entire list of features, as specified in the model definition, should be allocated. Note that if the specified partition does not have sufficient counts to satisfy the request, counts are allocated from subsequent partitions listed in the rules (provided that the client has access to those partitions). The following example shows the syntax:

  ```
  on hostname("myhost") {
    use "p1", "default"
    without requested features {
      all from "p1"
    }
    accept
  }
  ```

This is the default behavior if no other keywords are present.

The lack of an `accept` or `deny` action also means that the `on any` clause (see Default Behavior, below) will be in scope to be matched.

### Table 5-2 • Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>continue</td>
<td>The <code>continue</code> keyword causes rules on subsequent lines to be evaluated, allowing multiple rules that match a single request. The <code>continue</code> keyword therefore allows clients to accumulate counts from more than one partition. For a use case example, see Use Case: Allocating Counts from Multiple Partitions (“Continue” Action). This is the default behavior if no other keywords are present. The lack of an <code>accept</code> or <code>deny</code> action also means that the <code>on any</code> clause (see Default Behavior, below) will be in scope to be matched.</td>
</tr>
<tr>
<td>deny</td>
<td>If the conditions of the rule are met and the rule ends with <code>deny</code>, the feature request is refused. No counts are acquired and any previously held counts are returned to the server. No further rules are evaluated. If the conditions of the rule are not met and the rule ends with <code>deny</code>, the next rule is evaluated.</td>
</tr>
</tbody>
</table>
Chapter 5  More About License Server Functionality
Feature Partitions

For a use case example, see Scenario: Server-specified Counts.

Default Behavior

When a feature request does not satisfy any of the rules in the model definition, an on any() condition is expected that either denies or allows access to a partition. If no such on any() condition is provided, the default is that the feature will be served from the default partition (if features are available). This behavior is equivalent to the following code:

```plaintext
on any() {
    use "default"
    accept
}
```

The examples in this book generally do not explicitly show this final on any() condition.

Condition Types

This section describes the following condition types:

- **Hostid Condition**
- **Hostname (Device Name) Condition**
- **Hosttype (Device Type) Condition**
- **Vendor Dictionary Condition**

These condition types can be combined using AND, OR, and NOT, to form more complex rules. For more information, see AND, OR, and NOT Operators.

Note that in this section, the syntax describing each condition has been simplified. For a detailed syntax description, refer to Model Definition Grammar and Syntax—EBNF.

Hostid Condition

When a client requests a license from the license server, the client includes the unique hostid in the request to which the license server binds the licenses sent in the response. The hostid condition enables you to allocate licenses from a specified partition to one or more client devices that are identified by a hostid.

The hostid is specified as a value/type pair (for example, 7200014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

**Syntax**

```plaintext
hostid_condition = 'hostid', '(', parameter_list, ')' ;
```

**Example**

The following hostid condition allows any feature requests with hostid h1 or h2 to use features from partition p1.

```plaintext
on hostid("h1", "h2") {
    use "p1"
    accept
}
```
Hostname (Device Name) Condition

Your software producer can set a host name on a client device. (This host name is also sometimes referred to as a device name.) A host name is a human-readable “alias”—in contrast to the hostid—which can optionally be included in a capability request. For information about host names that might be available, contact your software producer.

Syntax

\[
\text{hostname\_condition} = \text{'hostname'}, \text{ '(' parameter\_list, ')'};
\]

Example

The following host name condition allows any feature requests with host name \text{ABC} to use features from partition \text{p2}.

\[
\text{on hostname(}'\text{ABC}'\text{) \{} \\
\text{ \hspace{1cm} use } \text{ p2} \\
\text{ \hspace{1cm} accept} \\
\text{\}}
\]

Note that the host name is case sensitive.

Hosttype (Device Type) Condition

Your software producer can set a host type on a client device. (This host type is also sometimes referred to as a device type.) A host type can optionally be included in a capability request. For information about host types that might be available, contact your software producer.

Syntax

\[
\text{hosttype\_condition} = \text{'hosttype'}, \text{ '(' parameter\_list, ')'};
\]

Example

The following host type condition allows any feature requests with host type \text{device} to use features from partition \text{p3}.

\[
\text{on hosttype(}'\text{device}'\text{) \{} \\
\text{ \hspace{1cm} use } \text{ p3} \\
\text{ \hspace{1cm} accept} \\
\text{\}}
\]

Note that the host type is case sensitive.

Vendor Dictionary Condition

The vendor dictionary enables the software producer to send custom data in a capability request (in addition to the FlexNet Embedded–specific data) to the license server and vice-versa. Basically, the vendor dictionary provides a means to send information back and forth between the client and server for any producer-defined purposes, as needed; FlexNet Embedded does not interpret this data.

Vendor dictionary data is stored as key–value pairs. The key name is always a string, while a value can be a string or a 32-bit integer value. Keys are unique in a dictionary and hence allow direct access to the value associated with them.

For information about vendor dictionary data that might be available, contact your software producer.
Syntax

\[
\text{vendor\_dictionary\_condition} = \text{\textquotesingle}\text{dictionary\textquotesingle}, \text{\textquotesingle}(), \text{\texttt{keyword\_parameter\_list}}, \text{\textquotesingle}\); \]

Example

The following vendor dictionary condition allows any feature requests from the engineering business unit (as defined in the vendor dictionary) to use features from partition p4.

\[
\text{on dictionary(\text{\texttt{\textquotesingle}business\_unit\textquotesingle} : \text{\textquotesingle}engineering\textquotesingle)} \{ \\
\quad \text{use \text{\texttt{p4}}}; \\
\quad \text{accept} \\
\} 
\]

Server Behavior When Distributing Feature Counts to Partitions

This section describes the distribution of feature counts to partitions for these situations:

- When a New Model Definition Is Uploaded
- When the Feature Count Changes on the License Server
- When Clients Return or Renew Counts to the Server

When a New Model Definition Is Uploaded

When the model definition is uploaded to the license server, feature counts are placed into partitions. Counts are allocated to partitions in the order that is specified in the model definition, starting with the topmost partition.

The following rules apply when allocating feature counts of the same name, but with different versions and expiry dates:

1. Features that match the version specified in the model definition are allocated first. Features with longer expiry dates are allocated before features with a shorter expiry date.

2. If no features are available that match the version specified in the model definition, features of the next higher version are allocated. Features with longer expiry dates are allocated before features with a shorter expiry date. (Features of a lower version are not allocated.)

Features holding a start date in the future will not be allocated to named partitions but will remain in the default partition. A feature with a future start date will stay in the default partition even beyond its start date until the model definition is reapplied to the license server.

Within partitions, counts are arranged in feature slices based on their feature ID, meaning each slice contains only counts from a single feature (which has a unique feature ID). Partitions that have the exact feature counts, including the exact feature versions, as specified in the model definition are considered fully satisfied.

When the available number of counts has been distributed, any partition for which insufficient counts are available will remain empty or will only be partially filled.

When the model definition on the license server changes, any counts that are currently in use by clients are recorded as used against the partition the counts originate from, provided that both of the following statements are true:
• The partition name and requested feature counts have not changed.
• No partitions higher in the model definition have been modified, added or removed.

If either condition does not hold true, used counts are recorded as used against the default partition. Note that adding or deleting partitions that are listed below a partition with counts in use by clients does not result in its used counts being recorded against the default partition after a model definition change.

**Optimization Logic For Model Definition Changes**

When the model definition changes, any partitions that are already fully satisfied are skipped when feature counts are placed into partitions. As a result, it is possible that existing features in fully satisfied partitions are not replaced with newly available features that have a longer expiry date. Instead, these newly available features are placed into lower-ranked named partitions (provided that these are not fully satisfied) or the default partition.

To ensure that higher-ranking partitions receive features with longer expiry dates, it is recommended that you delete and re-upload the model definition. This forces a redistribution of feature counts across all partitions, regardless of the current allocation of feature counts.

**When the Feature Count Changes on the License Server**

When feature counts are added, reduced or expire on the license server, the server reapplies the model definition. The server redistributes the counts across partitions as described in When a New Model Definition Is Uploaded.

**When Clients Return or Renew Counts to the Server**

When there have been no changes to the model definition or to the licenses held by the server, licenses returned by clients are returned to the original slice from which they were acquired.

However, in scenarios where clients were holding usage prior to a model change or a change to the licenses held by the server, counts returned by clients may need to be redistributed to reflect the new state of the server.

The following sections describe more about feature count redistribution:

• Basic Redistribution Rules
• Order of Redistributing Counts

In these sections, *slices in deficit* refers to slices that have fewer counts than they should have, according to the model definition and the number of counts the server has been provisioned with ("slice" < "computedCount").

**Basic Redistribution Rules**

When clients had counts in use prior to a model change or prior to a change of feature counts on the license server, the following basic redistribution rules apply:

• A client cannot renew counts that it could not have acquired, based on the current model definition that is in force on the license server.
• Counts can only be reallocated to slices with a matching feature ID.
Order of Redistributing Counts

The server redistributes returned counts in the following order:

1. The server allocates feature counts to reduce or eliminate overage before redistributing counts.
2. Counts are redistributed to slices in deficit that are accessible to the client, in the order in which partitions that are accessible to the client are specified in the current model definition.
3. Counts are redistributed to slices in deficit, irrespective of whether these are accessible to the client. The server redistributes counts to partitions in the order in which they are specified in the current model definition.
4. Returned counts are only returned to overdraft after all slices in deficit have been adjusted to the counts they should have according to the model definition. Overdraft counts are only ever available from the default partition.
5. Any remaining counts are returned to the default partition.

Server Behavior when Assigning Features to Clients

When the license server receives a capability request from a client, it tries to serve the feature counts matching the requested version from the first partition that is configured in the model definition. If the request cannot be fulfilled from the first partition, or can only be partially fulfilled, the server will try to serve any remaining counts to the client from further feature slices and partitions in a specific hierarchy. The following rules apply, in order, when fulfilling capability requests:

1. Regular counts take priority over overdraft counts (overdraft counts are only available from the default partition).
2. Counts are served from accessible partitions in the order partitions are specified in the model definition (starting from the top).
3. Counts that match the requested version are served before counts of a higher version (lower versions are not accepted).
4. Counts with the shorter expiry date are served before counts with a later expiry date.

The following diagrams illustrate this hierarchy. Figure 5-2 shows the order in which feature counts are assigned from partitions.
Figure 5-2: Flow of feature allocation

Figure 5-2 includes a subroutine (highlighted in green) that explains how counts are assigned if the request is served from more than one slice. This subroutine—see Figure 5-3—also comes into play when regular (non-overdraft) counts are exhausted, but the request has access to the default partition which includes overdraft counts:
Figure 5-3: Flow of feature allocation in the subroutine

Partitions vs. Reservations

Partitions are conceptually similar to reservations but offer significantly more flexibility in controlling your server's license pool. If currently no license reservations are defined for your license server, Revenera strongly recommends to use the partitions functionality to allocate licenses.

Read this section if one or more of the following statements apply to you:

- You want to know the pros and cons of using partitions or reservations. See Comparison of Partitions and Reservations.
• You are currently using reservations and would like to find out how partitions impact on the behavior of reservations. See Impact of Partitions Functionality on Using Reservations.

• You have previously been using reservations and would like migrate to partitions. See Migrating Reservations to Partitions.

• You have migrated to partitions but want to use reservations again. See Returning to Reservations after Migrating to Partitions.

If you have not been using reservations before and are now starting to use partitions, you can skip this section.

The following diagram illustrates your options:

---

**Comparison of Partitions and Reservations**

The following table lists the main differences between partitions and reservations.

**Table 5-3 • Partitions vs. Reservations**

<table>
<thead>
<tr>
<th>Action</th>
<th>Partitions</th>
<th>Reservations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counts are allocated based on...</td>
<td>• hostid</td>
<td>• hostid</td>
</tr>
<tr>
<td></td>
<td>• host name (device name)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• host type (device type)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• vendor dictionary data</td>
<td></td>
</tr>
</tbody>
</table>

If the license server has been provisioned with less counts than are posted in a reservation or are specified in the model definition, respectively, the license server allows access to feature counts based on the rules in the model definition until all available counts are in use by entitled clients. ...the license server rejects the entire reservation group.
Impact of Partitions Functionality on Using Reservations

The reservation feature is now implemented via partitions. Any existing reservation groups on the license server are automatically translated into partitions and rules which are saved in a model definition with the model name reservations. This model definition is regenerated immediately when reservations are edited.

The automatic generation of the reservations model definition has no impact on the reservations functionality. You can continue configuring reservations using the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager, and the license server will distribute licenses according to the reservations that have been configured.

However, if you want to continue using reservations, it is imperative that you do not configure a new model definition and upload it. Configuring a new model definition will remove all reservations on the license server.

Migrating Reservations to Partitions

When you migrate reservations to partitions, you can do either of the following:

- Use the reservations model definition that is generated automatically as a basis for your new partitions and rules.
- Create a new model definition from scratch.

The migration process described below allows for both strategies.

Before starting the migration, you might want to use the FlexNet License Server Administrator command-line tool (flexnetlsadmin) to view information about the reservations currently active on your license server. For procedural information, see Managing Reservations in Using the FlexNet License Server Administrator Command-line Tool.

**Important** • Note that after you upload a model definition, any reservations that had been configured will be removed and can no longer be accessed.

**Task**

To migrate reservations to partitions:

1. Download the reservations model definition using the FlexNet License Server Administrator. Enter a command similar to this:

   ```
   flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword} -passwordConsoleInput -model
   ```

   The model definition for your license server is displayed. For more information, see View the Model Definition.
This step is optional if you decide to create a new model definition that is not based on the original reservations.

2. Do either of the following
   - Edit the reservations model definition and rename it. (Renaming is required because model definitions with the name reservations or default cannot be uploaded.)
   - Create a new model definition (note that the names reservations and default are reserved and cannot be used).

A full description of the model definition grammar can be found in the appendix, Model Definition Grammar for Partitions. This appendix also includes a number of use cases and sample model definitions that help you customize the model definition to suit your requirements.

3. Upload the definition using the FlexNet License Server Administrator. Enter a command similar to this, using the file you edited or created as the input file:

```
flexnetlsadmin -server licenseServer_baseURL -authorize yourAdminName {yourAdminPassword| -passwordConsoleInput} -model -load definition.model
```

For more information, see Upload the Model Definition.

The new model definition takes effect immediately after upload.

**Returning to Reservations after Migrating to Partitions**

After you upload a model definition with a name other than reservations, any reservations that had been configured will be removed and can no longer be accessed.

To return to using reservations, you need to remove the active model definition. After that, you can configure reservations (using the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager).

For information about deleting the active model definition, see Delete the Model Definition.

**Limitations of Partitions**

Note the following about partitions:

- Named partitions support only concurrent features, but not metered features. Metered features will always remain in the default partition. This matches the behavior of reservations, because metered features cannot be reserved.

- Using feature selectors in combination with partitions may produce unexpected results. A feature selector is a key-value structure included as part of the feature’s definition created in the back office.

- The license server failover configuration does not support feature partitions. During a failover period, feature counts will only be distributed from the default partition.
License Reservations

Important • It is important to note that you can use either reservations or partitions. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations. For general information about partitions, see Feature Partitions.

If currently no license reservations are defined for your license server, Revena strongly recommends to use the partitions functionality to allocate licenses. Reservations are conceptually similar to partitions, but partitions offer significantly more flexibility in controlling your server’s license pool. The license reservations feature may be deprecated in a future release.

The license reservation feature available with the FlexNet Embedded license server (the local license server or a CLS instance) enables you to control the distribution of the server’s license pool within your enterprise.

With no reservations defined, the license server distributes licenses on a first-come-first-served basis to client devices requesting them. However, with the use of reservations, the server pre-allocates specific license counts to certain client devices or users to help ensure that these entities have access to the licenses they need. As the license server administrator, you can choose to pre-allocate the entire pool of licenses, a portion of the licenses, or none of the licenses. Any license not reserved remains in the shared pool of licenses and is available to any device on a first-come-first-served basis.

The following sections provide background information about reservations on the FlexNet Embedded license server:

- Overview of Reservation Types
- Reservation Hierarchy
- Managing Reservations
- Disabled Reservations
- License Allocation on the Server
- Processing the Capability Request When Reservations Are Used
- Reservation Limitations

Overview of Reservation Types

The FlexNet Embedded license server supports two types of license reservations—device-based and user-based.

- A device-based reservation is assigned to the unique hostid identifying the client device on which the software producer’s product is running.

- A user-based reservation is assigned to a unique user ID (also called a hostid). A user with reserved counts for a feature can access these counts from any client device. Thus, user-based reservations help to ensure that users who always need access to certain licenses can obtain them from any client device. (For example, users might need to access their licensed application from their laptop, tablet, or phone.)

For information about how the license server processes reservations to satisfy the feature count specified in a capability request, see Processing the Capability Request When Reservations Are Used.
**Reservation Setup**

When setting up a license reservation on the license server, you must provide a hostid to identify the entity—client device or user—to which the reservation belongs. Obtain from the producer the specific hostid types (for example, ETHERNET, STRING, USER, or other) that you can use to define a device-based or user-based reservation. See Managing Reservations for more information about setting up reservations.

**Reservation Hierarchy**

The reservations you post to the license server are ordered in a hierarchy, enabling more granularity in managing them. For example, **reservation entries** are the actual feature reservations that you define for a **reservation**, which is identified by the hostid for the specific client device or user to which the feature reservations apply. One or more reservations are assigned to a **reservation group**, which represents an entity to which the client devices and users identified by the reservations belong. This entity might be a department, a job role assigned to specific employees within the company, a physical location, or any other entity applicable to your company. This hierarchy enables you to view, add, or delete reservations at the group level, the reservation (hostid) level, or the reservation entry (feature) level.

When reservations are added to the license server, the following IDs are generated for the various hierarchal components that define a reservation:

- A **group ID** (GID) for a reservation group
- A **reservation ID** (RID) for the reservation, based on the hostid identifying a specific client device or user within the reservation group
- A **reservation-entry ID** (EID) for each reservation entry defined in a specific reservation

The following diagram illustrates the reservation hierarchy. For example, reading from right to left, trace the reservation entry **EID 1** for feature **Feat1**:

- **EID 1** belongs to the reservation **RID 1**, defined for the hostid **Host1**.
- **RID 1** is one of three reservations belonging to the reservation group **GID 1**, also named **Group1**.
Chapter 5  More About License Server Functionality
License Reservations

Managing Reservations

Both the FlexNet License Server Administrator command-line tool and the FlexNet License Server Manager provide mechanisms for managing reservations on the license server:

- The License Server Administrator command-line tool adds or updates reservations by loading an input file (that you provide) containing the reservation definitions, including updates. The tools provides commands that delete reservations at the reservation group and reservation levels. For more information, see Using the FlexNet License Server Administrator Command-line Tool.
- FlexNet License Server Manager allows you to create and post individual reservations one at a time or import reservations from a file. All reservations are managed from a single internally created reservation group. For more information, see Using the FlexNet License Server Manager.

The next sections describe what happens “behind the scenes” when reservations are managed on the license server:

- Definitions in JSON Format
- Add a New Reservation Group
- Add or Delete Reservations in an Existing Group

Definitions in JSON Format

When you provide an input file that contains reservation definitions, the file contents must be in JSON format. The reservation entries for a given reservation (identified by a hostid) use the following format:
Chapter 5  More About License Server Functionality

License Reservations

License Reservations

163

Then “name” field is optional. If omitted, the license server simply generates the group ID.

Add a New Reservation Group

The following shows an example of reservation definitions (in JSON format) that might be posted to define a new reservation group—in this case, a group named support, containing reservation entries for two reservations, one for hostid 7A7A77AAA77A and one for hostid Joe:

Table 5-4  Contents of Sample License Reservations File

<table>
<thead>
<tr>
<th>name</th>
<th>reservations</th>
</tr>
</thead>
<tbody>
<tr>
<td>support</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When this reservation information is processed without any errors, the license server does the following:

- Creates the reservation group support if it does not yet exist. If it does exist, an attempt to add another group named support will fail.
- Adds the two new reservations—one for hostid 7A7A77AAA77A (RID 193) and one for hostid Joe (RID 196)—to the reservation group support.

Note  A given hostid can be used only once not only within a given reservation group but within the entire reservation posting. An attempt to add a reservation group that includes a hostid already existing in another group will fail.

- Creates two reservation entries for hostid 7A7A77AAA77A—a reservation for 1 count of feature f1 and a reservation for 2 counts of feature f2.
- Creates one reservation entry for 3 counts of feature f3 for hostid Joe.

Errors that Cause Rejection of New Reservation Groups

When a new reservation group is processed during the posting phase, the entire group is rejected if any of following errors exist.
Chapter 5  More About License Server Functionality
License Reservations

- A group name is repeated within the reservation posting. A group name must be unique within the entire reservation posting.
- A hostid is used in more than one reservation group. A given hostid can be used only once, not only within a given group, but within the entire reservation posting.
- A hostid is missing. A group reservation cannot be created without at least one reservation, defined by a hostid and its reservation entries. Additionally, reservation entries not associated with a reservation cause the entire reservation group to be rejected.
- A reservation has no reservation entries associated with it.
- A reservation entry is invalid. For example, the entry might be missing a feature name, version, or reserved count, or the specified reserved count is less than or equal to zero.
- The feature or feature version specified in a reservation entry is not found in the license pool.
- The feature specified in a reservation entry is expired or has not yet started.
- The feature specified in a reservation entry is a metered feature.
- The license server has an insufficient count of a given feature to fulfill the total requested count for that feature across all reservation entries within the entire group. For more explanation, see the later section License Allocation on the Server.

Add or Delete Reservations in an Existing Group

Currently, the FlexNet License Server Administrator command-line tool lets you add or update reservations at the reservation-group level only. Therefore, if you want to add reservations to an existing group or reservation entries to an existing reservation, you must delete the entire reservation group and then re-create it with the changes. The input file used to recreate the group should contain the reservation group, reservations, and reservation entries used to create the group initially, plus the updates.

Likewise, the License Server Administrator command-line tool provides commands that let you delete reservation groups or reservations only. To remove a reservation entry from a specific reservation, you must drop the entire reservation group and recreate it without the specific reservation entry.

On the other hand, the FlexNet License Server Manager deletes the current reservation group and creates a new one every time a reservation list is submitted to the license server (that is, every time you add, edit, or delete reservations from the Reservations view).

Disabled Reservations

The “disabled” state for a reservation entry means that the entry is currently not active due to an insufficient feature count. A new reservation entry is never added with a “disabled” status. However, when license rights on a the license server change, reservation entries can become disabled due to an insufficient feature count on the license server. Should sufficient counts on the license server become available again, you can delete and recreate the reservation group to re-enable the reservation entries.
License Allocation on the Server

When reservations are posted for a new group, the license server determines the total feature counts required by all the reservations within the group. If the totals cannot be satisfied by the available feature counts on the license server, the entire new group is rejected. Additionally, the license server re-allocates reservations when license counts on the server change. See the following sections:

- When Adding a Reservation Group
- When Feature Counts Change on the Server

When Adding a Reservation Group

In the attempt to add a new reservation group, the license server determines the total feature counts required by all the reservation entries within the new group. If the totals cannot be satisfied by counts not reserved to other reservation groups on the license server, the entire new group is rejected.

Example Allocations When Adding a New Reservation Group

The examples described in this section use the following reservations to pre-allocate the licenses in the server’s license pool. All these reservations are being created in the single new reservation group `payroll`.

```
Table 5-5 • Sample Allocations When Adding a Reservation Group

```{ "name" : "payroll", "reservations" : [  
  { "hostId": { "value" : "111A11AAA11A", "type" : "ETHERNET" },  
   "reservationEntries": [  
     { "featureName": "f1", "featureVersion": "1.0", "featureCount": 1 },  
     { "featureName": "f3", "featureVersion": "1.0", "featureCount": 2 } ]},  
  { "hostId": { "value" : "222A22AAA22A", "type" : "ETHERNET" },  
   "reservationEntries": [  
     { "featureName": "f3", "featureVersion": "1.0", "featureCount": 1 } ]},  
  { "hostId": { "value" : "Joe", "type" : "USER" },  
   "reservationEntries": [  
     { "featureName": "f3", "featureVersion": "1.0", "featureCount": 3 } ]]
]}
```

Example 1: Sufficient Feature Counts

In this example, a license server is allocated 1 count of f1 and 8 counts of feature f3 from the back office. When processing the new reservation group, the license server does the following:

- Determines that a total of 1 count of f1 is required (for device 111A11AAA11A).
- Determines that a total of 6 counts of f3 is required—2 counts for device 111A11AAA11A, 1 count for device 222A22AAA22A, and 3 counts for user Joe.
- Reserves 6 counts of f3 and 1 count of f1.
- Keeps 2 unreserved counts of f3 available for any request on a first-come-first-served basis.
Example 2: Insufficient Feature Counts

If a license server is allocated 1 count of f1 and 4 counts of f3, it does the following when the reservations are posted:

- Determines that a total of 1 count of f1 is required (for device 111A11AAA11A).
- Determines that a total of 6 counts of f3 are required—2 counts for device 111A11AAA11A, 1 count for device 222A22AAA22A, and 3 counts for user Joe.
- Rejects the entire reservation group since only 4 counts of f3 are available.

Example 3: Features Not in the License Pool

If a license server is allocated no (0) counts of f1 but is allocated 8 counts of f3, it rejects the entire reservation group since f1 is not available in the license pool.

When Feature Counts Change on the Server

When features counts are reduced or expire on the license server, the server automatically disables certain reservation entries to adjust to the new feature count. The server uses no particular order as it processes the reservation groups, reservations, or reservation entries to determine which reservation entries to disable.

Processing the Capability Request When Reservations Are Used

The following sections describe how the license server processes capability requests to fulfill licenses when reservations are available:

- Basic Process for Granting Licenses
- Example Scenarios of the Basic Process for Granting Licenses

Basic Process for Granting Licenses

When reservations have been added and the license server receives a capability request from a client device, the server uses the following basic process in its attempt to grant the license count requested.

Note • The process described here assumes that the capability request is defined with no special capability request options that can control aspects of the license server’s behavior in determining which licenses are sent in the capability response. Consult the software producer for information about any such options that client code includes in the request and their effect on the basic fulfillment process.

1. Validates that the capability request contains the appropriate content for processing.
2. Returns all licenses currently served to the client device. (Reserved licenses remain reserved. Shared licenses are returned to the pool.)
3. Searches for reservations assigned to the client device and to the user sending the request from that client.
4. Determines whether desired features are specified in the request.

If no desired features are requested, the server grants the client device whatever licenses have been reserved for it through device-based reservations and through user-based reservations. If no reservations exist, no licenses are granted.

If a desired feature is specified, the server determines whether the specified feature is available through a reservation. If not, it determines whether any shared counts are still available. In summary, the server does the following:

a. Checks for the device-based reserved count first.

b. If device-based reserved count is not sufficient, checks the user-based reserved count for an additional count.

c. If the sum of device-based and user-based reserved counts is not sufficient, checks the available shared counts to fulfill the remaining requested count.

d. If enough reserved and shared counts are available, grants the licenses.

Example Scenarios of the Basic Process for Granting Licenses

The following scenarios illustrate instances of the basic process (described in the previous section) that the license server uses to grant licenses when reservations are involved. The sections describing the scenarios include the following:

- Feature Allocation on the License Server
- Starting State for Each Scenario
- Scenario 1—No Device or User Reservations Available
- Scenario 2—Device Reservations But No User Reservations Available
- Scenario 3—User Reservations But No Device Reservations Available
- Scenario 4—Device and User Reservations Available

For the sake of simplicity, the feature version is omitted in these scenarios. Anytime feature F1 is mentioned, assume that its version is 1.0.

Feature Allocation on the License Server

The license server has 10 counts of feature F1:

- 3 reserved counts for device D1
- 2 reserved counts for user U1
- 5 shared counts available

Starting State for Each Scenario

The starting state for each scenario is as follows:

- Device D1 has been served 4 counts (all 3 reserved counts, 1 shared count).
- Device D2 has been served 3 counts (3 shared counts).
• User U1 currently has no served counts.
• 1 shared count remains available.

**Scenario 1—No Device or User Reservations Available**

When user U7 (who has no reservations) sends a capability request from device D2 (which also has no reservations), the 3 counts of F1 currently served to D2 are returned to the shared counts. (Four shared counts are now available.)

The following describes possible scenarios of what happens next:

• If the request has no desired features, no counts are served since neither D2 nor U7 has reserved counts.
• If the request specifies 4 counts of F1, the remaining 4 available shared counts are served.
• If the request specifies 5 counts of F1, no counts of F1 are sent in the response since only 4 shared counts are available.

**Scenario 2—Device Reservations But No User Reservations Available**

When user U7 (who has no reservations) sends a capability request from device D1, the 4 counts of F1 currently served to D1 (3 reserved counts for D1 and 1 shared count) are returned. (Two shared counts are now available.)

The following describes possible scenarios of what happens next:

• If the request has no desired features, the 3 counts of F1 reserved for D1 are served.
• If the request specifies 4 counts of F1, the 3 counts reserved for D1 plus 1 shared count are served.
• If the request specifies 6 counts of F1, the 3 counts reserved for D1 and the remaining 2 shared counts are insufficient to fulfill the request, so no counts of F1 are sent in the response.

**Scenario 3—User Reservations But No Device Reservations Available**

If user U1 sends a capability request from device D2 (which has no reservations), the 3 counts of F1 currently served to D2 are returned to shared counts. (Four shared counts are now available.)

The following describes possible scenarios of what happens next:

• If the request has no desired features, the 2 counts of F1 reserved for U1 are served.
• If the request specifies 4 counts of F1, the 2 counts reserved for U1 plus 2 shared counts are served.
• If the request specifies 7 counts of F1, the two counts reserved for U1 and the remaining 4 shared counts are not sufficient to fulfill the request, so no counts of F1 are sent in the response.

**Scenario 4—Device and User Reservations Available**

If user U1 sends a capability request from device D1, the 4 counts of F1 currently served to D1 (3 reserved and 1 shared) are returned. (Two shared counts are now available.)

The following describes possible scenarios of what happens next:

• If the request has no desired features, all reserved features—3 counts of F1 for D1 and 2 counts for U1—are served.
If the requests specifies 4 counts of F1, the 3 counts reserved for D1 and 1 of the 2 counts reserved for U1 are served.

If the request specifies 8 counts of F1, the 3 counts reserved for D1, the 2 counts reserved for U1, and the 2 remaining shared counts are not sufficient to fulfill the request, so no counts are sent in the response.

**Reservation Limitations**

Note the following about reservations:

- A license server administrator can use either reservations or partitions on a license server. Reservations and partitions cannot coexist alongside each other. For more information, see Partitions vs. Reservations. For general information about partitions, see Feature Partitions.

- Reservations support only concurrent features; metered features cannot be reserved.

- The license server failover configuration does not support reservations.

**Online Synchronization to the Back Office**

When synchronization to the back office is enabled, the FlexNet Embedded local license server automatically sends a synchronization message to the back office on a periodic basis through an active internet connection with the back office. The type of data submitted in the synchronization message depends on the value specified for the policy setting `lfs.syncTo.IncludeAll` (see Data Synchronized During Online Synchronization for more information). When the back office receives and processes the synchronization message, it sends back a synchronization acknowledgment. The synchronization acknowledgment contains status information and any error information regarding the processing. Additionally, the server log records the event, along with any related errors or warnings.

After a successful synchronization session, information about device clients that have been served licenses is accessible to the back office.

*Note* • This synchronization process described here is “online”—that is, it uses direct internet access as a means of transmitting data to and from the back office. As an alternative to this type of data transmission, the FlexNet Embedded license server supports an offline synchronization process that does not require the server to have direct internet access. See Offline Synchronization to the Back Office.

The following sections describe synchronization to the back office:

- **Enablement**

- **Configuring the Synchronization**

**Enablement**

Your license server’s ability to synchronize to the back office is enabled by the software producer. If you need to change your license server’s current policy about synchronization, contact the software producer.
Data Synchronized During Online Synchronization

Your software producer determines the type of data that is collected during online synchronization using the policy setting `lfs.syncTo.includeAll`. If you need to change your license server’s current policy about the type of data that is synchronized, contact the software producer.

- `lfs.syncTo.includeAll=false`: This mode collects only the current state for each active client device at the point of synchronization and uploads this data to the back office.
- `lfs.syncTo.includeAll=true`: This mode collects all historical client actions (license distribution and metered usage on client devices, based on the server’s deduction records) in the synchronization history and uploads this data to the back office as part of the synchronization.

Note that `lfs.syncTo.enabled` (not editable) must be set to `true` for online synchronization.

Configuring the Synchronization

Using FlexNet License Server Administrator command-line tool or FlexNet License Server Manager, whichever tool is available to you, you can edit certain producer-defined settings that control synchronization:

- If using the FlexNet License Server Administrator command-line tool, see Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- If using the FlexNet License Server Manager, see Properties View in the Using the FlexNet License Server Manager chapter.

You can override the default for any of the editable `lfs.syncTo...` settings to control the synchronization process.

For a description of these settings and a list of those you can edit, see the appendix Reference: License Server Policy Settings.

Offline Synchronization to the Back Office

The traditional process of synchronizing license data from the FlexNet Embedded local license server to the back office relies on the license server having direct internet access to transmit data to the back office. For security reasons, such as data privacy, local license servers might have limited or no external network connections, making it difficult to synchronize data about concurrent-license distribution and metered usage on the server to the back office. The offline synchronization feature supports the ability to download this data from the license server on to another device that has an external network connection and then transmitting the data to the back office.

The following sections describe offline synchronization to the back office:

- Configuring Synchronization Page Size
- Synchronization Tools
- Offline Synchronization Process
Configuring Synchronization Page Size

Offline synchronization uses the producer-defined `lfs.syncTo.pagesize` setting to control the maximum number of transaction records allowed in a single synchronization message sent to the back office. If the amount of accumulated data is larger than the number of records allowed in a single synchronization message, multiple messages are sent during the synchronization session until all data is communicated to the back office.

You can review and edit the current value for this setting by using the license server's FlexNet License Server Administrator command-line tool or FlexNet License Server Manager, whichever tool is available to you:

- If using the FlexNet License Server Administrator command-line tool, see Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- If using the FlexNet License Server Manager, see Properties View in the Using the FlexNet License Server Manager chapter.

For more information about the `lfs.syncTo.pagesize` setting itself, see Reference: License Server Policy Settings.

Synchronization Tools

The license server provides these two tools to perform the offline synchronization operations.

- “serverofflinesynctool.bat” (or “serverofflinesynctool.sh”)
- “backofficeofflinesynctool.bat” (or “backofficeofflinesynctool.sh”)

“serverofflinesynctool.bat” (or “serverofflinesynctool.sh”)

This utility downloads the client records to be synced to the back office to a temporary storage location on (or accessible by) the device that has direct internet access to the back office. It is also used to upload and process the synchronization acknowledgment sent by the back office, purge old downloaded records, and force a download of all synchronization records when necessary.

If administrative security is enabled for the license server, the utility command (shown in the steps that follow) must include an `-authorize` option to specify the credentials needed to run the utility, as shown:

```
serverofflinesynctool.bat -authorize name password -url http://licenseServerHostName:port/api/1.0/...
```

For more information about administrative security and setting up authorization credentials, see Managing Administrative Security on a Local License Server or CLS Instance in the Getting Started chapter.

“backofficeofflinesynctool.bat” (or “backofficeofflinesynctool.sh”)

This utility uploads the client records to be synchronized from the temporary location to the back office. It also receives and saves the synchronization acknowledgment received from back office.

Offline Synchronization Process

Perform the offline synchronization process as described here.
Step 1: Download Records

The first step is to download the transaction records to be synchronized from the license server using the serverofflinesynctool.bat (in Windows) or serverofflinesynctool.sh (in Linux) tool:

```
serverofflinesynctool.bat -url http://licenseServerHostName:port/api/1.0/sync_message/offline
-generate path
```

where:

- `licenseServerHostName:port` is the server name and port for the license server URL from where the records are being downloaded (default port is 7070)
- `path` is the path on the local machine where the records will be temporarily stored

Additionally, if administrative security is enabled, you must include your authorization credentials in the command. See "serverofflinesynctool.bat" (or "serverofflinesynctool.sh") for details.

This command can be run on a system with an external network connection or on the hosting license server. If you run the command on the hosting license server, the files containing the data to be synchronized need to be copied to a machine with external network connectivity.

Once the download completes, a message stating the number of transaction records downloaded is displayed:

```
OfflineSync utility started.
Sync completed for 3 device records.
```

If there are no new records to download, the message displays the following:

```
OfflineSync utility started.
No new data is available.
```

Step 2: Synchronize to the Back Office

Next, use the backofficeofflinesynctool.bat (in Windows) or backofficeofflinesynctool.sh (in Linux) tool to synchronize the records to the back office:

```
backofficeofflinesynctool.bat -url https://siteID.compliance.flexnetoperations.com/deviceservices -out filename path
```

The following describes the parameters used in the command:

- `servername:port` is the server name and port for the back-office URL.
- `filename` is the name of the file to which the synchronization acknowledgment will be written to (for example, `sync_ack.bin`).
- `path` is the path on the local machine where the transaction records to be synchronized were stored by the serverofflinesynctool tool. To specify a specific file, provide the path and file name.

A synchronization acknowledgment message is returned:

```
Successfully sent sync data and received a sync acknowledgment.
```

The synchronization acknowledgment received from the back office is written to the output file (default is `syncack.bin` in the current directory).

```
MessageType="Server sync acknowledgment"
MessageTime="Jan 13, 2019 10:53:46 AM"
LastSyncTime="Jan 13, 2019 10:52:45 AM"
SourceIDType=String
```
**SourceID=BACK_OFFICE**
**SourceIds=BACK_OFFICE**
**TargetID=A088B436F208**
**TargetIDType=Ethernet**

**Step 3: Update the Synchronization Time**

The synchronization acknowledgment needs to be processed on the license server to update the last time of synchronization so that it knows that the data has been synchronized to the back office. When executing the tool, use the same `path` value used to download and synchronize the data to the back office so that the tool can remove the old synchronized data file after it processes the response. (Additionally, if administrative security is enabled, you must include your authorization credentials in the command, as described in “serverofflinesynctool.bat” (or “serverofflinesynctool.sh”).)

```
serverofflinesynctool.bat -url http://licenseServerHostName:port/api/1.0/sync_ack/offline
-process filename path
```

The server responds with the following message:

```
OfflineSync utility started.
Purging file 20140613T105312.fnesync
```

**Force a Download of All Synchronization Records**

The `-force` option can be used with the `serverofflinesync` tool if you encounter an error similar to the following:

```
“Mismatched or out of order time stamp in sync message”: “Expected sync time Jan 24, 2019 11:04:09 AM, got Jan 24, 2019 11:05:00 AM”
```

Checking the synchronized records in the back office reveals that those records processed after a mismatch are not synchronized. To correct the problem, you must force a download of all transaction records to account for any records missed during synchronization. The `-force` option allows you to start the record collection and download from the last successful synchronization time.

```
serverofflinesynctool.bat -url http://licenseServerHostName:port/api/1.0/sync_message/offline
-generate path -force
```

This will dump all the records since the `lastSyncTime` specified in the last synchronization acknowledgment.

**Task**

**To force a download of all records since the last time of synchronization**

1. Process the synchronization acknowledgment on the license server (as described in Step 3: Update the Synchronization Time). This will update the records up to the point of the mismatch. Only those records that were synchronized to the back office are deleted.

2. Recover the records missing from synchronization. However, if you try to download the records again, the server thinks it has already downloaded all the records and responds with no new data is available. To remedy the problem, use the `-force` option with `serverofflinesynctool`. (If administrative security is enabled, you must also include your authorization credentials in the command, as described in “serverofflinesynctool.bat” (or “serverofflinesynctool.sh”).)

```
serverofflinesynctool.bat -url http://licenseServerHostName:port/api/1.0/sync_message/offline
-generate path -force
```

This will dump all the records since the `lastSyncTime` specified in the last synchronization acknowledgment.

3. Run the `backofficeofflinesynctool` to upload the data to the back office.

4. Once the data has been successfully synchronized, process the acknowledgment on the server to update the synchronization date and delete the synchronized data.
Status of Synchronization to the Back Office

You can periodically check your synchronization status to view the last time synchronization was run and the number of transaction records that need to be synchronized. This information is especially helpful when you are using offline synchronization because you can determine whether a synchronization is due based on back-office policies or general maintenance needs. For example, you might be late in reporting metered usage to the back office, causing you to be out of compliance with the producer; or you might have a large volume of client transaction records that need to be synchronized, increasing disk space requirements on the server machine.

### Task

**To check your synchronization status**

Using the FlexNet License Server Administrator command-line tool, run the `-status` command, as shown in this example:

```
flexnetlsadmin.bat -server licenseServer_baseURL -status
```

(See the Using the FlexNet License Server Administrator Command-line Tool chapter for details about the `-status` command.)

The output includes the `RecordsPendingSync` and `Time elapsed since last sync` items to provide a synchronization status. (In this case, two records are pending synchronization, and the last synchronization occurred 23 hours previously.)

```
Copyright (c) 2015-2017 Flexera. All Rights Reserved.

(version) Version : 2017.02
(buildVersion) Build Version : 198983
...

RecordsPendingSync : 2

Time elapsed since last sync: 23 hours
```

The `RecordsPendingSync` item might indicate that you have exceeded a metered-usage count or an unsynchronized-record limit. The `Time elapsed since last sync` item (the lowest unit of time for which is “hours”) might indicate that you have missed a scheduled synchronization.

### Synchronization From the Back Office

Another type of synchronization is synchronization of license data from the back office to the FlexNet Embedded local license server. Synchronization from the back office enables a server's license-distribution and metered-usage state to be restored from the back office after catastrophic server failure.

The following sections describe synchronization from the back office:

- Enablement
- Synchronization Process
- Viewing Restored Data
- Reviewing Synchronization Settings
Enablement

Your license server’s ability to synchronize from the back office is enabled by the software producer. If you need to change your license server’s current policy about synchronization, contact the software producer.

Synchronization Process

Once the license server is up and running, if trusted storage is not present, the server first sends a capability request to the back office to obtain its pool of licenses. The server then sends a synchronization request message to the back office to obtain the server’s license-distribution and metered-usage data required to restore the deduction records on the server. (If synchronization from the back office is enabled, the server log contains an entry stating as such.)

Once the synchronization from the back office is complete, the license server can start responding to capability requests from client devices. Moreover, server’s transaction records will be reflected in the back office with updated timestamps the next time synchronization to the back office is performed. These updated records indicate a successful restoration of the server’s license-distribution and metered-usage state.

Viewing Restored Data

When the synchronization from the back office is complete, use your license-server administrator tool to review the restored license pool and usage data:

- If using the FlexNet License Server Administrator command-line tool, see Viewing Features Installed on the License Server and Monitoring License Distribution to Clients in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- Using the FlexNet License Server Manager, see Devices View and Feature Usage View in the Using the FlexNet License Server Manager chapter.

Reviewing Synchronization Settings

Using FlexNet License Server Administrator command-line tool or FlexNet License Server Manager, whichever tool is available to you, you can review the producer-defined settings that control synchronization from the back office:

- If using the FlexNet License Server Administrator command-line tool, see Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- If using the FlexNet License Server Manager, see Properties View in the Using the FlexNet License Server Manager chapter.

For a description of these settings, see the appendix Reference: License Server Policy Settings. For the restoration of the server’s license pool, see the capability-polling settings (beginning with lfs.capability...). For the synchronization from the back office, see the regular synchronization setting lfs.syncFrom.enabled.
License Server Failover

The FlexNet Embedded local license server includes support for license server failover. Failover involves two local license servers—the main license server and the back-up license server—that work together to ensure that licenses in the enterprise remain available for serving to client devices should the main license server fail.

The two license servers in a failover configuration must be of the same version.

In a failover configuration, the main FlexNet Embedded local license server periodically sends synchronization messages to the back-up license server and the back-up license server uses this information to mirror the state of the main server.

If a client device determines that the main license server has failed—by a failure to respond to the client's capability request, for example—it can communicate with the back-up license server, which then takes over responsibility for serving the pool of licenses until the main license server is restored, or until the configured maintenance interval (as defined by the software producer) has elapsed.

The back-up license server does not synchronize to the back office or to the main license server. Therefore, data about clients served by the back-up license server during a failover period is not stored in the back office. However, once the main license server has come back online, license-distribution data collected from that point on is once again synchronized to the back office.

The following sections describe license server failover:

- Configuring Server Failover
- Editing Failover Configuration Settings
- (Optional) Automatic Registration of the Failover Pair
- Additional Failover Considerations

Configuring Server Failover

Failover configuration is actually set up on the back-up license server. Follow these procedures to complete the configuration.

Step 1: Install and Configure the License Servers

Install both the main and back-up license servers. As part of the installation process, configure both servers with basically the same local settings, with some allowable differences. For example, you can specify different PORT values for the license servers. Additionally, if specifying the BACKUP_SERVER_HOSTID setting, do so only on the main license server (see (Optional) Automatic Registration of the Failover Pair).

For more information about installing a license server and editing its local settings file—flexnetls.settings on Windows or /etc/default/flexnetls-producer_name on Linux, see Getting Started.

Step 2: Register the Main and Backup License Servers

Using the FlexNet Operations End-User Portal, register the main license server with the back-up license server as a failover pair on the back-office server. (This registration is performed by specifying the hostid of the back-up license server on the portal's Create Server page for the main license server.) The hostid for each the main and the back-up license server must be of the same type.
Note • It may be necessary to enable support for server failover in FlexNet Operations.

For more information about registering the failover pair from the Create Server page for the main license, see the online help system for the FlexNet Operations End-User Portal.

This step can be replaced with an automatic registration of the failover pair through the capability request sent from the main license server. See (Optional) Automatic Registration of the Failover Pair for details.

Step 3: Start Up the License Servers

Start up the license servers. Any license rights already mapped to the failover pair through the main license server are automatically activated on the both license servers. If you need to activate licenses manually, see Step 6: Activate Licenses on the Servers.

You can now use the administrator tool provided with the license server to update policy settings, as described next in Steps 7 and 8.

Step 4: Enable Failover Support on the Back-Up Server

Update policy settings on the back-up license server, using the provided license-server administrator tool, such as the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager:

1. Set the following license server policy settings on the back-up license server:
   - fne.syncTo.mainUri - Set to the URI of the main license server (in the format http://licenseServerHostName:port/fne/bin/capability).
   - fne.syncTo.enabled - Set to true.

2. (Optional) Set other failover-related policy settings, such as other fne.syncTo... settings or the licensing.backup.uri and the licensing.main.uri settings. (Use the format http://licenseServerHostName:port/fne/bin/capability for URI settings.)

See Editing Failover Configuration Settings for information about editing these license-server policies.

(Optional) Step 5: Edit Producer Settings on the Main Server

Configure one or both of the following license server policy settings on the main license server to include these URIs as reference information sent in capability responses to client devices. Use the format http://LicenseServerHostName:port/fne/bin/capability for either URI.

- licensing.backup.uri
- licensing.main.uri

See Editing Failover Configuration Settings for information about editing these license-server policies.
Step 6: Activate Licenses on the Servers

If no license rights were activated on the license servers at startup, perform these steps to provision the servers with licenses:

1. Start up both license servers.
2. Send one or more rights IDs in a capability request to activate identical license rights on both license servers via the current (main) license server. For information about activating license rights, see the appropriate information:
   - If using the FlexNet License Server Administrator command-line tool, see Activating License Rights on the Server in the Using the FlexNet License Server Administrator Command-line Tool chapter.
   - If using the FlexNet License Server Manager, see Offline Server Updates View in the Using the FlexNet License Server Manager chapter.

Step 7: Verify Failover Roles

Verify the failover role of each license server after the servers receive their license rights. (The roles are determined by the back office and included in the capability response.)

Verification can be performed by using the FlexNet License Server Administrator command-line tool (for example, flexnetlsadmin -server LicenseServer_baseURL -status). Additionally, the primary server log will state the failover role of the given license server.

Editing Failover Configuration Settings

Using FlexNet License Server Administrator command-line tool or FlexNet License Server Manager, whichever tool is available to you, you can edit the policy settings used to configure the main and back-up servers, as described in the previous section:

- If using the FlexNet License Server Administrator command-line tool, see Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- If using the FlexNet License Server Manager, see Properties View in the Using the FlexNet License Server Manager chapter.

For a description of these settings, see the appendix Reference: License Server Policy Settings.

(Optional) Automatic Registration of the Failover Pair

As described in Step 2: Register the Main and Backup License Servers, the license server administrator must register the main and back-up license servers as a failover pair in FlexNet Operations. One way to perform this registration is to do so manually through the FlexNet Operations End-User Portal.

Another way to perform this registration is to use the capability request sent to FlexNet Operations by the main license server at startup as a means of conveying the back-up license server’s hostid to the back-office server. FlexNet Operations then uses the hostid information in the capability request to automatically register both license servers as a failover pair, thus avoiding the extra step of having to access the End-User Portal to perform the registration.
### Task

**To set up this automatic registration process**

1. Shut down both the main and back-up license servers.

2. On the main license server, update the local license-server settings to include the BACKUP_SERVER_HOSTID setting. This value identifies the hostid of the back-up license server. It must be the same hostid type (such as “ethernet”) as the type of hostid used by the main (current) license server. See Getting Started for details about updating local settings.

3. Start up the main license server first.
   
The main license server sends a capability request containing the back-up license server's hostid to FlexNet Operations, where both servers are then registered as a failover pair.

4. Start up the back-up license server once the main license server has successfully communicated with FlexNet Operations. Both servers can now obtain license rights either through a rights ID sent in a capability request or through capability polling if licenses rights are subsequently mapped in FlexNet Operations to the registered failover pair.

#### Note

- **You must wait until the main license server has successfully started and communicated with FlexNet Operations before starting the back-up license server. Otherwise, FlexNet Operations might assign the back-up license server to the “main server” role, generating an error when FlexNet Operations then attempts to register the actual main license server.**

5. Configure the license servers as described in Step 4: Enable Failover Support on the Back-Up Server and (Optional) Step 5: Edit Producer Settings on the Main Server.

You can view details for the main and back-up server by searching for the main license server (from the Search Servers page) in the FlexNet Operations End-User Portal. The View Server page for the main license server shows both license servers registered as a failover pair.

### Additional Failover Considerations

The following lists additional considerations when enabling failover configuration:

- Since the back-up license server does not synchronize to the back office or the main license server, data about clients served by the back-up server during a failover period is permanently missing in the back office. Once the main server has come back online, the new license-distribution data can be synchronized to the back office.

- Metered-usage data is currently not supported in a failover scenario.

- The failover process might be incompatible with the type of FlexNet Embedded licensing with which your purchased product is enabled and secured. Contact the producer for details.

- Failover functionality is dependent on the clocks of the main license server and back-up license server being accurate, synchronized with each other, and in the same time zone. If any of these requirements are not met, unpredictable behavior can occur.

- The failover configuration does not support reservations.
• The failover configuration does not support feature partitions. During a failover period, feature counts will only be distributed from the default partition.

Outgoing HTTPS

The FlexNet Embedded local license server is capable of HTTPS communication with FlexNet Operations—that is, outgoing HTTPS.

For a secure connection, a truststore file containing root and intermediate certificates is required.

• If the back-office server certificate is signed by a known certificate authority (as is the case for Revenera-hosted back offices), then little or no configuration is needed on the server. See Default Server Configuration.

• If the producer has provided their own truststore, or has changed the password found in the Java cacerts file, you need to perform some configuration steps. See Server Configuration When Another Certificate Is Used for instructions. Typically, a producer provides their own truststore file when the back office is an “on-premises” version of FlexNet Operations and the server certificate has been signed by an unknown (private) certificate authority.

Default Server Configuration

In most scenarios, HTTPS will work “out of the box”. A secure connection to the back office requires access to the appropriate root and intermediate certificates for connection validation. The default behavior is to use the truststore shipped with the Java runtime environment (JRE). This is the “cacerts” file found in the lib/security directory of the JRE installation. The file is encrypted, and is installed with a password of “changeit”.

No additional configuration should be needed for Revenera-hosted FlexNet Operations instances and instances where FlexNet Operations is hosted on AWS, provided that a version of Oracle Java SE 8, OpenJDK 8, or OpenJDK 11 or later is used.

If any SSL-related errors appear in the server log, refer to the following section, Server Configuration When Another Certificate Is Used.

Server Configuration When Another Certificate Is Used

You need to override the default server configuration if either of these conditions are true:

• The producer has changed the password for cacerts.

• You need to connect to an “on-premises” installation of FlexNet Operations that is using a private certificate authority whose root certificates are not found in the cacerts file.

As always, consult the producer if you need to determine the name and location of the truststore file.

There are different ways to override the default server configuration, depending on whether you are newly installing the local license server or whether you have an older installation.
New Installations

Update the `local-configuration.yaml` file to override the default server configuration. On Linux systems, this file is generated in the `/opt/flexnetls/producer` directory during installation. On Windows, it is located in the same directory as `flexnetls.jar`.

You might need to change the password or the path to the truststore file, or both.

Edit the following settings in `local-configuration.yaml` so that they look similar to this:

```yaml
https-out:
  # Set to true to enable
  enabled: true
  # Path to truststore containing server certificate.
  truststore-path: path-to-your-truststore
  # Truststore password. You can obfuscate this with java -jar flexnetls.jar -password your-password-here
  truststore-password: your-password-here
  # Switch off if you’re having host validation problems (not recommended)
  host-verify: true
  # Set to true if you’re using self-signed certificates (not recommended)
  self-signed: false
```

The password is not required to be stored in plain text; you can obfuscate it first. The following shows an example session to obfuscate the password:

```
$ java -jar flexnetls.jar --password=abracadabra
abracadabra => OBF:1ri71v1r1v2n1ri71shq1ri71shs1ri71v1r1v2n1ri71
```

Be sure to set the `truststore-password` value to the entire obfuscated string, including the `OBF:` prefix.

For detailed instructions about editing the `local-configuration.yaml` file, see Edit "local-configuration.yaml" (Windows) and Edit "local-configuration.yaml" (Linux).

Existing Installations

If you are using an older version of the local license server, which provides access to the truststore file using a “client” configuration file, you can do either of the following:

- Update the existing “client” configuration file to use the new values. See Update the “client” Configuration File.
- Transfer the HTTPS values from the “client” configuration file to `local-configuration.yaml`. See Transferring HTTPS Values from “client” Configuration File to YAML File.

Update the “client” Configuration File

<table>
<thead>
<tr>
<th>Task</th>
<th>To update the “client” configuration file</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Update the following values in the existing “client” configuration file:</td>
</tr>
<tr>
<td></td>
<td><code>truststore.path=path-to-client-certificate</code></td>
</tr>
<tr>
<td></td>
<td><code>truststore.password=password</code></td>
</tr>
<tr>
<td>2.</td>
<td>If required, use the appropriate step to tell the license server where the “client” configuration file is located:</td>
</tr>
</tbody>
</table>
In Windows, edit the `flexnetls.settings` file to include this option to specify the path to the configuration file:

```
HTTPS_CLIENT_CONFIG=path-to-configuration-file
```

In Linux, edit following variable in the `/etc/default/flexnetls-producer_name` file, replacing the comment with the path to the configuration file:

```
HTTPS_CLIENT_FILE= empty, can replace by -https-client-configuration path
```

See Editing the Local Settings Post-Installation in the Getting Started chapter for complete instructions on editing this local settings file.

**Transferring HTTPS Values from “client” Configuration File to YAML File**

Instead of using the “client” configuration file, you can use the `https-out` setting in `local-configuration.yaml`. The following table shows how to transfer the values from the “client” configuration file to `local-configuration.yaml`. Note that the settings `truststore-path` and `truststore-password` in `local-configuration.yaml` use a hyphen instead of a dot.

**Table 5-6 •**

<table>
<thead>
<tr>
<th>“client” Configuration File Content</th>
<th>local-configuration.yaml File Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>truststore.path=path-to-client-certificate</code></td>
<td><code>https-out:</code></td>
</tr>
<tr>
<td><code>truststore.password=password</code></td>
<td><code># Set to true to enable</code></td>
</tr>
<tr>
<td></td>
<td><code>enabled: true</code></td>
</tr>
<tr>
<td></td>
<td><code># Path to truststore containing server certificate.</code></td>
</tr>
<tr>
<td></td>
<td><code>truststore-path: path-to-your-truststore</code></td>
</tr>
<tr>
<td></td>
<td><code># Truststore password. You can obfuscate this with java -jar flexnetls.jar -password your-password-here</code></td>
</tr>
<tr>
<td></td>
<td><code>truststore-password: your-password-here</code></td>
</tr>
<tr>
<td></td>
<td><code># Switch off if you're having host validation problems (not recommended)</code></td>
</tr>
<tr>
<td></td>
<td><code>host-verify: true</code></td>
</tr>
<tr>
<td></td>
<td><code># Set to true if you're using self-signed certificates (not recommended)</code></td>
</tr>
<tr>
<td></td>
<td><code>self-signed: false</code></td>
</tr>
</tbody>
</table>

**Incoming HTTPS**

In most cases, communication between client devices and the FlexNet Embedded local license server need not be secure. However, in deployments where Internet security is required, the local license server can be configured to support exchanges with the client devices over a secure HTTPS connection.

Best practice is to offload this HTTPS protocol work to a separate device that specializes in this type of processing. However, if this practice is not feasible or desirable, perform these steps to enable the license server to support incoming HTTPS connections.
Step 1: Obtain Certificate

You will require a “server” certificate from a certificate authority (CA) such as Verisign. This certificate must be in Java keystore format (not a PEM text file) and secured by a password. Your certificate provider will have instructions on how to request it in this format (or convert to it.)

Step 2: Enable Access to “server” Certificate

Access to the “server” certificate can be provided in different ways, depending on whether you are newly installing the local license server or whether you have an existing installation.

New Installations

For new installations of the local license server, specify the location of the keystore file holding the “server” certificate in the local-configuration.yaml file using the https-in parameter. On Linux systems, the local-configuration.yaml file is generated in the /opt/flexnetls/producer directory during installation. On Windows, it is located in the same directory as flexnetls.jar.

You might need to change the password or the path to the truststore file, or both.

Edit the following settings in local-configuration.yaml so that they look similar to this:

```yaml
https-in:
  # Set to true to enable
  enabled: true
  # HTTPS listening port
  port: 1443
  # Path to keystore
  keystore-path: path-to-your-keystore
  # Keystore password. You can obfuscate this with
  keystore-password: your-password-here
```

Note the following:

- The password is not required to be stored in plain text; you can obfuscate it first. The following shows an example session to obfuscate the password:

  ```bash
  $ java -jar flexnetls.jar --password=abracadabra
  abracadabra => OBF:1ri71v1r1v2n1r171shq1r171shs1r171v1r1v2n1r17
  ```

  Be sure to set the keystore-password value to the entire obfuscated string, including the OBF: prefix.

- The standard HTTPS port is 443 (as defined by the HTTPS protocol)

- Best practice on Linux is to use a port number greater than 1024 to avoid issues with possibly having to obtain extra privileges to use a lesser port number. Should port 443 be required at a customer enterprise, you can always use iptables to redirect the port from 443.

- Ensure that a firewall is not blocking the license server.

For detailed instructions about editing the local-configuration.yaml file, see Edit “local-configuration.yaml” (Windows) and Edit “local-configuration.yaml” (Linux).
Existing Installations

If you are using an existing installation of the local license server, which provides access to the "server" certificate using a "server" configuration file, you can do either of the following:

- Update the existing "server" configuration file to use the new values. See Update the "server" Configuration File.
- Transfer the HTTPS values from the "client" configuration file to local-configuration.yaml. See Transferring HTTPS Values from "server" Configuration File to YAML.

Update the "server" Configuration File

Task To update the "server" configuration file

1. Update the following values in the existing "server" configuration file:

   ```
   keystore.path=path-to-server-certificate
   keystore.password=password
   https.port=1443
   ```

2. If required, use the appropriate step to tell the license server where the "server" configuration file is located:

   - In Windows, edit the flexnetls.settings file to include this option to set the path to the configuration file:

     ```
     HTTPS_SERVER_CONFIG=path-to-configuration-file
     ```

   - In Linux, edit following variable in the /etc/default/flexnetls-producer_name file, replacing the comment with the path to the configuration file:

     ```
     HTTPS_SERVER_FILE=    #empty, can replace by -https-server-configuration path
     ```

See Editing the Local Settings Post-Installation in the Getting Started chapter for complete instructions on editing the local settings file.
Transferring HTTPS Values from “server” Configuration File to YAML

Instead of using the “server” configuration file, you can use the https-in setting in local-configuration.yaml. The following table shows how to transfer the values from the “server” configuration file to local-configuration.yaml. Note that the settings keystore-path and keystore-password in local-configuration.yaml use a hyphen instead of a dot.

Table 5-7 -

<table>
<thead>
<tr>
<th>“client” Configuration File Content</th>
<th>local-configuration.yaml File Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystore.path=path-to-server-certificate</td>
<td>https-in:</td>
</tr>
<tr>
<td>keystore.password=password</td>
<td># Set to true to enable</td>
</tr>
<tr>
<td>https.port=1443</td>
<td>enabled: true</td>
</tr>
<tr>
<td></td>
<td># HTTPS listening port</td>
</tr>
<tr>
<td></td>
<td>port: 1443</td>
</tr>
<tr>
<td></td>
<td># Path to keystore</td>
</tr>
<tr>
<td></td>
<td>keystore-path: path-to-your-keystore</td>
</tr>
<tr>
<td></td>
<td># Keystore password. You can obfuscate this with java -jar flexnetls.jar -password your-password-here</td>
</tr>
<tr>
<td></td>
<td>keystore-password: your-password-here</td>
</tr>
</tbody>
</table>

Step 4: Define Scope of HTTPS Communications

At this point, you should have a license server that is operating in both HTTP and HTTPS modes. The final step is to define the scope of HTTPS usage for your license server. For example, maybe you want to run the FlexNet Embedded local license server entirely in HTTPS mode. To enable this scope, switch off the HTTP listening port by changing the PORT value to zero (0) in the local-configuration.yaml file (Windows and Linux), the flexnetls.settings file (in Windows) or the /etc/default/flexnetls-producer_name file (in Linux). See Editing the Local Settings Post-Installation in the Getting Started chapter for complete instructions on editing the local settings file.

Proxy Support for Communication with the Back Office

For security purposes, your corporate network might require a networking proxy as an intermediary between your internal systems and the Internet. To address this requirement, the FlexNet local license server supports communication with FlexNet Operations through an HTTP proxy. The local license server depends on Java Runtime to provide networking services.

Note • The FlexNet Embedded local license server supports both authenticating and non-authenticating HTTP proxies.

The following sections help you set up the license server to use this proxy:

- Configuring the License Server for Proxy Support
• Obfuscating the Proxy Password

Configuring the License Server for Proxy Support

Use this information to configure the license server for HTTP communications with the back office through a networking proxy:

• Proxy Configuration Basics
• Supported Proxy Parameters
• Parameter Format

Proxy Configuration Basics

To configure the license server to use a networking proxy when communicating with the back office, you must define a set of proxy parameters in the server’s local settings file on the server (flexnetls.settings on Windows or /etc/default/flexnetls-producer_name on Linux). These parameters are passed to the Java Runtime system to identify the HTTP proxy.

You specify these parameters for the EXTRA_SYSPROPERTIES setting, which you must “uncomment” if it is currently “commented” in this file.

For complete instructions on editing the local settings file, refer to the Configuring, Installing, and Starting the Local License Server and Editing the Local Settings Post-Installation sections in the Getting Started chapter.

Supported Proxy Parameters

The following proxy parameters are supported:

• http.proxyHost
• http.proxyPort
• http.proxyUser
• http.proxyPassword

Consider the following:

• If not included, the http.proxyPort parameter defaults to 80.
• The http.proxyUser and http.proxyPassword parameters are optional. They are required only if the proxy requires authentication.
• Best practice is to use an obfuscated password instead of a plain-text value for http.proxyPassword. See Obfuscating the Proxy Password for encryption details.

Parameter Format

Follow these instructions when adding the proxy parameters to the EXTRA_SYSPROPERTIES setting in the local settings file:
• Each parameter specified for EXTRA_SYSPROPERTIES must use the following Java system syntax:
  \(-D\text{property}=value\)

• The parameters are separated from each other with a space, and the entire set of parameters are enclosed in double quotations. The following shows an example EXTRA_SYSPROPERTIES setting with proxy parameters specified:

  ```
  EXTRA_SYSPROPERTIES="-Dhttp.proxyHost=10.90.3.133 -Dhttp.proxyPort=3128 -Dhttp.proxyUser=user1a -Dhttp.proxyPassword=user1apwd35"
  ```

  Even if you specify only a single parameter, it must be enclosed in double quotations:

  ```
  EXTRA_SYSPROPERTIES="-Dhttp.proxyHost=10.90.3.133"
  ```

### Obfuscating the Proxy Password

Best practice is to obfuscate the proxy password and use this encrypted value for the \text{http.proxyPassword} parameter.

The following is an example session showing both the command used to obfuscate a plain-text password (in this case, user1apwd35) and the resulting obfuscated string:

```
$ java -jar flexnetls.jar --password=user1apwd35
user1apwd35 => OBF:1ri71v1r1v2n1ri71shq1ri71shs1ri71v1r1v2n1ri7
```

Use the entire obfuscated string, including its OBF: prefix, for the \text{http.proxyPassword} value, as shown in this example:

  ```
  -Dhttp.proxyPassword=OBF:1ri71v1r1v2n1ri71shq1ri71shs1ri71v1r1v2n1ri7
  ```

### Trusted Storage Backup and Restoration

If the producer has enabled your local license server to perform backups of trusted storage, you can restore the last backup copy, should trusted storage become corrupted, without having to contact FlexNet Operations. The following sections describe the trusted-storage backup and restoration process:

- Overview of the Backup Process
- Running a Trusted Storage Restoration

### Overview of the Backup Process

You can restore trusted storage from a backup copy only if the producer has enabled trusted-storage backups to occur on the license server. If you need information about enabling these backups, contact the producer.

The following provides an overview of the backup process for trusted storage on the license server:

- **One.** The license server automatically initiates a trusted-storage backup whenever the following events occur:
  - A capability response from the back office is processed
  - A quantity of capability requests from FlexNet Embedded clients have been processed (based on producer-defined thresholds for the license server)
• **Two.** Once the backup is initiated, the license server suspends the processing of capability responses and requests until the backup completes.

• **Three.** If the backup is successful, the backed-up trusted storage data is stored in the file `tsBackup/flexnetls_licenses.mv.db.ts`, located in the directory containing trusted storage. (A failed backup can be an indicator that trusted storage is corrupt.)

The license server maintains only one copy of the backed-up data at any given time.

## Running a Trusted Storage Restoration

If trusted storage is determined to be corrupt, you can restore the last successful backup of trusted storage.

### Task To restore trusted storage from the backup

1. Stop the license server.

2. Use the `-restore-database` argument with the command that runs the license server script to restore trusted storage.

   The `path` value used in this command is the path to the trusted-storage backup file `flexnetls_licenses.mv.db.ts`, located under `tsBackup` in the trusted storage directory. Note that the explicit path values listed in this step use the default value (`$(base.dir)`) for the license server policy `server.trustedStorageDir`, which defines the trusted storage directory. You might need to adjust the path based on this policy definition on your license server. (See Reference: License Server Policy Settings.)

   If more than one user log-in will be doing a restore, `server.trustedStorageDir` should be set to a path other than `$({base.dir})`. See the producer for updating this policy.

   **On Windows**

   - In console mode, enter:
     ```
     flexnetls.bat -restore-database path
     ```
     where `path` is `C:\Users\user_name\flexnetls\producer_name\tsBackup\flexnetls_licenses.mv.db.ts`.

   - As a service, enter:
     ```
     flexnetls -restore-database path
     ```
     where `path` is `C:\Windows\ServiceProfiles\NetworkService\flexnetls\producer_name\tsBackup\flexnetls_licenses.mv.db.ts`.

   **On Linux**

   - In console mode, enter:
     ```
     ./flexnetls -restore-database path
     ```
     where `path` is `/var/opt/flexnetls/producer_name/tsBackup/flexnetls_licenses.mv.db.ts`.

   - As a service (Systemd or SysV), use the same command as used for console mode.

3. Restart the license server.
Best practice is never to delete any files in directory containing the corrupted trusted storage; allow the restoration process handle the fix. However, if for some reason you need to remove the corrupted data, delete only flexnetls_licenses.mv.db.

One basic method you can use to verify that trusted storage has been restored to its previous uncorrupted state is to check the feature counts in trusted storage, using the license server administrator tool provided by the producer.

**Public Key Upload**

Depending on the type of FlexNet Embedded licensing with which your purchased product is enabled, you might be required to upload a public key from the client to the license server. The producer will inform you whether this upload is necessary and will provide either the public key itself or the instructions on how to obtain it. You can upload this key using the FlexNet Embedded License Server Administrator command-line tool or the FlexNet License Server Manager or a tool or method provided by the producer. The producer will direct you as to which means to use.

In general, the public key is an RSA key (2048-bit DER-encoded) in an octet-stream format. You must have license-server administrator credentials to upload it.
Managing a CLS Instance

If you are using a CLS instance (a Cloud Licensing Service license server), which is hosted in the Cloud, you must use the FlexNet Operations End-User Portal to manage the server. For management instructions, you can access the portal’s help system, or refer to this chapter, which highlights important license-server administration procedures extracted from the portal’s help system.

For more information about the CLS instance and its relationship to the FlexNet Embedded local license server, see Local License Servers vs. License Servers Hosted in the Cloud in the Introduction chapter.

This chapter covers the following procedures for managing the CLS instance:

- Attributes of the CLS Instance
- Searching for a CLS Instance
- Working with CLS Instances
- Creating a CLS Instance
- Managing Administrative Security

Attributes of the CLS Instance

The following attributes are used when searching for, creating, and managing CLS instance.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment</td>
<td>The deployment type for the license server. For a CLS instance, this is Cloud.</td>
</tr>
</tbody>
</table>
Searching for a CLS Instance

An existing CLS instance is often most easily located by performing a quick search based on one or more attributes of the license server.

Table 6-1 • CLS Instance Attributes (cont.)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>License Server ID</td>
<td>The unique identity for the license server.</td>
</tr>
<tr>
<td></td>
<td>License Server ID is not shown when creating license servers with the Cloud deployment type. In this case, FlexNet Operations automatically generates a License Server ID. However, you can see this attribute on the View Server page for the CLS instance.</td>
</tr>
<tr>
<td>Auto Provisioned</td>
<td>Indicates whether the producer created the CLS instance through the auto-provision feature in FlexNet Operations.</td>
</tr>
<tr>
<td>Model</td>
<td>The device model used to create the CLS instance.</td>
</tr>
<tr>
<td>Site Name</td>
<td>A label used to distinguish between different license servers if your organization has multiple CLS instances.</td>
</tr>
<tr>
<td>Organization</td>
<td>The name of your organization specified as Organization ID (or Owner ID).</td>
</tr>
<tr>
<td>Server Status</td>
<td>The status of the license server as either ACTIVE, RETURNED, or OBSOLETE.</td>
</tr>
</tbody>
</table>

Searching for a CLS Instance

An existing CLS instance is often most easily located by performing a quick search based on one or more attributes of the license server.

Task

To search for a CLS instance

1. Click Devices > Devices. This opens the Devices page.
2. (Optional) To filter the devices list to show only CLS instances:
   a. On the Devices page, hover the cursor over the filter icon to display the filter selection list.
   b. On the selection list, select Cloud License Servers for the device Type, and select one or more Status types.
   c. Click Apply to filter the devices according to your selections.
3. Perform a simple or advanced search for the license server.

Simple search

a. On the Devices page, click the drop-down list next to the filter icon, and select the search criterion.

b. Type the search string in the text field next to the drop-down.

c. Click Search.
a. On the Devices page, click the + icon (next to the Search button) to open the Advanced Search window.

b. Type the search criterion in the attribute text fields.

c. Click Search.

When you select a CLS instance from the Devices list, its View Server page opens. From this page, you can view the server's attributes and access options to perform server management activities—such as viewing server history or moving a server. See Working with CLS Instances for further information.

## Working with CLS Instances

From the View Server page for a CLS instance, you can view the server's attributes or access options to manage the server. The following topics address the server-management activities you can perform.

For instructions on accessing a license server's View Server page, see the previous section Searching for a CLS Instance.

### Table 6-2 • Overview of CLS Management Tasks

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View the History of a License Server</td>
<td>Shows how to view a history of the license server's transactions.</td>
</tr>
<tr>
<td>View Served Clients of a CLS Instance</td>
<td>Shows how to view the clients currently served by the license server.</td>
</tr>
<tr>
<td>Map Entitlements to a CLS Instance</td>
<td>Explains how to map add-ons to the license server.</td>
</tr>
<tr>
<td>Remove Licenses from a CLS Instance</td>
<td>Explains how to remove add-ons from the license server.</td>
</tr>
<tr>
<td>Move a CLS Instance</td>
<td>Describes how to change the owner of the license server.</td>
</tr>
</tbody>
</table>

## View the History of a License Server

You can view the history of a specific CLS instance through its View Server page. The Server History page lists all the events that have occurred in the server’s history—from its creation, to ownership changes, to activation requests.

<table>
<thead>
<tr>
<th>Task</th>
<th>To view the history of a CLS instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From the View Server page for the CLS instance, click View &gt; View History.</td>
</tr>
<tr>
<td></td>
<td>The End-User Portal opens the Server History page.</td>
</tr>
</tbody>
</table>
View Served Clients of a CLS Instance

Through the View Server page for the CLS instance, you can view the client devices currently served by the instance.

<table>
<thead>
<tr>
<th>Task</th>
<th>To view the served devices of a license server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From the View Server page for the CLS instance, click View &gt; View Served Devices.</td>
</tr>
<tr>
<td></td>
<td>The End-User Portal opens the Devices page, showing the a list of client devices currently served by the CLS instance.</td>
</tr>
</tbody>
</table>

Map Entitlements to a CLS Instance

CLS instances are often provisioned with pre-built capabilities. However, these servers also permit the addition of new features or more capacity via additional entitlements.

When the current owner of a CLS instance is entitled to additional capabilities, you can manually map these entitlements to the server through the View Server page.

<table>
<thead>
<tr>
<th>Task</th>
<th>To map an entitlement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>From the View Server page for the CLS instance, click Action &gt; Map Entitlements. The End-User Portal opens the Map Entitlements page. The table on this page lists the entitlements that can be mapped to the current CLS instance.</td>
</tr>
<tr>
<td>2.</td>
<td>In the table, enter a count in Qty to Add to specify the quantity for each entitlement you want to map.</td>
</tr>
<tr>
<td>3.</td>
<td>(Optional) For any entitlement that is enabled, you can override its Expiration, as shown in the table, with a shorter-term expiration. Click the pencil icon next to the expiration date, and select a new date from the displayed calendar. (After the change is saved, the new expiration appears in the Licenses table on the View Server page and is denoted by an asterisk in the table on the device's Map Entitlements page.)</td>
</tr>
<tr>
<td>4.</td>
<td>Click Save.</td>
</tr>
<tr>
<td></td>
<td>The End-User Portal maps the entitlements to the current CLS instance and returns to server’s View Server page.</td>
</tr>
</tbody>
</table>

Remove Licenses from a CLS Instance

Licenses already mapped to a CLS instance can be manually removed through its View Server page.

Note - Metered-model licenses cannot be removed.
Task **To remove licenses**

1. From the View Server page for the CLS instance, click **Action > Remove Licenses**. The End-User Portal opens the **Remove Entitlements** page. The **Licenses on Device** table on this page lists licenses that can be removed from the current server.

2. In the **Licenses on Device** table, enter a count in **Qty to Remove** for those licenses you want to remove.

3. Click **Save**.

The End-User Portal removes the specified license counts for the current CLS instance and returns to the server’s **View Server** page.

**Move a CLS Instance**

A CLS instance that has not yet been assigned to a customer organization can be moved from your organization to a related organization. For example, the parent organization can move the license server from one of its child organizations (a regional division, for instance) to another child organization.

When a CLS instance is moved, any entitlements that belong to that license server are moved as well. The result of such a move on the receiving organization is that new entitlements are added or existing entitlements that match moved entitlements are incremented.

This action is performed through the **View Server** page for the CLS instance.

Task **To move a license server**

1. From the View Server page for the CLS instance, click **Action > Move Server**. The End-User Portal opens the **Move Server** page.

2. For **To Organization**, select the organization to receive the CLS instance.

3. Click **Submit**.

The End-User Portal changes the owner of the license server and returns to the server’s **View Server** page.

**Creating a CLS Instance**

In some cases, if your enterprise is large, it might be desirable to have multiple CLS instances. You can define new CLS instances in the End-User Portal. And, once defined, a CLS instance can submit its own capability requests to FlexNet Operations and serve its own licenses to served clients.

For descriptions of the attributes used to create a CLS instance, see **Attributes of the CLS Instance**.

Task **To create a CLS instance**

1. From the End-User Portal menu bar, click **Devices > Create Device**. The **Device | New Device** page opens.

2. Type a device **Name**.
3. Select the **Runs license server?** option. For **Deployment**, select **Cloud**.

4. Optionally, select a server **Model** from the drop-down list.

5. Optionally, select an **Organization** from the drop-down list. The organization identifies the customer or partner organization of the server.

6. Optionally, type a value for **Site Name**.

7. Click **Save**.

The End-User Portal creates the new CLS instance and opens its **View Server** page.

---

### Managing Administrative Security

If administrative security is enabled for your CLS instance, a default license-server administrator account is created for you when the CLS instance is created. This account gives you access to a full range of administrative functionality on the CLS instance, including privileges to create and manage other enterprise user accounts. For details, see Managing Administrative Security on a Local License Server or CLS Instance in the Getting Started chapter.

This section highlights basic tasks that you perform as the license server administrator:

- **Set Your Administrator Password**
- **Create and Manage Other User Accounts**
- **Manage the License Server**

---

### Set Your Administrator Password

You are responsible for setting the password for your default administrator account before you access the CLS instance. (The default administrator name for this account is "admin", which is case-sensitive.)

#### Task

**To set your password**

1. From the **View Server** page for the CLS instance, click **Action > Set Password**.

2. Provide your password and re-enter it to confirm. (The password must meet the criteria specified in **Credential Requirements** in the Getting Started chapter.)

---

### Create and Manage Other User Accounts

As a license server administrator, you can use your credentials to create other user accounts—either for enterprise users or for another administrator. You can perform these operations using the FlexNet License Server Administrator command-line tool (see Using the FlexNet License Server Administrator Command-line Tool) or the custom administrative tool provided by the producer.
Manage the License Server

Use the FlexNet License Server Administrator command-line tool or the producer's custom administrative tool to perform other administrative tasks such as creating and managing reservations, suspending the server, and querying features and information about licenses served to FlexNet Embedded clients. For instructions, see Using the FlexNet License Server Administrator Command-line Tool or the producer's instructions.
Reference: License Server Policy Settings

Use the following chart as reference to the license server policy settings provided by the producer to control the various operations that your local license server can perform, such as synchronization to and from the back office, licensing distribution, logging, license server failover, and others.

The **Editable?** column indicates whether the license server administrator can update the given policy.

**Note** • These policies are of interest to the administrator of a local license server. The administrator of a CLS instance cannot edit license server policies.

For more information about the process of overriding settings, go to the appropriate link:

- Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.
- Properties View in the Using the FlexNet License Server Manager chapter. (As a reference, the chart shows the field name equivalents used in FlexNet License Server Manager for various policy settings in `producer-settings.xml`.)

### Table A-1 • License Server Policy Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>database.backup-enabled</code></td>
<td>The property that determines whether the license server takes a backup of trusted storage at given times and stores it on the server. Should trusted storage become corrupt, the license server administrator can then restore it from the backup without contacting the back office. The default is <code>false</code>. See Trusted Storage Backup and Restoration for details.</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Licensing Policies**
### Table A-1 - License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>licensing. clientExpiryTimer</td>
<td>The frequency of checks for expired features on clients. (An expired feature is one whose borrow interval has expired or that has reached its expiration date as defined in the back office.) Expired features found during a given check are returned to the license-server feature pool. The frequency value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. The default frequency for these checks is 2s. Consider increasing this value if the current frequency is interfering with capability-request processing or with overall throughput, particularly when features have large borrow intervals. (The minimum value is 1s.)</td>
<td>Yes</td>
<td>Client Expiry Timer Interval</td>
</tr>
<tr>
<td>licensing. dropClientEnforceDelay</td>
<td>The delay that is enforced between client deletion requests. This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. This setting can also be used to disallow the deletion of clients; in this case, set the value DROP_CLIENT_DISALLOWED. (Default is 0s, meaning deleting client records is allowed and there is no enforced delay between deletions.)</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>licensing. responseLifetime</td>
<td>The lifetime of a served-license response on the client. This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. If this value is 0 (zero), the response has an unlimited lifetime. (Default is 1d.)</td>
<td>No</td>
<td>Response lifetime</td>
</tr>
<tr>
<td>licensing.allow VirtualClients</td>
<td>The property that determines whether virtual client devices are allowed to obtain licenses. Default is true.</td>
<td>No</td>
<td>Allow virtual clients</td>
</tr>
<tr>
<td>licensing.allow VirtualServer</td>
<td>The property that determines whether the license server is allowed to run on a virtual host. Default is true.</td>
<td>No</td>
<td>Allow virtual server</td>
</tr>
</tbody>
</table>
The time unit to which the borrow interval used by the license server rounds up. Valid values include day, hour, minute, or second. (The default is second.)

For example, if the borrow interval (which is always expressed in seconds) is 60 seconds, and the borrow granularity is day, then a license issued at 5:05:01 PM expires at 11:59:59 PM—the borrow interval (5:06:01 PM) rounded to the end of the nearest day. Likewise, if granularity is minute, expiration is at 5:06:59 PM. If the granularity is second, expiration is 5:06:01 PM.

This setting is used for those client devices that do not specify one.

**Note** • For FlexNet Embedded client SDKs released before version 4.0, the granularity is always “day”, regardless of this setting.

The borrow interval for served licenses. This server borrow interval is only considered if the back office does not specify a borrow interval for a feature within the license model. The current version of FlexNet Operations mandates that a borrow interval (also referred to as the feature borrow interval) be specified for new license models.

This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 1w.)

For information on how to determine the effective borrow interval, see licensing.borrowIntervalMax.
Appendix A  Reference: License Server Policy Settings

Table A-1 • License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
</table>
| licensing.borrowIntervalMax | Restricts the borrow period of the clients. This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. Default is NOT_CONFIGURED (0). This is also referred to as the admin borrow interval. The following helps you determine the effective borrow interval.  
  • If the feature borrow interval has been set in the back office, the borrow interval is the lowest of the following values:
    • feature borrow interval (set in the back office)
    • client borrow interval (set in a client capability request)
    • admin borrow interval (set using licensing.borrowIntervalMax)
  • If the feature borrow interval has not been set in the back office, the borrow interval is the lowest of the following values:
    • server borrow interval (defined in producer-settings.xml by the property licensing.borrowInterval)
    • client borrow interval (set in a client capability request)
    • admin borrow interval (set using licensing.borrowIntervalMax)

   A feature’s current borrow expiration can never exceed the final expiration time for that feature. In addition, a borrow-interval granularity may be applied to the effective borrow interval.

   This parameter cannot be used for metered features. |
| licensing.renewInterval | The default renew interval is set as a percentage of the effective borrow interval. This value specifies how often—if ever—the client may attempt to recontact the local license server. Successful contact extends the expiration based on the effective borrow period (in other words, the timer for the effective borrow interval is restarted). If set to zero, the renew interval is at client discretion. (Default is 15.) |

  **Important** • This specification by itself does not lead to enforcement. The client-side APIs must extract this value from the license server capability response and take appropriate action.  

  For information on how to determine the effective borrow interval, see licensing.borrowIntervalMax.
### Table A-1 • License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>licensing.hostId Validation Interval</td>
<td>The frequency with which the license server validates that its host ID has not changed. This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If this value is 0 (zero), validation is disabled. Default is 2m.</td>
<td>No</td>
<td>Host ID validation interval</td>
</tr>
</tbody>
</table>
| licensing.default TimeZone | Defines the time zone that will be applied when determining a feature's expiry date, start date, and issue date. Valid values:  
  • UTC—If UTC is set, a feature's start date is the start of the specified day in Coordinated Universal Time (UTC). Equally, a feature will expire at the end of the day of the configured expiry date in UTC time. This is the default value.  
  • SERVER—If SERVER is set, a feature's start date is the start of the specified day in the server's default time zone. Equally, a feature will expire at the end of the day of the configured expiry date in the server's default time zone. | No        | N/A                              |
| licensing.security.json.enabled | The option that enables (true) or disables (false) security for JSON capability exchanges. Contact the producer to determine whether this policy applies to the licensed product you are using and whether the policy should be enabled. (Default is true.) | Yes       | N/A                              |
| licensing.backup.uri | (Defined on back-up or main license server in a failover configuration; optional) The URI of the back-up license server to be included as reference information in the capability response to the client device. Use the following format:  
  http://server:port/fne/bin/capability  
  where server:port is the back-up license server's name and port number, as in:  
  http://22.22.2.222:7070/fne/bin/capability | Yes       | Backup URI                       |
| licensing.main.uri | (Defined on back-up or main license server in a failover configuration; optional) The URI of the main server to be included as reference information in the capability response to the client device. Use the following format:  
  http://server:port/fne/bin/capability  
  where server:port is the main license server's name and port number, as in:  
  http://11.11.1.111:7070/fne/bin/capability | Yes       | Main URI                         |

### License Server Settings
### Table A-1 - License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.trusted StorageDir</td>
<td>The directory in which trusted storage resides. (Default is <code>${base.dir}</code>, which points to the <code>flexnetls/producer_name</code> folder in the service’s or user’s home directory.)</td>
<td>No</td>
<td>Trusted storage directory</td>
</tr>
<tr>
<td>server.accessLog Pattern</td>
<td>The format to use for the access log name. (Default is <code>access_yyyy_mm_dd.request.log</code>.) To disable the creation of the access log, set <code>server.accessLogPattern=none</code>.</td>
<td>No</td>
<td>Access log pattern</td>
</tr>
<tr>
<td>server.publisher DefinedHostId. policy</td>
<td>The property that determines whether to enable support for the use of a producer-defined hostid to identify the license server. To enable support, use the value <code>STRICT</code>. (Default is <code>false</code>, meaning support for this feature is disabled.)</td>
<td>No</td>
<td>Publisher defined hostid policy</td>
</tr>
<tr>
<td>server.extended HostId.enabled</td>
<td>The property that enables support for the use of extended hostids to identify the license server. (Default is <code>true</code>.)</td>
<td>No</td>
<td>Extended hostid enabled</td>
</tr>
<tr>
<td>server.forceTS ResetAllowed</td>
<td>The property that determines whether trusted storage can be reset when unsynchronized data still exists on the license server. (Default is <code>false</code>.)</td>
<td>No</td>
<td>TS force reset</td>
</tr>
<tr>
<td>server.backup Maintenance. interval</td>
<td>(Defined on back-up license server in a failover configuration; required) The maximum amount of time that the back-up server can serve licenses in a failover event. This value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. If this value is set to 0, the back-up license server will serve licenses for an unlimited time while in failover mode. (Default is <code>3d</code>).</td>
<td>No</td>
<td>Backup maintenance interval</td>
</tr>
<tr>
<td>server.sync Compatibility</td>
<td>(Used for migration from the FlexNet Embedded server application) The property that enables proper conversion of time units used for synchronization to and from the back office during the migration from the FlexNet Embedded server application to the FlexNet Embedded local license server. (Default is <code>false</code>).</td>
<td>No</td>
<td>Sync compatibility</td>
</tr>
</tbody>
</table>

**Back Office URL**

| Ifs.url | The URL for back office to which the license server sends capability requests and synchronization data. The property is required for the online deployment model of the license server. | No | N/A |

### Policies for Polling Back Office for License Updates
The property that determines whether capability-request polling is enabled. If polling is enabled, a capability request is sent to the back office periodically to update the license server’s license rights. This property is used for the online deployment model of the license server. (Default is true.)

The amount of time between polling sessions to the back office. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 1d; minimum is 10s.)

The number of polling attempts allowed if the initial attempt fails. (Default is 3.)

The amount of time between polling attempts, if the initial attempt fails. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 30s; minimum is 1s.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>lfs.capability. enabled</td>
<td>The property that determines whether capability-request polling is enabled. If polling is enabled, a capability request is sent to the back office periodically to update the license server’s license rights. This property is used for the online deployment model of the license server. (Default is true.)</td>
<td>No</td>
<td>Capability polling enabled</td>
</tr>
<tr>
<td>lfs.capability. repeats</td>
<td>The amount of time between polling sessions to the back office. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 1d; minimum is 10s.)</td>
<td>Yes</td>
<td>Capability polling interval</td>
</tr>
<tr>
<td>lfs.capability. retryCount</td>
<td>The number of polling attempts allowed if the initial attempt fails. (Default is 3.)</td>
<td>Yes</td>
<td>Capability polling retry count</td>
</tr>
<tr>
<td>lfs.capability. retryRepeats</td>
<td>The amount of time between polling attempts, if the initial attempt fails. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 30s; minimum is 1s.)</td>
<td>Yes</td>
<td>Capability polling retry interval</td>
</tr>
</tbody>
</table>

Policies for Synchronizing to Back Office
The property that determines whether synchronization to the back office is enabled. This property should be viewed in combination with `lfs.syncTo.includeAll`:

- `lfs.syncTo.enabled=true` and `lfs.syncTo.includeAll=true`: (Online synchronization) This mode collects all historical client actions in the synchronization history and uploads this data to the back office as part of the synchronization.
- `lfs.syncTo.enabled=true` and `lfs.syncTo.includeAll=false`: (Online synchronization) This mode collects only the current state for each active client device at the point of synchronization and uploads this data to the back office.
- `lfs.syncTo.enabled=false` and `lfs.syncTo.includeAll=true`: (Offline synchronization) This mode collects all historical and current client actions. This data is retained on the license server until the offline synchronization tools are run (see Offline Synchronization to the Back Office).
- `lfs.syncTo.enabled=false` and `lfs.syncTo.includeAll=false`: No synchronization data is collected (synchronization is disabled). Client data is deleted from the license server as soon as the client expires. (Default is false.)

Table A-1 • License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
</table>
| `lfs.syncTo.enabled` | The property that determines whether synchronization to the back office is enabled. This property should be viewed in combination with `lfs.syncTo.includeAll`:  
- `lfs.syncTo.enabled=true` and `lfs.syncTo.includeAll=true`: (Online synchronization) This mode collects all historical client actions in the synchronization history and uploads this data to the back office as part of the synchronization.  
- `lfs.syncTo.enabled=true` and `lfs.syncTo.includeAll=false`: (Online synchronization) This mode collects only the current state for each active client device at the point of synchronization and uploads this data to the back office.  
- `lfs.syncTo.enabled=false` and `lfs.syncTo.includeAll=true`: (Offline synchronization) This mode collects all historical and current client actions. This data is retained on the license server until the offline synchronization tools are run (see Offline Synchronization to the Back Office).  
- `lfs.syncTo.enabled=false` and `lfs.syncTo.includeAll=false`: No synchronization data is collected (synchronization is disabled). Client data is deleted from the license server as soon as the client expires. (Default is false.) | No | Synchronization to backoffice enabled |
| `lfs.syncTo.pagesize` | The maximum number of client records to include in a synchronization message to the back office. A smaller page size limits the memory overhead at the expense of having multiple synchronization transactions. (Default is 100; minimum is 10; maximum is 256.) | Yes | Synchronization to backoffice page size |
| `lfs.syncTo.threads` | The number of parallel threads allocated to handle the synchronization of metered-usage and license-distribution data to the back office. (Default is 1.) | Yes | N/A |
| `lfs.syncTo.repeats` | The amount of time between synchronization sessions to the back office. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 5m; minimum is 10s.) | Yes | Synchronization to backoffice interval |
| `lfs.syncTo.retry Count` | The number of synchronization attempts to the back office allowed when an initial attempt fails. (Default is 4.) | Yes | Synchronization to backoffice retry count |
Appendix A  Reference: License Server Policy Settings

Table A-1 • License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>lfs.syncTo.retry</td>
<td>The amount of time between synchronization attempts when an initial attempt fails. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 5m; minimum is 1s.)</td>
<td>Yes</td>
<td>Synchronization to backoffice retry interval</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lfs.syncTo.delay</td>
<td>At license server startup, the amount of time the server should wait before initiating a synchronization session to the back office. (Default is 2s; minimum is 2s.)</td>
<td>Yes</td>
<td>Synchronization to backoffice delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lfs.syncTo.includeAll</td>
<td>The property that determines whether historical license-distribution data for concurrent features is collected and sent to the back office as part of the synchronization. This property should be viewed in combination with lfs.syncTo.enabled:</td>
<td>No</td>
<td>Synchronization to backoffice include historical data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• lfs.syncTo.enabled=true and lfs.syncTo.includeAll=true: (Online synchronization) This mode collects all historical client actions in the synchronization history and uploads this data to the back office as part of the synchronization.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• lfs.syncTo.enabled=true and lfs.syncTo.includeAll=false: (Online synchronization) This mode collects only the current state for each active client device at the point of synchronization and uploads this data to the back office</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• lfs.syncTo.enabled=false and lfs.syncTo.includeAll=true: (Offline synchronization) This mode collects all historical and current client actions. This data is retained on the license server until the offline synchronization tools are run (see Offline Synchronization to the Back Office).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• lfs.syncTo.enabled=false and lfs.syncTo.includeAll=false: No synchronization data is collected (synchronization is disabled). Client data is deleted from the license server as soon as the client expires. (Default is true.)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Policies for Synchronizing from Back Office

lfs.syncFrom.enabled The property that determines whether license-recovery from the back office is enabled. If recovery is enabled, the metered-usage data and license-distribution state for concurrent features is recovered from the back office when the license server initially starts up with a new or reset trusted storage. (Default is false.)

No License server recovery from backoffice enabled

Policies for License Server Failover
Table A-1 • License Server Policy Settings (cont.)

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>fne.syncTo.enabled</td>
<td>(Defined on back-up license server only; required) The property that determines whether to enable “license server to license server” synchronization in a failover configuration. (Default is true.)</td>
<td>Yes</td>
<td>Synchronization to FNE enabled</td>
</tr>
<tr>
<td>fne.syncTo.mainUri</td>
<td>(Defined on back-up license server only; required) The URI of the main license server in a failover configuration. Use the following format: <a href="http://server:port/fne/bin/capability">http://server:port/fne/bin/capability</a> where server:port is the main license server’s name and port number, as in: <a href="http://11.11.1.11:7070/fne/bin/capability">http://11.11.1.11:7070/fne/bin/capability</a></td>
<td>Yes</td>
<td>Main FNE server URI</td>
</tr>
<tr>
<td>fne.syncTo.repeats</td>
<td>(Defined on back-up license server only) The amount of time between synchronization sessions from the main server to the back-up server in a failover configuration. (The back-up server initiates the sessions.) The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. (Default is 300s; minimum is 5m.)</td>
<td>No</td>
<td>Synchronization to FNE interval</td>
</tr>
<tr>
<td>fne.syncTo.pagesize</td>
<td>(Defined on back-up license server only) The maximum number of client records to include in a synchronization message to the back-up server. A smaller page size limits the memory overhead at the expense of having multiple synchronization transactions. (Default is 100.)</td>
<td>Yes</td>
<td>Synchronization to FNE pagesize</td>
</tr>
<tr>
<td>fne.syncTo.retryCount</td>
<td>(Defined on back-up license server only) The number of synchronization attempts from the main server allowed when an initial attempt fails. (Default is 1.)</td>
<td>Yes</td>
<td>Synchronization to FNE retry count</td>
</tr>
<tr>
<td>fne.syncTo.retryRepeats</td>
<td>(Defined on back-up license server only) The amount of time between synchronization attempts when an initial attempt fails. The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. The default is 60s.</td>
<td>Yes</td>
<td>Synchronization to FNE retry repeat interval</td>
</tr>
</tbody>
</table>

Security Policies
### Security Policies

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>security.enabled</td>
<td>The option that enables (true) or disables (false) administrative security on the license server.</td>
<td>Yes</td>
<td>REST Security enabled</td>
</tr>
<tr>
<td></td>
<td>When administrative security is enabled, operations used to administer the license server are “secured” (that is, credentials are required to perform them). See Managing Administrative Security on a Local License Server or CLS Instance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When this option is true, the remaining policies in this Security Policies section are in effect.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Default is false.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>security.token.duration</td>
<td>The duration of the JSON web token (generated when a user successfully authenticates credentials on the license server). When the token expires, credentials must be re-entered to re-authorize.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>The value can be specified with an optional unit-suffix letter—s, m, h, d, or w—indicating seconds, minutes, hours, days, or weeks. If no suffix is used, the server assumes the value is in seconds. The default is 1d.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This policy is not editable in the FlexNet License Server Manager.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>security.http.auth.enabled</td>
<td>The option that enforces the use of HTTPS to perform secured administrative (and possibly licensing) operations on the license server.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>• When false, the policy enforces the use of HTTPS to perform secured operations. (An error is generated for any attempt to perform a secured operation using HTTP.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When true, the policy allows either HTTP or HTTPS to perform secured operations. (This is the default.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This policy is not editable in the FlexNet License Server Manager.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>security.ip.whitelist</td>
<td>The list of IP addresses for those components (devices) that you determine should have access to the license server without having to provide credentials. For example, you might want a machine in your IT department to have such access to the license server for fixing issues or performing maintenance.</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>List only IP4 or IP6 addresses; and separate each address with a comma, as this example value shows:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>111.222.2.2, 111.333.3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This policy is not editable in the FlexNet License Server Manager.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The option that determines whether or not users need credentials for “read” access to the license server’s endpoints:

- When the value is `true`, all user accounts are automatically given “read” rights (ROLE_READ) and do not need to provide credentials for “read” access.

- When the value is `false`, a given user account must be explicitly assigned ROLE_READ in order to perform “read” operations. (The exception occurs when no role is assigned to an account, in which case ROLE_READ is assigned as the only role by default.) Credentials are then required to perform any “read” operation. If an account is not authorized for ROLE_READ, no “read” access is given. This setting provides additional protection against unauthorized queries on the license server.

This policy is not editable in the FlexNet License Server Manager.

(Default is `false`.)

### Logging Policies

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>logging.directory</td>
<td>The directory to which the license server writes the log for the license server. The default is <code>${base.dir}/logs</code>, where <code>${base.dir}</code> points to the <code>flexnetls/producer_name</code> folder in the service’s or user’s home directory.</td>
<td>No</td>
<td>Logging directory</td>
</tr>
</tbody>
</table>
The lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG. For example, if FATAL is set, only messages about fatal events are recorded. However, if WARN is set, fatal-event, error, and warning messages are recorded.

(Default is INFO.)

**Logging categories**

**FATAL**—Errors that prevent the server from starting up

**ERROR**—Serious errors

**WARN**—Warnings

**INFO**—Informational messages

**LICENSING**—Server responses such as, for example, capability responses and JSON replies

**POLICY**—Additional information for checkout filters (these are selective license filters customizable by the publisher)

**DEBUG**—Additional debug-level information. The license server should not use a logging level of DEBUG for a long period, because it can have a negative impact on license server performance. It is not recommended to use DEBUG on production license servers.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Editable?</th>
<th>In FlexNet License Server Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>logging.threshold</td>
<td>The lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG. For example, if FATAL is set, only messages about fatal events are recorded. However, if WARN is set, fatal-event, error, and warning messages are recorded.</td>
<td>Yes</td>
<td>Logging threshold</td>
</tr>
<tr>
<td>graylog.host</td>
<td>The host name of a Graylog server, if any, to which logging messages are sent.</td>
<td>Yes</td>
<td>Graylog host</td>
</tr>
<tr>
<td>graylog.threshold</td>
<td>The lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG. For example, if FATAL is set, only messages about fatal events are recorded. However, if WARN is set, fatal-event, error, and warning messages are recorded. (Default is WARN.)</td>
<td>Yes</td>
<td>Graylog threshold</td>
</tr>
</tbody>
</table>
SysV Alternative for Installation on Linux

As an alternative to installing the FlexNet Embedded local license server as a Linux service using systemd (described in Getting Started), you can use the SysV init runlevel system to perform the installation. This type of installation is an option only if your operating system supports SysV.

This following sections describe how use SysV to install and run the license server as a Linux service:

- Files Required for License Server Installation Using SysV
- Configure, Install, and Start the License Server
- Components Installed
- Edit Local Settings Post-Installation
- Uninstall the License Server Service

Files Required for License Server Installation Using SysV

The following files, distributed with the license server, support the installation and running of the license server service under SysV:

- producer-settings.xml
- flexnetls
- configure

Residing in the same directory as flexnetls.jar, these files work together to generate the components required to install and start the license server service in a SysV configuration.

For a list of all files (aside from the SysV installer files) that can be distributed with the license server, see Components Installed.
Appendix B  SysV Alternative for Installation on Linux
Configure, Install, and Start the License Server

Configure, Install, and Start the License Server

The instructions in this section describe how to configure, install, and run the local license server as a Linux service using a SysV init runlevel system.

Task To install and start the license server as a service on Linux

1. From the installation directory, execute `sudo ./configure` to generate the `/etc/default/flexnetls-producer_name` file, which defines license server settings for your local environment. Do either of the following:

   • To generate the file with the default settings, simply issue the `sudo ./configure` command.

   • To generate the file with settings specific to your environment, run the command with one or more of the options described in the following table to edit settings as needed. For example, to update the port and user name for the license server service, you might enter the following, where 7071 and user1 are the new values to which you are updating the port and user name:

     `sudo ./configure --port 7071 --user user1`

     Note • Any setting value that uses a space must be enclosed in double quotations (for example “user 1”).

Table B-1 • Local Settings for the License Server on Linux

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--program-dir dir</td>
<td>The installation location of the license server service (default is /opt/flexnetls/producer_name).</td>
</tr>
<tr>
<td>--data-dir dir</td>
<td>The location of trusted storage (default is /var/opt/flexnetls/producer_name).</td>
</tr>
<tr>
<td>--port port</td>
<td>The listening port used by the license server (default is 7070). If the machine on which the license server is running uses multiple network interfaces, you can use the port option to specify the interface that you want the license server to use. Simply include the IP address for the interface in square brackets, as shown in this example: --port [127.0.0.1].1443</td>
</tr>
<tr>
<td>--user user_name</td>
<td>The user name under which the license server service runs (default is flexnetls).</td>
</tr>
<tr>
<td>--group group_name</td>
<td>The group name under which the license server service runs (default is flexnetls).</td>
</tr>
</tbody>
</table>
2. Execute `sudo ./flexnetls install` to install the license server as a service. If the server service is installed properly, you receive the message `Service flexnetls-producer_name installed`, where `flexnetls-producer_name` is the name of the service.

3. Execute `sudo service flexnetls-producer_name start` to start the service. You should receive a message stating that the service has started.

4. To confirm that the service is running, execute `sudo service flexnetls-producer_name status`. You should receive the message `flexnetls is running`. (To stop the service, execute `sudo service flexnetls-producer_name stop`.)

5. To view the license server log, navigate to the server's logging directory (by default, `/var/opt/flexnetls/producer_name/logs`), and review the contents of the appropriate `.log` file.

---

**Note** • Once you install the license server as service, you cannot use any of the “flexnetls” non-service command-line options, such as “console” or “install”.

## Components Installed

These components are included in the license server installation on Linux.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flexnetls.jar</td>
<td>The Java executable file for the FlexNet Embedded local license server.</td>
</tr>
<tr>
<td>producer-settings.xml</td>
<td>The configuration file, generated by the producer, that defines policies for license server operations. Some of these settings can be overridden using the FlexNet License Server Administrator command-line tool or the FlexNet License Server Manager.</td>
</tr>
<tr>
<td>flexnetls</td>
<td>The shell script used to start the FlexNet Embedded local license server when the service is installed under a SysV init runlevel system.</td>
</tr>
</tbody>
</table>
### Appendix B  SysV Alternative for Installation on Linux

#### Components Installed

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>configure</strong></td>
<td>A utility used to generate and edit a local settings file that contains information needed to run the license server in your local environment. This file is used only when the service is installed under a SysV init runlevel system.</td>
</tr>
</tbody>
</table>
| Truststore file provided by producer | Optional file containing root and intermediate certificates used to validate an HTTPS connection to FlexNet Operations.  
  If the producer is using the Revenera-hosted version of FlexNet Operations, no truststore file is required.  
  If the producer is using the “on-premises” version of FlexNet Operations and the server certificate has been signed by an unknown (private) certificate authority, then the truststore file is required. |
| regid.2009-06.com(...).swidtag | The software ID tag file for use with software asset management tools. |
| • backofficeofflinesynctool | (Optional) The set of tools allowing the license server to perform offline synchronization to the back office. The software producer determines whether to provide these tools.  
  The tools also require the flxPublicTools.jar, EccpressoAll.jar, and commons-codec-1.9.jar files. |
| • serverofflinesynctool | (Optional) The set of tools allowing the license server to perform offline synchronization to the back office. The software producer determines whether to provide these tools.  
  The tools also require the flxPublicTools.jar, EccpressoAll.jar, and commons-codec-1.9.jar files. |
| • flexnetlsadmin.sh | (Optional) Files used to run the FlexNet License Server Administrator command-line tool. The software producer determines whether to provide this tool. |
| • flexnetlsadmin.jar | (Optional) Files used to run the FlexNet License Server Administrator command-line tool. The software producer determines whether to provide this tool. |
| flsm.war | (Optional) The file required to run the FlexNet License Server Manager. The software producer determines whether to provide this tool. |
| fne.war | (Optional) The file needed to migrate your custom legacy server administration tool to the FlexNet Embedded local license server. The software producer determines whether to provide this tool. |
| • bin directory | (Optional) Special set of files needed to run FNLicensingService, a service used to capture the VM-UUID on those Linux virtual platforms that do provide an appropriate mechanism for retrieving this ID. If you intend for your local license server to run on such a virtual Linux platform and require it to use the VM-UUID as its hostid, consult the software producer for further instructions. |
| • install-vm-only.sh | (Optional) Special set of files needed to run FNLicensingService, a service used to capture the VM-UUID on those Linux virtual platforms that do provide an appropriate mechanism for retrieving this ID. If you intend for your local license server to run on such a virtual Linux platform and require it to use the VM-UUID as its hostid, consult the software producer for further instructions. |
| • Readme.text | (Optional) Special set of files needed to run FNLicensingService, a service used to capture the VM-UUID on those Linux virtual platforms that do provide an appropriate mechanism for retrieving this ID. If you intend for your local license server to run on such a virtual Linux platform and require it to use the VM-UUID as its hostid, consult the software producer for further instructions. |
Edit Local Settings Post-Installation

As part of the installation process, the license server package generates a local settings file that configures the license server for your local environment, as described in the previous section, Configure, Install, and Start the License Server. The installation process provides a facility that lets you change default settings before settings file is generated.

Once the license server is installed, you can edit the local settings at any time, using the following instructions.

**Task**

To modify local settings after installing the license server as a service on Linux

1. Using `sudo` privileges, open the `/etc/default/flexnetls-producer_name` file in a text editor, and edit the settings as needed. (For example, you can uncomment settings or change existing values.) See the previous section Configure, Install, and Start the License Server for a description of the base settings.

Note that the following additional settings, which you can also edit, are created in the file at the time it is generated with the `configure` tool:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACTIVE_HOSTID=value/type</strong></td>
<td>The hostid to use for the license server. The hostid can only be set using a configuration file if no hostid has yet been specified for the license server. Once a hostid has been set, it can only be changed using the FlexNet License Server Administrator or the FlexNet License Server Manager. The syntax is <code>value/type</code> (for example, 7200014f5df0/Ethernet). If no value is specified for this setting and no active hostid has yet been set for the license server, the license server uses, by default, the first available Ethernet address on the machine. If using a dongle ID or a hostid other than the first available Ethernet address, specify it here. For more information about hostids, see Understanding Hostids.</td>
</tr>
</tbody>
</table>

**Important** • It is not recommended to change the hostid of a license server that has licenses mapped in the back office (FlexNet Operations). If the hostid is changed to a value that is different to that specified for the license server in the back office, any existing licenses mapped to the license server that is locked to the old hostid in the back office will be orphaned.

To prevent this from happening, it is best practice to return the license server in the back office. During the return operation, the producer can transfer the licenses to a different device; this can be the same machine with the desired hostid. After the transfer, wait for the license server to synchronize with the back office (server synchronization occurs based on synchronization policies or on-demand).
<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS_SERVER_FILE=path</td>
<td>The path to the HTTPS “server” configuration file used to support <em>incoming</em> HTTPS from client devices. For details on how to create this configuration file and set up the license server for HTTPS, see <em>Incoming HTTPS</em> in More About License Server Functionality.</td>
</tr>
<tr>
<td>HTTPS_CLIENT_FILE=path</td>
<td>The path to the HTTPS “client” configuration file used to support <em>outgoing</em> HTTPS communication to FlexNet Operations.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> • The “client” configuration file is being deprecated and will be removed in a future release. Instead of the “client” configuration file, specify parameters to access the truststore file in the <code>https-out</code> setting in <code>local-configuration.yaml</code>. For more information about HTTPS setup on the license server, see <em>Outgoing HTTPS</em>.</td>
</tr>
<tr>
<td>SERVER_ALIAS=alias</td>
<td>A user-defined name (sometimes called <em>host name</em>) for the license server. This name is added to server’s capability requests to the back office, where it is then saved and used to identify the license server in the FlexNet Operations Producer and End User portals.</td>
</tr>
<tr>
<td></td>
<td>One important use for this setting is that the alias can be included in the initial capability request sent at server registration, providing a helpful name by which users can identify the new server in the portals. If no alias is sent at registration, the server is identified by its hostid.</td>
</tr>
<tr>
<td>EXTENDED_SUFFIX=suffix</td>
<td>The suffix used for the Extended Host ID feature. This value is not set by default. Contact the software producer for details.</td>
</tr>
<tr>
<td>EXTRA_SYSPROPERTIES=</td>
<td>One or more system properties (each in <code>-D key=value</code> format) used by the license server to support different types of functionality. If entering multiple properties, enclose the entire set of properties in double quotations.</td>
</tr>
<tr>
<td>-property1=value1</td>
<td>For example, if you plan to have the license server communicate with the back office through an HTTP proxy, use this setting to identify the proxy parameters needed to configure the server. (For details, see Proxy Support for Communication with the Back Office in the More About License Server Functionality chapter.) The following shows example proxy parameters listed as `-D system properties for this setting:</td>
</tr>
<tr>
<td>-property2=value2...</td>
<td>EXTRA_SYSPROPERTIES=&quot;-Dhttp.proxyHost=10.90.3.133 -Dhttp.proxyPort=3128 -Dhttp.proxyUser=user1a -Dhttp.proxyPassword=user1apwd35&quot;</td>
</tr>
<tr>
<td></td>
<td>The entire set of parameters must be enclosed in double quotations, even if you specify only a single parameter, such as EXTRA_SYSPROPERTIES=&quot;-Dhttp.proxyHost=10.90.3.133&quot;.</td>
</tr>
</tbody>
</table>
Uninstall the License Server Service

Use these steps to uninstall the local license server service installed with SysV.

<table>
<thead>
<tr>
<th>Task</th>
<th>To uninstall the license server service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Execute the following command: <code>sudo service flexnetls-producer_name stop</code>.</td>
</tr>
<tr>
<td>2.</td>
<td>Delete the <code>/opt/flexnetls/producer_name</code> folder.</td>
</tr>
<tr>
<td>3.</td>
<td>Delete the <code>/etc/init.d/flexnetls-producer_name</code> and <code>/etc/default/flexnetls-producer_name</code> files.</td>
</tr>
<tr>
<td>4.</td>
<td>Optionally, delete these files (listed here with their default locations):</td>
</tr>
<tr>
<td></td>
<td>• The trusted storage files in <code>/var/opt/flexnetls/producer_name</code></td>
</tr>
<tr>
<td></td>
<td>• The log files in <code>/var/opt/flexnetls/producer_name/logs</code></td>
</tr>
</tbody>
</table>

Trusted storage and log locations are defined by the license server policies `server.trustedStorageDir` and `logging.directory`, respectively, the defaults for which are based on `${bases.dir}`. Depending on the values set for these policies on your server, your trusted storage and log files might be in locations different from those mentioned in this step. See Reference: License Server Policy Settings.
Appendix B  SysV Alternative for Installation on Linux

Uninstall the License Server Service


Model Definition Grammar for Partitions

The model definition defines partitions and conditions for rules that allow or deny access to features. The following sections provide information and examples for creating model definitions:

- Model Definition Grammar and Syntax—EBNF
- Partition Use Case Examples and Their Model Definitions
- Model Definition Examples for Reservations Converted to Partitions

Model Definition Grammar and Syntax—EBNF

The following section shows all possible grammar structures of the language used to write a model definition. It is written using the EBNF (Extended Backus–Naur form), which is a formal notation that can be used to describe other languages.

Each definition can have only one model. Reserved model names are reservations and default.

```
model = 'model', model_name, '{', partitions, rule*, '}';
model_name = quoted_string;
partitions = 'partitions', '{', partition*, '}';
partition = 'partition', partition_name, '{', feature_spec*, '}';
partition_name = quoted_string;
```

feature_spec = feature_name, feature_version, amount;
feature_name = quoted_string;
feature_version = major_version, [ '.', minor_version ];
major_version = digit+;
minor_version = digit+;
amount = integer_amount | percentage_amount | 'remainder';
integer_amount = digit+;
percentage_amount = digit+, '%';
rule = 'on', compound_matcher, '{', rule_body, '}';
compound_matcher = negatable_matcher, compound_rhs*;
compound_rhs = compound_and_rhs | compound_or_rhs;
compound_and_rhs AND, negatable_matcher;
compound_or_rhs OR, negatable_matcher;
negatable_matcher = negated_matcher | simple_matcher;
negated_matcher = NOT, simple_matcher;
simple_matcher = hostid_matcher | hostname_matcher | hosttype_matcher | vendor_dictionary_matcher | any_matcher;
hostid_matcher = 'hostid', '(', parameter_list, ')';
hostname_matcher = 'hostname', '(', parameter_list, ')';
hosttype_matcher = 'hosttype', '(' parameter_list, ')';
vendor_dictionary_matcher = 'dictionary', '(', keyword_parameter_list, ')';
any_matcher = 'any', '(', ')';
parameter_list = parameter, (',', parameter)*;
parameter = quoted_string;
keyword_parameter_list = keyword_parameter, (',', keyword_parameter)*;
keyword_parameter = keyword_key, ':', keyword_value;
keyword_key = quoted_string;
keyword_value = '*' | quoted_string;
rule_body = [use_statement] [server_specified] [action];
use_statement = 'use', partition_name_list;
Note • When reading the EBNF, note the following:

- Optional elements are represented inside square brackets ([ ]).
- An asterisk (*) denotes that the preceding symbol or parenthesized expression may occur one or more times.
- A plus sign (+) denotes that the preceding symbol or parenthesized expression may occur zero or more times.
- Feature versions have the format NNNN.NN.
- Strings cannot be multi-line.
- Multiple feature_spec elements can optionally be separated by commas.
- If a backslash character is used inside a string, it must be escaped using a second single backslash ( \ ).
- If a double-quote character is used inside a string, it must be escaped using a backslash ( \" ).

Partition Use Case Examples and Their Model Definitions

The following examples illustrate possible use cases for sharing feature counts between partitions:

- Use Case: Simple Allow List
- Use Case: Simple Block List
- Use Case: Sharing Counts Between Business Units
- Use Case: Assigning Extra Counts To Business Unit
- Use Case: Exclusive Use of Features Counts for Business Unit
- Use Case: Exclusive Use of Feature Counts for Business Unit With Exception of Specific Clients
- Use Case: Assigning Features Based on Combined hosttype and hostname Properties
Use Case: Device-specific Handling—Sharing Feature Counts Based On Hosttype
Use Case: Partition Receiving Entire Remaining Feature Count
Use Case: Letting Server Specify Counts
Use Case: Accumulating Counts from Multiple Partitions (“Continue” Action)

Background Information for Use Cases
The following is information to keep in mind when reading the information in this section.

Vendor Dictionary Data
Many of the following use cases use vendor dictionary data to define conditions for license allocation.

The vendor dictionary enables the software producer to send custom data in a capability request (in addition to the FlexNet Embedded–specific data) to the license server and vice-versa. Basically, the vendor dictionary provides a means to send information back and forth between the client and server for any producer-defined purposes, as needed; FlexNet Embedded does not interpret this data.

Vendor dictionary data is stored as key–value pairs. The key name is always a string, while a value can be a string or a 32-bit integer value. Keys are unique in a dictionary and hence allow direct access to the value associated with them.

A common theme across the use cases in this appendix is that feature counts are shared between business units called Sales and Engineering. These business units would be identified by “business-unit” : [“engineering”, “sales”] entries in the vendor dictionary.

For information about vendor dictionary data that might be available, contact your software producer.

Default Behavior If No Rule Conditions Are Met
When a feature request does not meet any of the conditions in the model definition, an on any() condition is expected that either denies or allows access to a partition. If no such on any() condition is provided, the default is that the feature will be served from the default partition (if features are available). This behavior is equivalent to the following code:

```java
on any() {
    use "default"
    accept
}
```

The examples in this chapter generally do not explicitly show this final on any() condition.

Use Case: Simple Allow List
Requirement: Allow access to specific hostids

In this scenario, only declared hostids are allowed access. Note that no partitions are needed in this scenario.
Instead of specifying all hostids in one condition within a rule, you can also include multiple rules with the "on hostid" condition in the model definition.

**Model Definition Example**

```plaintext
model "exampleModel" {
    on hostid("F01898AD8DD3/ETHERNET", "5E00A4F17201/ETHERNET") {
        use "default"
        accept
    }
    on any() {
        deny
    }
}
```

**Note** • The hostid is specified as a value/type pair (for example, 720014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

**Use Case: Simple Block List**

**Requirement**: Deny access to specific hostids

In this scenario, all users except a declared few are allowed access.

**Model Definition Example**

```plaintext
model "exampleModel" {
    on hostid("user1@example.com/USER", "user2@example.com/USER") {
        deny
    }
    on any() {
        use "default"
        accept
    }
}
```

**Note** • The hostid is specified as a value/type pair (for example, 720014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

**Use Case: Sharing Counts Between Business Units**

**Requirement**: Share feature counts between two business units.

This scenario features two business units called Sales and Engineering, which have been defined by entries in the vendor dictionary.
The model definition defines two partitions, one for the Engineering business unit and another for the Sales business unit. The definition shares the total feature count of 10 for feature f1 equally between both business units, meaning each business unit gets 5.

*Model Definition Example*

```plaintext
model "exampleModel" {
    partitions {
        partition "engineering" {
            "f1" 1.0 5
        }
        partition "sales" {
            "f1" 1.0 5
        }
    }
    on dictionary("business-unit" : "engineering") {
        use "engineering"
        accept
    }
    on dictionary("business-unit" : "sales") {
        use "sales"
        accept
    }
}
```

**Use Case: Assigning Extra Counts To Business Unit**

*Requirement:* Make specified number of features accessible only to one particular business unit.

This scenario features the same business units as before—Sales and Engineering.

For this demonstration, let’s assume that the total feature count for f1 is 10. Of the total count, 3 licenses are allocated exclusively to the engineering partition. In addition, engineering is allowed access to the remaining 7 licenses (on a first-come-first-served basis) from the default partition.

The Sales business unit (which has no pre-allocated counts in this scenario) only has access to the default partition.

*Model Definition Example*

```plaintext
model "exampleModel" {
    partitions {
        partition "engineering" {
            "f1" 1.0 3
        }
    }
    on dictionary("business-unit" : "engineering") {
        use "engineering", "default"
        accept
    }
}
```
Use Case: Exclusive Use of Features Counts for Business Unit

Requirement: Grant exclusive use of features to a specified business unit

This scenario uses the same business units and partitions as the previous example: the default partition, and a partition called engineering for the Engineering business unit. Other business units also exist (for example, the Sales business unit).

Let's assume that the total feature count for a feature called “cad” is 10. The entire count (expressed here as 100%) is allocated to the engineering partition, for exclusive use by the Engineering business unit. This effectively denies access to feature “cad” for any capability request that comes from a different business unit (that is, any request that does not include the vendor dictionary entry “business-unit” : “engineering”).

Model Definition Example

model "exampleModel" {
  partitions {
    partition "engineering" {
      "cad" 1.0 100% // entire feature count
    }
  }
  on dictionary("business-unit" : "engineering") {
    use "engineering"
    accept
  }
}

Use Case: Exclusive Use of Feature Counts for Business Unit With Exception of Specific Clients

Requirement: Grant exclusive use of features to a specified business unit, with the exception of specified clients.

This scenario uses the same framework as the previous use case, Use Case: Exclusive Use of Features Counts for Business Unit, but adds a filter that blocks specified clients.

Capability requests are granted for all clients from the business unit Engineering, except for requests coming from clients with the hostid "5e00a4f17204/ETHERNET" or "5e00a4f17205/ETHERNET".

Model Definition Example

model "exampleModel" {
  partitions {
    partition "engineering" {
      "cad" 1.0 100% // entire feature count
    }
  }
  on dictionary("business-unit" : "engineering") and not hostid("5e00a4f17204/ETHERNET", "5e00a4f17205/ETHERNET") {
    use "engineering"
    accept
  }
}
Appendix C  Model Definition Grammar for Partitions
Partition Use Case Examples and Their Model Definitions

Note • The hostid is specified as a value/type pair (for example, 7200014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

For information about the operators AND and NOT used in this example, see AND, OR, and NOT Operators.

Use Case: Exclusive Use of Feature Counts for Business Unit and Specified Clients from other Business Units

Requirement: Grant exclusive use of features to a specified business unit and to a number of specified clients.

This scenario uses the same framework as the use case described in Use Case: Exclusive Use of Feature Counts for Business Unit With Exception of Specific Clients. However, instead of filtering out specified clients, it also grants access to a number of select clients that are not part of the Engineering business unit.

Capability requests are granted for all clients from the business unit Engineering, and for clients with the hostid "5e00a4f17204/ETHERNET" or "5e00a4f17205/ETHERNET" (which may well be part of other business units).

Model Definition Example

```plaintext
model "exampleModel" {
  partitions {
    partition "engineering" {
      "cad" 1.0 100% // entire feature count
    }
  }

  on dictionary("business-unit" : "engineering") or hostid("5e00a4f17204/ETHERNET", "5e00a4f17205/ETHERNET") {
    use "engineering"
    accept
  }
}
```

Note • The hostid is specified as a value/type pair (for example, 7200014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

For information about the operators AND and NOT used in this example, see AND, OR, and NOT Operators.

Use Case: Assigning Features Based on Combined hosttype and hostname Properties

Requirement: Allow only clients that match specified hosttype and hostname properties access to a partition.

This scenario uses the same Engineering business unit and partition that you already know from previous examples.
In this example, a producer wants to allow client devices that match both a particular host type and host name access to the engineering partition. This can be achieved by combining the hosttype and hostname conditions using the AND operator. Requests from devices that match only the host type or host name, or none of these, are served from the default partition.

**Model Definition Example**

```
model "exampleModel" {
    partitions {
        partition "engineering" {
            "f1" 1.0 100
        }
    }

    on hosttype("tv") and hostname("device-1"){
        use "engineering"
        accept
    }
}
```

**Use Case: Device-specific Handling—Sharing Feature Counts Based On Hosttype**

**Requirement:** Use the hosttype property to route the capability request to the appropriate partition.

In this scenario there are three different models of routers, which are distinguished through the hosttype property in the capability request. Counts of the “router” feature are split between three different partitions—called small, medium, and large—and the “on hosttype” condition is used to route the request to the appropriate partition.

The hosttype is a value that is optionally set by the software producer on a client device. (This hosttype is also sometimes referred to as a device type.) A hosttype is a human-readable “alias”—in contrast to the hostid—which can optionally be included in a capability request. Contact your software producer to find out about hosttypes that are available to you.

**Model Definition Example**

```
model "exampleModel" {
    partitions {
        partition "small" {
            "router" 1.0 5
        }
        partition "medium" {
            "router" 1.0 10
        }
        partition "large" {
            "router" 1.0 20
        }
    }

    on hosttype("small") {
        use "small"
        accept
    }
}
```
Use Case: Partition Receiving Entire Remaining Feature Count

**Requirement:** Distribute all counts of a feature between three partitions as equally as possible, with none in the default partition.

If you declare three partitions, two with a count specifier of 33% and one with 34%, integer rounding might cause some counts to slip into the default partition. You can avoid this by replacing the last count specifier with the `remainder` keyword, which will match all available remaining counts.

Note that any partition lower than the one specifying the `remainder` keyword in the model definition receives zero counts for that feature.

**Model Definition Example**

```plaintext
model "exampleModel" {
  partitions {
    partition "p1" { // Total feature count of f1: 10
      "f1" 1.0 33% // Feature count in p1: 3
    }
    partition "p2" {
      "f2" 1.0 33% // Feature count in p2: 3
    }
    partition "p3" {
      "f3" 1.0 remainder // Feature count in p3: 4
    }
  }
}
```

Use Case: Letting Server Specify Counts

**Requirement:** Let the server specify the desired counts, instead of the client.

The following model definition demonstrates how to let the server specify the features that should be assigned to a client device. The model uses the `without requested features` directive in combination with a feature specification. Note that the server cannot override a capability request containing desired features coming from the client.

The following rule assigns a count of feature f1 to each capability request with the hostname "myhost".

**Model Definition Example**

```plaintext
model "exampleModel" {
  partitions {
    partition "p1" {
      "f1" 1.0
    }
    partition "p2" {
      "f2" 1.0
    }
    partition "p3" {
      "f3" 1.0
    }
  }
}
```
You can also combine the `without requested features` directive with `all from "partitionName"`. This variation will result in the first capability request receiving all counts from the specified partition. For a scenario where `all from "partitionName"` might be used, see Scenario: Server-specified Counts.

**Note •** Server-specified desired feature counts are automatically marked with the "partial" attribute, indicating that partial checkout is allowed if the feature’s count falls short of the desired count.

### Use Case: Accumulating Counts from Multiple Partitions ("Continue" Action)

**Requirement:** Allow a capability request to collect feature counts from more than one partition

This use case defines two partitions for feature counts. If a capability request comes from Europe (containing the key-value pair "region"/"Europe", defined by the producer using vendor dictionary data), it gets access to partition p1. If the request comes from a device labeled “10A” (containing the key-value pair "model" : "10A", defined by the producer using vendor dictionary data), it gets access to partition p2. If both conditions are true, the request gets access to both p1 and p2.

**Model Definition Example**

```plaintext
data model "exampleModel" {  partitions {    partition "p1" {       "f1" 1.0 5    }    partition "p2" {       "f1" 1.0 5    }  }  on dictionary("region" : "Europe") {    use "p1"    continue  }  on dictionary("model" : "10A") {    use "p2"    continue  }
```

```
Model Definition Examples for Reservations Converted to Partitions

The following examples show model definitions that might result from converting reservations to partitions. Reservations can be converted to partitions either manually or automatically. Manual conversion involves creating a model definition that reproduces the allocation achieved previously through reservations. If you do not convert reservations manually, they will automatically be converted into partitions and rules. For more information, see Partitions vs. Reservations.

Scenario: Counts Reserved By Hostid

The original reservation model consists of per-hostid reserved counts, together with the ability to get counts from the default shared pool. Reservations might look like this when expressed in a model definition:

```plaintext
model "model-1" {
    partitions {
        partition "reservation-1" {
            "f" 1 0 1
        }
    }
    on hostid("F01898AD8DD3/ETHERNET") {
        use "reservation-1", "default"
        accept
    }
    on any() {
        use "default"
        accept
    }
}
```

Note that creating rules based on hostids is possible, but generally not recommended. Maintaining large lists of rules with the hostid condition is tedious and prone to error. Instead, consider basing the rules on hostname or hosttype properties or vendor dictionary key/value pairs.

Note • The hostid is specified as a value/type pair (for example, 7200014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

Scenario: Server-specified Counts

Sometimes, a capability request does not contain any desired features. In this case, the server can be configured to specify the features that should be assigned to a client device. If a reservation exists for the client that sent the capability request, the reserved counts are acquired. The following model reservation example demonstrates how this behavior is replicated with partitions. The without requested features directive replicates the server-
specified count, and all from "partitionName" allocates all feature counts from the specified partition to the first client sending a capability request that fulfills the rule’s conditions. The all from "partitionName" instruction essentially acts as a shortcut to the feature counts specified for the relevant partition in the model definition. If the specified partition does not have sufficient counts to satisfy the request, counts are allocated from subsequent partitions listed in the rules (provided that the client has access to those partitions).

Let’s assume that the original reservation model consists of per-hostid reserved counts, together with the ability to get counts from the default shared pool. The resulting model definition and rule might look like this:

```plaintext
model "reservations" {
  partitions {
    partition "reservation-1" {
      "f1" 1.0 1
    }
  }
  on hostid("F01898AD8DD3/ETHERNET") {
    use "reservation-1", "default"
    without requested features {
      all from "reservation-1"
    }
    accept
  }
}
```

The without requested features directive can also be combined with a feature specification to produce more fine-grained rules. For a use case example, see Use Case: Letting Server Specify Counts.

As noted in the previous scenario, creating rules based on hostids is generally not recommended. Instead, consider basing the rules on hostname or hosttype properties or vendor dictionary key/value pairs.

---

**Note** - Server-specified desired feature counts are automatically marked with the "partial" attribute, indicating that partial checkout is allowed if the feature’s count falls short of the desired count.

**Note** - The hostid is specified as a value/type pair (for example, 7200014f5df0/ETHERNET). If a hostid condition does not specify the hostid type, it is assumed that the hostid is of type string.

### Scenario: Reservation JSON Converted to Model

The following scenario illustrates the changes that occur when the reservations JSON file is automatically converted to an equivalent model definition.

**Initial Reservation JSON**

This is the JSON code representing the original reservation.

```json
{
  "name": "reservation group #1",
  "reservations": [
    {
      "hostId": {
```
"value": "h1",
"type": "STRING"
},
"reservationEntries": [
{
"featureName": "f1",
"featureVersion": "1.0",
"featureCount": 2
}
]
]

Resulting Model Definition

The reservation JSON above would result in the following automatically-generated model definition:

```plaintext
model "reservations" {
  partitions {
    partition "p_116" {
      "f1" 1.0 2
    }
  }

  on hostid("h1") {
    use "p_116"
    without requested features { all from "p_116" }
    continue
  }

  on any() {
    use "default"
    accept
  }
}
```

A capability request from a client with hostid h1 is allowed access to counts in partition p_116. If no features are available in p_116, the server serves counts from the default partition.

The `continue` keyword accommodates the scenario where a capability request contains two hostids (for example, h1 and an additional hostid), and the rules for both hostids need to run. Hostid h1 will be granted access to p_116, and the `continue` keyword ensures that the next rule is evaluated (as opposed to `accept`, which would have stopped evaluation of any subsequent rules).

If a capability request does not contain any desired features, the `without requested features` directive returns all counts that were reserved for the client that sent the capability request (provided such a reservation exists). The directive `all from "p_116"` allocates all feature counts from partition p_116.
Logging Functionality on the Local License Server

This appendix describes the logging functionality available for the local license server, specifically:

- Logging Style
- Custom Log Configurations
- Integration of License Server Logging With External Systems

Logging Style

A logging style configuration parameter can be used to configure rollover, JSON formatting and timestamp behaviour. The different logging styles can be specified in the `local-configuration.yaml` file, using the `loggingStyle` property.

Set the property using the following format (the space after the colon is mandatory):

```yaml
loggingStyle: value
```

The property can take one of the following values:

**Logging style values:**

- **CONTINUOUS**—Rollover is externally handled, if at all. Timestamp values use the local time zone. This is the default if no logging style is specified.
- **CONTINUOUS_UTC**—Same as CONTINUOUS, but with UTC timestamp.
- **DAILY_ROLLOVER**—The log file is closed at midnight local time, compressed with gzip, and a new log file is started. Timestamp values use the local time zone.
- **DAILY_ROLLOVER_UTC**—Same as DAILY_ROLLOVER, but with UTC timestamp.
- **JSON_ROLLOVER**—Logs are formatted using JSON. The log file is closed at midnight local time, compressed with gzip (suitable for filebeat), and a new log file is started. Always uses ISO 8601 UTC timestamps.
- **JSON**—Logs are emitted as JSON to stdout only (suitable for Docker). Always uses ISO 8601 UTC timestamps.
Appendix D  Logging Functionality on the Local License Server

Custom Log Configurations

Note • The Docker-friendly JSON logging style is not supported when the local license server is run as a Windows or Linux service (because it only writes to stdout). If set, the style will be changed to DAILY_ROOLVER.

Overriding the Timestamp Format

If you have specific timestamp requirements (not local, not UTC), the timestamp format can be adapted. To change the timestamp, you must supply an override timestamp format as a Java system property. Example:

-Dlogging.timestamp="%d{MMM-dd-yyyy HH:mm:ss,SSS}"

This option is not available for JSON logging, because ISO 8601 UTC is standard.

Custom Log Configurations

If the preconfigured logging styles do not meet your requirements, you can customize the logging configuration. To do so, prepare an external configuration file and specify the path to that file in an environment variable ($LOGGING_CONFIGURATION).

To prepare the external configuration file, extract a log4j configuration from the local license server .jar file and modify it to suit your requirements. Use a command like the following to view the embedded configuration:

$ unzip -p flexnetls.jar BOOT-INF/classes/log4j-default.xml

Integration of License Server Logging With External Systems

The local license server can pass log entries to external log monitoring software. Some common configurations are documented here, but others are possible by adapting these instructions.

Common configurations described in this section are:

- Graylog
- Elastic Stack
- logz.io
- Log4J Socket Appender
- syslog logging (Linux)
- systemd logging (Linux)

Examples for implementing some of these configurations are provided in the Example Configurations section.

Graylog

The local license server supports Graylog’s GELF protocol (see graylog.org/features/gelf). GELF mode is enabled by supplying the following configuration values in producer-settings.xml:
• **graylog.host**—The host name of the Graylog server to which logging messages are sent.

The format for graylog.host is: [protocol:]host[:port]

The protocol and port components are optional, and default to "udp" and 12201, respectively. The supported protocol values are "udp" and "tcp".

• **graylog.threshold**—The lowest level of log-message granularity to record—FATAL, ERROR, WARN, INFO, LICENSING, POLICY, or DEBUG. For example, if FATAL is set, only messages about fatal events are recorded. However, if WARN is set, fatal-event, error, and warning messages are recorded. (Default is WARN.)

Instead of using producer-settings.xml, these values can also be supplied using the FlexNet License Server Administrator—see Managing License Server Policy Settings in the Using the FlexNet License Server Administrator Command-line Tool chapter.)

The section Graylog Logging shows an example of how you can implement this configuration.

### Elastic Stack

To use Elastic Stack logging, you must configure the server for JSON logging. To do so, in the local-configuration.yaml file, specify the following property and value:

```yaml
loggingStyle: JSON_ROLLOVER
```

The Elastic Stack consists of 3 components:

- **Logstash.** This component receives log entries, filters and transforms them and then passes the entries onwards.
- **Elasticsearch.** This component stores the log entries.
- **Kibana.** This component is the GUI where log entries can be searched and visualised.

The following methods are available to pass the log entries to Logstash:

- **Using Logstash’s native support of the GELF protocol**—Set up the graylog.host and graylog.threshold configuration properties to refer to Logstash.

- **Using Elastic Stack’s Filebeat to monitor the server’s logging directory**—The use of Filebeat to send log entries to the Elastic stack is generally preferred to the use of GELF, because the JSON output is richer and more flexible than the GELF log entry encapsulation, and it is more robust when network interruptions happen. For more information about Filebeat, go to elastic.co/beats/filebeat.

The sections Elastic Stack and Filebeat and Elastic Stack and GELF show examples of how you can implement this configuration.

### logz.io

logz.io is a managed and customised Elastic Stack logging offering. Refer to the methods in the previous section, Elastic Stack, for a short explanation of how to configure the server to work with it. For more information about logz.io, go to logz.io.
Log4J Socket Appender

The technique offered by Log4J Socket Appender comes in useful if you want to pipe log entries to another host using TCP/IP.

The section Log4J SocketAppender shows an example of how you can implement this configuration.

syslog logging (Linux)

syslog logging has been replaced by systemd logging in most Linux distros.

systemd logging (Linux)

Revenera do not currently recommend integration with systemd logging due to performance concerns. Note that syslog logging can forward to systemd logging in some Linux distros.

If you need to do this, use the -systemd-logging command line flag. Revenera strongly recommend not doing this with log threshold values of DEBUG, and measuring the impact before taking the configuration to production.

Example Configurations

This section contains examples for external logging configurations.

- Graylog Logging
- Elastic Stack and Filebeat
- Elastic Stack and GELF
- Log4J SocketAppender

Graylog Logging

The following example demonstrates Graylog logging.

Task | To enable Graylog logging
--- | ---
1. | Start Graylog. You could run Graylog using Docker or run it as a a virtual machine appliance under VirtualBox, depending on your preference.
2. | Set graylog.host to localhost to specify the Graylog server and set your desired log threshold (for example, INFO or LICENSING). If using the FlexNet License Server Administrator, you would use a command such as the following:

```
flexnetlsadmin -authorize admin -passwordConsoleInput -config -set graylog.host=localhost
graylog.threshold=LICENSING
```

Instead of using the FlexNet License Server Administrator you could also request a new producer-settings.xml file from your software producer.
3. Restart the license server. Log information will now be sent to Graylog on port 12201 using UDP (the default port and protocol).

4. Configure Graylog to receive the log information. For information, consult the Graylog documentation.

### Elastic Stack and Filebeat

This section explains how to log to a Docker installation of the Elastic Stack (Elasticsearch, Logstash and Kibana), using filebeat to send log contents to the stack. Log data is persisted in a Docker volume called "monitoring-data".

This configuration enables you to experiment with local license server logging to the Elastic Stack. However, it is not recommended for production use, due to insufficient high availability, backup and indexing functionality.

This section is split into the following steps:

1. Preparing the Directory Structure
2. Preparing the Local License Server
3. Creating the "docker-compose" File
4. Creating the Elasticsearch Files
5. Adding a Logstash Configuration
6. Building the Elastic Stack
7. Preparing to Use filebeat
8. Sending Log Entries to Logstash
Preparing the Directory Structure
This demo uses the following directory structure:

```
+-- demo/
    |   +-- docker-compose.yml
    |   +-- elasticsearch/
    |       |   +-- Dockerfile
    |       |   +-- elasticsearch.yml
    |   +-- filebeat/
    |   +-- server/
    |       |   +-- flexnetls.jar
    |       |   +-- producer-settings.xml
    |       |   +-- local-configuration.yaml
    |   +-- logstash.conf
```

Preparing the Local License Server
Follow the steps below to prepare the server to log to a Docker installation. This step assumes that Docker is installed.

**Task**

**To prepare the local license server for logging to a Docker installation:**

1. Copy the local license server files—flexnetls.jar, producer-settings.xml, and local-configuration.yaml—into the server directory.

2. Configure the server for logging in JSON format. Add the following entry to the top section of local-configuration.yaml:

   ```yaml
   loggingStyle: JSON_ROLLOVER
   ```

   This causes logs to be written in JSON format to the default location of $HOME/flexnetls/$PUBLISHER/logs.

3. Start the license server using the following command from the server directory:

   ```bash
   $ java -jar flexnetls.jar
   ```

4. Confirm that log files with a .json extension are being created.
Creating the “docker-compose” File

Use the sample below to create the docker-compose.yml file. Note that this demo uses the producer name “acme”.

docker-compose.yml

```
version: '2.2'
services:
elasticsearch:
    build:
        context: elasticsearch/
        container_name: elasticsearch
        volumes:
            - monitoring-data:/usr/share/elasticsearch/data
        ports:
            - "9200:9200"
            - "9300:9300"
        environment:
            ES_JAVA_OPTS: "-Xmx512m -Xms512m"
logstash:
    image: docker.elastic.co/logstash/logstash:7.9.1
    container_name: logstash
    ports:
        - "5000:5000"
        - "5044:5044"
        - "12201:12201/udp"
    expose:
        - "5044/tcp"
        - "12201/udp"
    logging:
        driver: "json-file"
    environment:
        LS_JAVA_OPTS: "-Xmx256m -Xms256m"
        volumes:
            - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
        depends_on:
            - elasticsearch
kibana:
    image: docker.elastic.co/kibana/kibana:7.9.1
    ports:
        - "5601:5601"
    depends_on:
        - elasticsearch
volumes:
    monitoring-data:
        driver: local
```
Creating the Elasticsearch Files

Create the following files in the elastic search directory.

```yaml
table
<table>
<thead>
<tr>
<th>elasticsearch/Dockerfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM elasticsearch:7.9.1</td>
</tr>
<tr>
<td>COPY ./elasticsearch.yml /usr/share/elasticsearch/config/elasticsearch.yml</td>
</tr>
<tr>
<td># FileRealm user account, useful for startup polling.</td>
</tr>
<tr>
<td>RUN bin/elasticsearch-users useradd -r superuser -p esuser admin</td>
</tr>
<tr>
<td>&amp;&amp; yum update -y</td>
</tr>
<tr>
<td>&amp;&amp; yum install -y jq</td>
</tr>
<tr>
<td>&amp;&amp; yum upgrade</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>elasticsearch/elasticsearch.yml</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.name: &quot;docker-cluster&quot;</td>
</tr>
<tr>
<td>network.host: 0.0.0.0</td>
</tr>
<tr>
<td># minimum_master_nodes need to be explicitly set when bound on a public IP</td>
</tr>
<tr>
<td># set to 1 to allow single node clusters</td>
</tr>
<tr>
<td># Details: <a href="https://github.com/elastic/elasticsearch/pull/17288">https://github.com/elastic/elasticsearch/pull/17288</a></td>
</tr>
<tr>
<td>discovery.zen.minimum_master_nodes: 1</td>
</tr>
<tr>
<td>## Use single node discovery in order to disable production mode and avoid bootstrap checks</td>
</tr>
<tr>
<td>discovery.type: single-node</td>
</tr>
<tr>
<td>node.data : true</td>
</tr>
<tr>
<td>discovery.seed_hosts : []</td>
</tr>
</tbody>
</table>
```
Adding a Logstash Configuration

Add a Logstash configuration such as the following to the demo directory:

```yaml
input {
  beats {
    port => 5044
  }
}

output {
  stdout { codec => rubydebug }

  if ( "lls-logs" in [tags] ) {
    elasticsearch {
      hosts => ["elasticsearch:9200"]
      id => "lls-logs"
      index => "lls-logs-%{+YYYY.MM.dd}"
      codec => "json"
    }
  }
}
```

Building the Elastic Stack

You can now build the Elastic Stack by executing this command in the demo directory:

```
$ docker-compose build
```

Preparing to Use filebeat

Download and expand filebeat into the filebeat directory. This tool will send JSON log entries to the Elastic Stack. You can obtain a copy from www.elastic.co/downloads/beats/filebeat. On Linux you can also use your package manager (DEB, RPM) to install filebeat. Using the package manager will install it as a system service with systemd bindings. In this case the configuration can be found in /etc/filebeat/.
The filebeat distribution contains an example `filebeat.yml` file. Replace it with this:

```
filebeat/filebeat.yml

filebeat.registry.path: $(HOME)/.filebeat-registry

filebeat.config:
  modules:
    path: $(path.config)/modules.d/*.yml
    reload.enabled: false

filebeat.inputs:
  - type: log
    json.keys_under_root: true
    json.overwrite_keys: true
    json.add_error_key: true
    encoding: utf-8
    tags: ["lls-logs"]
    index: "%{[agent.name]}-lls-%{+yyyy.MM.dd}"
    paths:
      - $(HOME)/flexnetls/acme/logs/*.json

output.logstash:
  hosts: ["localhost:5044"]
```

Note that this sample uses the producer name "acme". In your file, replace “acme” with your producer name, as found in `producer-settings.xml`.

**Sending Log Entries to Logstash**

Perform the steps below to log entries to logstash. You can then view the entries in Kibana.

<table>
<thead>
<tr>
<th>Task</th>
<th>To send log entries to logstash</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Bring up the Elastic Stack by executing this command from the demo directory:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$ docker-compose up -d</td>
</tr>
<tr>
<td>2.</td>
<td>Start the license server (if it is not already running).</td>
</tr>
<tr>
<td>3.</td>
<td>Start filebeat running to start sending log entries to Logstash:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$ ./filebeat -e</td>
</tr>
<tr>
<td>4.</td>
<td>Open Kibana with a browser by going to <a href="http://localhost:5601">http://localhost:5601</a>. In Kibana's home page, click <strong>Connect to your Elasticsearch index</strong>.</td>
</tr>
<tr>
<td>5.</td>
<td>Create an index pattern of 'lls-logs-*'. When asked, set '@timestamp' as the primary time field. For information on index patterns, see <a href="http://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html">www.elastic.co/guide/en/kibana/current/tutorial-define-index.html</a>.</td>
</tr>
<tr>
<td>6.</td>
<td>Click <strong>Discover</strong> (in the grid menu); this should display license server log entries. Consult the Kibana documentation for information about searching the log entries.</td>
</tr>
</tbody>
</table>
Elastic Stack and GELF

The docker-compose configuration is the same as in section Creating the "docker-compose" File, but with a change to the Logstash part (note that the log entrypoint key would appear on a single line):

```yaml
version: '2.2'
services:
elasticsearch:
  build:
    context: elasticsearch/
  container_name: elasticsearch
  volumes:
    - monitoring-data:/usr/share/elasticsearch/data
  ports:
    - "9200:9200"
    - "9300:9300"
  environment:
    ES_JAVA_OPTS: "-Xmx512m -Xms512m"
logstash:
  image: docker.elastic.co/logstash/logstash:7.9.1
  container_name: logstash
  ports:
    - "5000:5000"
    - "5044:5044"
    - "12201:12201/udp"
  expose:
    - "5044/tcp"
    - "12201/udp"
  logging:
    driver: "json-file"
  environment:
    LS_JAVA_OPTS: "-Xmx256m -Xms256m"
  entrypoint: logstash -e 'input { gelf { } } filter { json { skip_on_invalid_json => true source => "message" } } output { elasticsearch { id => "lls" hosts => ["http://elasticsearch:9200"] } }'
  depends_on:
    - elasticsearch
kibana:
  image: docker.elastic.co/kibana/kibana:7.9.1
  ports:
    - "5601:5601"
  depends_on:
    - elasticsearch
volumes:
  monitoring-data:
    driver: local
```

The Filebeat agent is not required in this scenario.
Log4J SocketAppender

A SocketAppender can be used to relay log entries to another host. For more information, see logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/net/SocketAppender.html.

Important • Data loss can occur if there are network errors and you use a SocketAppender exclusively. Therefore, the use of Filebeat is generally preferred.

The following procedure describes how to relay entries to another host, using localhost for test purposes.

Task

To relay entries to another host

1. Prepare a custom log4j configuration file, as described in Custom Log Configurations.

2. Add the SocketAppender to the configuration file, using code similar to the following:

```xml
<appender name="RELAY" class="org.apache.log4j.net.SocketAppender" >
  <!-- Demo purposes: relay to localhost -->
  <param name="Port" value="7080" />
  <param name="RemoteHost" value="localhost" />
  <param name="ReconnectionDelay" value="60000" />
  <param name="Threshold" value="INFO" />
</appender>
```

3. Add the appender to the root logger, using code similar to the following:

```xml
<root>
  <level value="INFO"/>
  <appender-ref ref="ASYNC"/>
  <appender-ref ref="RELAY"/>
</root>
```

Try it out

You can test the relay of log entries. log4j includes a simple server that can be used to receive the log messages. You can download log4j.jar from logging.apache.org. The simple server will receive log messages and write them locally. Start with a simple logging configuration (call it log4j-relay.properties) like the following:

```properties
log4j.rootLogger=INFO, file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=logfile.log
log4j.appender.file.MaxFileSize=1MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=[%d] [%t] [%m%n
```

Then start log4j's simple server like this:

```bash
java -classpath log4j-1.2.17.jar org.apache.log4j.net.SimpleSocketServer 7080 log4j-relay.properties
```

Log messages from flexnetls will start appearing in logfile.log. You can confirm this using the `tail -f logfile.log` command (Linux).