

1. LM-X License Manager Documentation	5
1.1 LM-X End Users Guide	6
1.1.1 Resources	7
1.1.2 Getting started	8
1.1.2.1 Optional license features	9
1.1.3 How a protected application finds its license	10
1.1.3.1 License files	11
1.1.3.2 Search paths	13
1.1.3.3 Adding or removing license file paths	14
1.1.3.3.1 Adding or removing license files from the path using the LM-X End-user Configuration tool	15
1.1.3.3.2 Adding license files to the path manually	16
1.1.3.3.3 Formats for specifying license files	17
1.1.3.4 Environment variables	18
1.1.4 End-user tools	21
1.1.4.1 Installing end-user tools and LM-X License Server	22
1.1.4.2 LM-X End-user utility	23
1.1.4.3 LM-X End-user Configuration Tool	25
1.1.4.4 LM-X License Server Client	26
1.1.4.4.1 Accessing LM-X License Server Client	27
1.1.4.4.2 Using LM-X License Server Client	28
1.1.4.4.3 Viewing server information and statistics	30
1.1.4.4.4 Restarting or shutting down the server	31
1.1.4.4.5 Viewing HostIDs	32
1.1.4.4.6 Viewing license usage statistics	33
1.1.4.4.7 Viewing the log file	34
1.1.4.5 LM-X License Server	35
1.1.4.5.1 Protocols	36
1.1.4.5.2 License server configuration file	37
1.1.4.5.3 lmx_server_conf.c file	44
1.1.4.5.4 License server log file	45
1.1.4.5.5 Running the license server from a command line	46
1.1.4.5.6 Unix exit signals	48
1.1.4.5.7 Upgrading the license server	49
1.1.5 Optional features	50
1.1.5.1 Configuring LM-X License Manager for high network connection rates	51
1.1.5.2 High Availability Licensing	52
1.1.5.2.1 How to install HAL license servers	53
1.1.5.3 Pay Per Use feature	54
1.1.5.3.1 Enabling and using usage databases	55
1.1.5.3.2 Database structure	56
1.1.5.3.3 Usage table example	57
1.1.5.4 Borrowing a license	58
1.1.5.5 Automatic server discovery	59
1.1.5.6 Queuing licenses	60
1.1.6 Logs	61
1.1.7 Error messages	64
1.1.8 End-user solutions	65
1.1.8.1 Managing multiple applications running against multiple versions of LM-X	66
1.1.9 LM-X End User's FAQ	67
1.1.9.1 Usage Information	68
1.1.10 Troubleshooting end-user problems	69
1.1.10.1 Enabling extended logging in LM-X	70
1.1.10.2 Communication issues	71
1.1.10.3 Locale issues	72
1.1.10.4 Operating system issues	73
1.1.10.5 System clock check issues	74
1.1.10.6 System security issues	75
1.1.11 LM-X Reference for FlexNet Users	76
1.1.11.1 Comparison of license files	77
1.1.11.2 Comparison of license paths	78
1.1.11.3 Comparison of license server setup	79
1.1.11.4 The LM-X License Server	80
1.1.11.5 Comparison of license server parameters	81
1.1.11.6 Administration	82
1.1.11.6.1 Managing licenses	83
1.1.11.6.2 Enabling and configuring HAL	84
1.1.11.6.3 Enabling and configuring Pay Per Use	85
1.1.11.6.4 Enabling and configuring license server logging	86
1.1.11.6.5 Editing the configuration file	87
1.1.11.6.6 Miscellaneous settings	88
1.1.12 Removed features	89
1.1.12.1 Installing and uninstalling a license server on Windows	90
1.1.12.2 Installing a license server on Mac OS X	92
1.1.12.3 LMX_AtExit	93
1.1.12.4 Web-based UI	94
1.2 LM-X License Manager Developer Documentation	95
1.2.1 LM-X License Manager Quick Start	96
1.2.1.1 Download the LM-X SDK distribution	97
1.2.1.2 Install the LM-X SDK	98

1.2.1.2.1	Install the LM-X SDK on Windows	99
1.2.1.2.2	Install the LM-X SDK on Unix	104
1.2.1.3	Compile the LM-X SDK	106
1.2.1.3.1	Compile the LM-X SDK on Windows	107
1.2.1.3.2	Compile the LM-X SDK on Unix	114
1.2.1.4	Define a license policy	115
1.2.1.4.1	Example XML license template scenarios	118
1.2.1.5	License your first application	119
1.2.1.6	Distribute your LM-X licensed software to end users	123
1.2.1.7	Activate your license using License Activation Center	124
1.2.1.7.1	Different approaches to license activation	125
1.2.2	LM-X Developers Manual	126
1.2.2.1	Getting started with LM-X	127
1.2.2.1.1	Introduction to LM-X	128
1.2.2.1.2	How LM-X works	129
1.2.2.1.3	LM-X distribution content	131
1.2.2.1.4	Application licensing strategies	133
1.2.2.1.5	Controlling license behavior	135
1.2.2.1.6	LM-X community resources and references	136
1.2.2.2	Protecting your application	137
1.2.2.2.1	License file content	138
1.2.2.2.2	Feature descriptions	141
1.2.2.2.3	Optimizing license checkout speed	172
1.2.2.2.4	License server	173
1.2.2.2.5	Supplying a license server for platforms you lack in-house	175
1.2.2.2.6	Using alternative compilers	176
1.2.2.2.7	HostIDs	177
1.2.2.2.8	LM-X security configuration file	182
1.2.2.2.9	LM-X tools	183
1.2.2.2.10	Distributing applications to end users	185
1.2.2.2.11	Trial licenses	189
1.2.2.2.12	Automatic server discovery (floating licenses only)	190
1.2.2.2.13	Heartbeats	191
1.2.2.2.14	License queuing	196
1.2.2.2.15	Secure store	197
1.2.2.2.16	Client store	198
1.2.2.2.17	Blacklisting issued licenses	199
1.2.2.2.18	Licensing for virtual machines and cloud computing	200
1.2.2.2.19	Token-based licensing	201
1.2.2.2.20	Pay Per Use	213
1.2.2.2.21	Dynamic license reservations	214
1.2.2.2.22	Specifying username and hostname for checkouts	215
1.2.2.2.23	Upgrade licenses	216
1.2.2.3	Building	217
1.2.2.3.1	The LM-X protected application and license generator	218
1.2.2.3.2	Generating licenses	219
1.2.2.3.3	LM-X maintenance	231
1.2.2.4	Client API specification	237
1.2.2.4.1	Basic API functions	238
1.2.2.4.2	Advanced API functions	253
1.2.2.4.3	Administrative API functions	307
1.2.2.4.4	Java and .NET client APIs	313
1.2.2.4.5	Return codes	314
1.2.2.5	Developer solutions	317
1.2.2.5.1	Building LM-X for multiple platforms	318
1.2.2.5.2	Building LM-X SDK for both 32-bit and 64-bit .NET framework	319
1.2.2.5.3	Choosing between a floating and node-locked license	320
1.2.2.5.4	Deciding whether to lock a license to one or multiple HostIDs	321
1.2.2.5.5	Determining which HostID to use	322
1.2.2.5.6	Determining which IP address to lock your license to	323
1.2.2.5.7	Detecting a real license	324
1.2.2.5.8	Implementing licensing for critical-use applications	325
1.2.2.5.9	Managing license expiration for licenses running non-stop	326
1.2.2.5.10	Providing demo versions of your software	327
1.2.2.5.11	Restricted access licensing	328
1.2.2.5.12	Using CPU cores	329
1.2.2.5.13	Using dates for version numbers	331
1.2.2.5.14	Using LmxLicenseProvider as an implementation of the abstract base class provided by .NET LicenseProvider	332
1.2.2.6	Advanced topics	333
1.2.2.6.1	Multithreading	334
1.2.2.6.2	IPv6 support	335
1.2.2.6.3	Performing a silent installation	336
1.2.2.6.4	MacOS on ARM support	338
1.2.2.7	LM-X Developer's FAQ	339
1.2.2.7.1	Setup	340
1.2.2.7.2	Training	342
1.2.2.7.3	Usage	346
1.2.2.8	Release Notes	349

1.2.2.8.1 LM-X License Manager 5 Release Notes .....	350
1.2.2.8.2 LM-X License Manager 4 Release Notes .....	358
1.2.2.9 Troubleshooting developer problems .....	432
1.2.2.9.1 Compilation issues .....	433
1.2.2.9.2 Debugging issues .....	434
1.2.2.9.3 Enabling extended logging .....	435
1.2.2.9.4 HASP issues .....	436
1.2.2.9.5 Installation issues .....	437
1.2.2.9.6 Interprocess issues .....	438
1.2.2.9.7 Licensing issues .....	439
1.2.2.9.8 Runtime issues .....	440
1.2.2.9.9 Virtual machine issues .....	441
1.2.2.9.10 High CPU utilization issues .....	442
1.2.2.10 Reporting a bug in LM-X .....	443
1.2.2.11 Supported platforms .....	444
1.2.2.12 Glossary .....	446
2. Node-locked version vs full version .....	447

# LM-X License Manager Documentation

This is the home of X-Formation's LM-X License Manager documentation. Here, you can find information for vendors who use LM-X for license management and for end users who have LM-X licensed applications.

## LM-X License Manager Documentation for Developers

If you are a vendor, see documentation for using LM-X License Manager to protect and license your software applications, which includes:

- [LM-X License Manager Quick Start](#)
- [LM-X Developers Manual](#)

## LM-X License Manager Documentation for End Users

If you are an end user, see documentation for managing LM-X License Manager servers and licensed applications, which includes:

- [LM-X End Users Guide](#)

## Release Notes

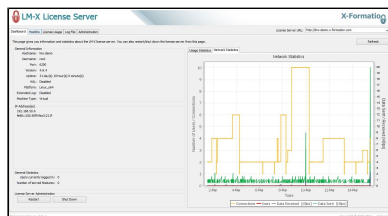
Please refer to [LM-X License Manager Release Notes](#) for details on updates and features introduced in specific versions.

## Resources

If you have a question about using LM-X License Manager, please contact our [support team](#). You may also want to [submit feedback or suggest new features](#) in LM-X.

Other handy links:

- [Troubleshooting developer problems](#)
- [Troubleshooting end user problems](#)
- [LM-X Developer's FAQ](#)



# LM-X End Users Guide

The *LM-X End Users Guide* is intended for corporate use by system administrators of LM-X protected applications, as well as engineers, software developers and others who are end users of LM-X-served applications.

In addition to this document, if you are planning to use a network license, you should have obtained an LM-X license server, an LM-X license file, and an LM-X license server configuration file (optional) from your application vendor.

This guide includes information to help you administrate your LM-X protected applications, including a description of LM-X license files, how to use LM-X end user tools, and how to install/uninstall an LM-X license server.

Software application vendors should refer to the *LM-X Developers Manual* for information about using LM-X to license their products.

## Resources

As part of the LM-X License Manager user community, you have access to our Knowledgebase at <https://kb.x-formation.com> to get answers to commonly asked questions about LM-X product features, installation, usage and related topics. The Knowledgebase is updated regularly with new information to help you quickly and easily find the answers you're looking for.

For those transitioning from a FlexNet/FLEXlm license management system to LM-X, this document includes a helpful [reference of LM-X equivalencies](#) for some common FlexNet/FLEXlm actions.

# Getting started

Getting your LM-X licensed application up and running takes little to no effort. Depending on what type of licensing you have for the application, you will need the following:

Type of License	What you need
Trial	No license or setup required.
Standalone (also referred to as node-locked)	A license file installed on, and often locked to, an individual machine.
Network (also referred to as floating or shared)	A license server deployed at your site and a license file that is <a href="#">installed on that license server</a> .

The [LM-X End-user Configuration Tool](#) helps you obtain information your vendor may request from you in order to create your node-locked or floating license; for example, your application vendor may ask for your HostID.

The LM-X End-user Configuration Tool also gives you a quick and easy way to add or remove license files from your license path, install a license server, and query a license server all from one simple dialog. These same tasks can also be run from a command line on either Windows or Unix with the [LM-X End-user utility](#).

## Optional license features

LM-X includes the following optional features that make your network license usage easier and more reliable. The features available to you depend on what your software vendor has provided. Contact your software vendor for more information about available features, or if you have a [license server configuration file](#) included in your software distribution, review this file to determine which features are included in your license.

Feature	Description
Automatic server discovery	Locates the LM-X License Server automatically. All you need to do is install the license server and application, and LM-X does the rest to ensure you get up and running right away.
High-availability licensing (HAL)	<p>Lets you specify backup (redundant) license servers that will continue to enable license hosting in the event that your primary license server goes down.</p> <p>See <a href="#">How to install HAL license servers</a> for more information on HAL.</p>
License queuing	<p>Issues licenses based on a list of requests that are waiting for a license. When licenses are in high demand, requests for a license can be added to a queue, and then filled in the order of the queue as licenses become available. This is particularly useful for ensuring proper scheduling for automatic jobs and implementing fair usage of shared licenses.</p> <p>To enable license queuing, you must set the environment variable LMX_QUEUE as described in <a href="#">Environment variables</a>. Also see <a href="#">Queuing licenses</a> for further information on license queuing.</p>
License borrowing	<p>Lets you use a license to run a network application for a limited time when you are unable to connect to the license server. For example, you can borrow a license if you are taking your machine home for the weekend or going on a business trip. Effectively, borrowing a license gives you a temporary node-locked license.</p> <p>To enable license borrowing, you must set the environment variable LMX_BORROW, as described in <a href="#">Environment variables</a>.</p>
Grace licenses	Lets you keep using a network license for a specified period of time when the license server is down, ensuring uninterrupted access to the application so you can complete your work.

In addition to the features listed above, optional features that system administrators can use in order to control usage of application licenses include:

- Allow/deny users/groups from using the license server (including/excluding users and computers) to prevent specified users from accessing applications. This may be used as an additional security measure.
- Limit the number of licenses that can be used by individual users or groups to implement fair/desired distribution of licenses.
- Reserve a number of licenses that can be used by individual users or groups to implement fair/desired distribution of licenses.

For information about using these options, see [License server configuration file](#).



## How a protected application finds its license

This chapter describes the different license file types, where an application searches for its license, and how to add or remove license files from the license path environment variable. It also describes additional environment variables you can optionally set for your license.

# License files

*The information on this page refers to LM-X v5.0, which added the upgrade license type. If you are using an older version of LM-X, please refer to [documentation for earlier versions](#).*

Licenses are text files that can be accessed by a licensed application in two distinct ways:

- Locally, from a local hard drive
- Remotely, from a dedicated license server across a network

If you open a license file in a text editor, you will see a definition for the license in the following format:

```
FEATURE Feature_Name
{
  VENDOR = XYZ
  ...
  COUNT = 5
  ...
}
```

When the COUNT keyword exists in the license file, the license is intended for use on a license server; otherwise, the license is considered to be local and can be used directly with the protected application. The example above indicates that there are 5 network licenses for the application.

The following is an example of a local license file (the actual key has been abbreviated):

```
FEATURE f1
{
  VENDOR = XFORMATION
  VERSION = 1.5
  END = 2015-01-01
  KEY = F2DNR9K...
}
```

The following is an example of a network license file (the actual key has been abbreviated):

```
FEATURE f1
{
  VENDOR = XFORMATION
  VERSION = 1.5
  END = 2015-01-01
  KEY = F2DNR9K...
  COUNT = 12
}
```

The license file path may be set in the license server configuration file. See [License server configuration file](#) for more information about specifying the license server path.

The vendor may include various optional settings in a license that supply license information or control how the license may be used. The settings that can be included in a license are shown below.

```

VERSION = (Version number of the feature.)
LICENSEE = (User or company to whom the license was issued.)
START = (Date on which the feature becomes valid.)
END = (Date on which the feature expires.)
MAINTENANCE_START = (Date the license maintenance plan begins.)
MAINTENANCE_END = (Date the license maintenance plan expires.)
ISSUED = (Date the license was created.)
SN = (Custom serial number for the license.)
DATA = (Additional information regarding the license.)
COMMENT = (Additional information regarding the license.)
OPTIONS = (Additional licensing options.)
PLATFORMS = (Platform(s) to which usage is restricted.)
COUNT = (Number of network licenses that can be issued simultaneously.)
TOKEN_DEPENDENCY = (Reference to a real license upon which a token license depends.)
SOFTLIMIT = (Number of "overdraft" licenses.)
HAL_SERVERS = 3 (Enables redundant servers, or High Availability Servers.)
BORROW = (Number of hours a borrowed license may be used.)
GRACE = (Number of hours a grace license may be used.)
HOLD = (Number of minutes licenses are held before being checked in.)
USERBASED = (Number of licenses reserved for named users.)
HOSTBASED = (Number of licenses reserved for named hosts.)
TIME_ZONES = -12 to 13 (Allowed time zones, relative to GMT.)
SHARE = HOST|USER|CUSTOM or TERMINALSERVER and/or VIRTUAL (Type of license sharing in use.)
SYSTEMCLOCKCHECK = TRUE|FALSE (Enables/disables the system clock check.)
HOSTID_MATCH_RATE = (Percentage of hostids that must match for successful hostid verification.)

VERSION = 0.0 to 9999.9999 (Version number of the feature.)
LICENSEE = "string" (User or company to whom the license was issued.)
START = YYYY-MM-DD (Date on which the feature becomes valid.)
END = YYYY-MM-DD (Date on which the feature expires.)
MAINTENANCE_START = YYYY-MM-DD (Date the license maintenance plan begins.)
MAINTENANCE_END = YYYY-MM-DD (Date the license maintenance plan expires.)
ISSUED = YYYY-MM-DD (Date on which the license was created.)
SN = "string" (Custom serial number for the license.)
DATA = "string" (Additional information regarding the license.)
COMMENT = "string" (Additional information regarding the license.)
OPTIONS = "string" (Additional licensing options for the license.)

COUNT = 1 to 2147483647 or UNLIMITED (Number of network licenses that can be issued simultaneously)

PLATFORMS = "platform strings" (Platform(s) to which usage is restricted.)

TOKEN_DEPENDENCY = "FEATURE=FeatureName VERSION=0.0 to 9999.9999 COUNT=1 to 2147483647"
KEYTYPE = EXCLUSIVE or ADDITIVE or UPGRADE or TOKEN (Reference to a real license upon which a token license depends.)
SOFTLIMIT = 5 (Number of "overdraft" licenses.)
HAL_SERVERS = 3 (Enables redundant servers, or High Availability Servers.)
BORROW = 1 to 8760 (Number of hours a borrowed license may be used.)
GRACE = 1 to 168 (Number of hours a grace license may be used.)
HOLD = 1 to 1440 (Number of minutes licenses are held before being checked in.)
USERBASED = 1 to 2147483647 or ALL (Number of licenses reserved for named users.)
HOSTBASED = 1 to 2147483647 or ALL (Number of licenses reserved for named hosts.)
TIME_ZONES = -12 to 13 (Allowed time zones, relative to GMT.)

SHARE = HOST|USER|CUSTOM or TERMINALSERVER and/or VIRTUAL (Type of license sharing in use.)

SYSTEMCLOCKCHECK = TRUE|FALSE (Enables/disables the system clock check.)
HOSTID_MATCH_RATE = 0 to 100 (Percentage of hostids that must match for successful hostid verification.)

```

**Note:** If there are multiple license files that contain the same feature (for example, a license file that contains "feature f1 version 1.0" and another license file that contains "feature f1 version 1.5"), LM-X will only use the first license file that is read. Any other license files that contain the same feature will be ignored. If this occurs, you will see a warning message in the [license server log](#) that informs you which license files contain the same feature.

In such cases, you should delete the license files that are not needed or [remove the license file from the search path](#) so the unneeded license files are not read.

## Search paths

Every LM-X protected application has a search path for licenses. The application will search for licenses in the following order.

Order	Search Path
1	Preset path and/or automatic server discovery (this information is provided by your application vendor).
2	The environment variable <code>VENDOR_LICENSE_PATH</code> . The <code>VENDOR</code> name is the same as that specified in the license file.
3	The generic environment variable <code>LMX_LICENSE_PATH</code> , which is used by all applications protected by LM-X.

**Note:** Paths are separated with a semicolon ( ; ) on Windows systems and a colon ( : ) on Unix systems.

Using the above search paths, the application will try to find a license in the following order:

1. Borrowed license.
2. License embedded as a string in the software.
3. Node-locked (local) license.
4. Network (floating) license (this includes automatic server discovery).
5. Grace license.
6. Trial license.

**Note:** LM-X attempts to use local licenses before it tries to use counted network licenses to optimize license utilization.

At each step, if no license is found in the specified source, the application will continue to the next source in the path. You may specify unlimited multiple paths if desired; for example under Windows, `LMX_LICENSE_PATH=6200@server1;6200@server2;6200@server3`. The protected application stops searching as soon as it finds a valid license.

## Adding or removing license file paths

The following sections describe how to add or remove license files from the path for Windows and Unix systems and gives formats for specifying license files.

## Adding or removing license files from the path using the LM-X End-user Configuration tool

Under Windows, you can use the [LM-X End-user Configuration Tool](#), lmxconfigtool.exe, to add or remove license files from the LMX\_LICENSE\_PATH environment variable. The syntax of these paths is:

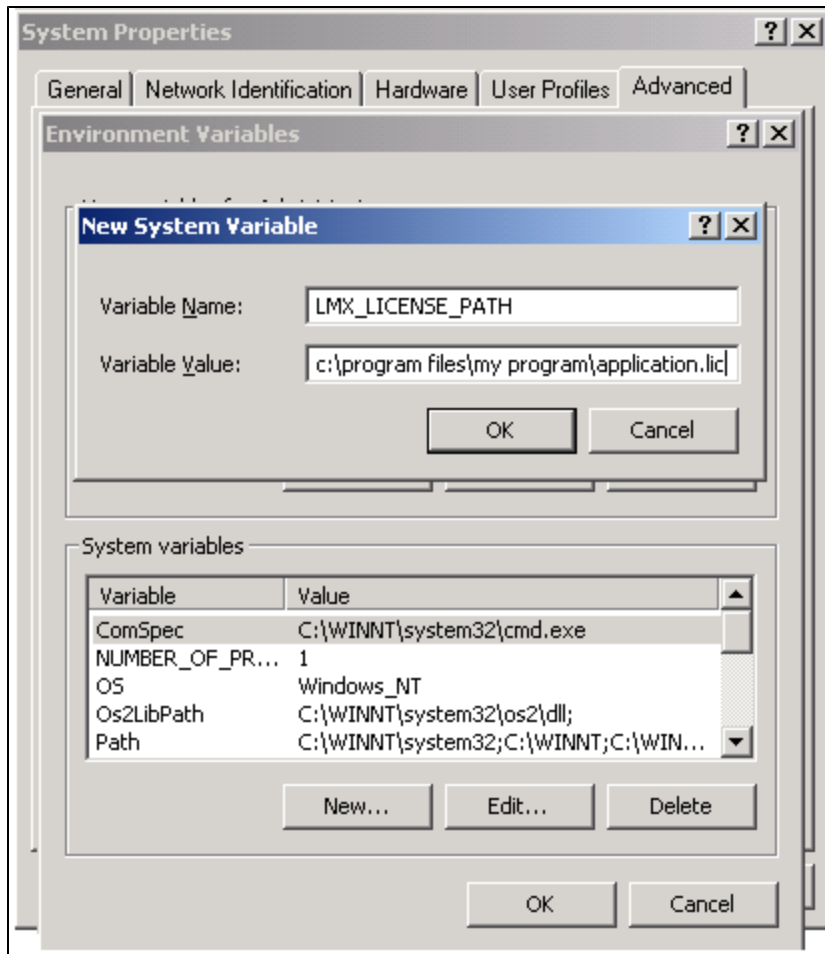
Platform	Syntax
Windows	LMX_LICENSE_PATH = file1;file2;host%port:@host2;directory;...
Unix	LMX_LICENSE_PATH = file1:file2:host%port:@host2:directory:...

Note that multiple paths are separated with a semicolon ( ; ) on Windows systems and a colon ( : ) on Unix systems.

## Adding license files to the path manually

If needed, you can add license files to the path manually, as described below. This method can also be used to specify other environment variables, described in [Environment variables](#).

- On Windows, you can use the system control panel.



- On Unix systems, you can add the following in `/etc/profile` (or see your system documentation):  
`export LMX_LICENSE_PATH=/path/to/license.lic`

# Formats for specifying license files

Possible formats for specifying license files are detailed in the following table.

Type	Description	Syntax	Example
Local license	Full path to license file	Windows: C:\path\to\license.lic  Unix: /path/to/license.lic	Windows: C:\application\licenses\mylicense.lic  Unix: /home/henrik/application/nl.lic
Local license	Full path to license directory – all licenses (*.lic) in specified directory will be read	Windows: C:\application\licenses{color}  Unix: /application/licenses/	Windows: C:\application\licenses{color}  Unix: /application/licenses/
Network license on license server	Network license stored on a specified license server (default port is 6200)	hostname%tcpport	intranet.foobar.com%5678
Network license on license server	Network license stored on a specified license server with an optional specified port (defaults to port 6200)	[tcpport]@hostname	@intranet.foobar.com
Network HAL license	Network license installed on 3 redundant servers	Windows: @hostname1;@hostname2; @hostname3  Unix: @hostname1:@hostname2: @hostname3	Windows: @primary_server; @secondary_server;@third_server  Unix: @primary_server: @secondary_server:@third_server

**Note:** You should always consult with your application vendor on how to set up their application.



# Environment variables

You can set the environment variables listed in the table below as needed. Note that the environment variables available to you may be limited depending on whether your vendor has allowed their use as part of your software license.

All environment variables are undefined by default, and can be defined by setting the value to a number greater than 0, for example, 1. Any additional details on variable values are given in the Value column below.

**Important:** You must restart the client for changes to environment variables to take effect.

Variable name	Value	Description
<code>VENDOR_LICENSE_PATH</code> or <code>LMX_LICENSE_PATH</code>	The path to one or more LM-X managed licenses.  For example: <code>LMX_LICENSE_PATH=6200@server1</code>	This environment variable lets you set the path to the license file.  You may specify a particular vendor using <code>VENDOR_LICENSE_PATH</code> , where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_LICENSE_PATH</code> is a generic environment variable used by all applications protected by LM-X.  See <a href="#">Search paths</a> for information on how an LM-X application searches for licenses.  You can set the license path using the <a href="#">LM-X End-user Configuration tool</a> . Adding or removing license files from the path using the LM-X End-user Configuration tool, or manually, as described in section 3.3.2, Adding license files to the path manually.
<code>VENDOR_AUTOMATIC_SERVER_DISCOVERY</code> or <code>LMX_AUTOMATIC_SERVER_DISCOVERY</code>	A string that can be set to 1 (or any other integer) to enable running an automatic server discovery.  For example: <code>LMX_AUTOMATIC_SERVER_DISCOVERY=1</code>	When this environment variable is set, automatic server discovery is enabled. You may specify a particular vendor using <code>VENDOR_AUTOMATIC_SERVER_DISCOVERY</code> , where <code>VENDOR</code> is the name of the application vendor, as specified in the license file.  <code>LMX_AUTOMATIC_SERVER_DISCOVERY</code> is a generic environment variable used by all applications protected by LM-X.  You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a> .
<code>VENDOR_AUTOMATIC_SERVER_DISCOVERY_SERVER</code> or <code>LMX_AUTOMATIC_SERVER_DISCOVERY_SERVER</code>	A string that can be set to 1 (or any other integer) to enable running an automatic server discovery within a client.  For example: <code>LMX_AUTOMATIC_SERVER_DISCOVERY_SERVER=1</code>  To disable automatic server discovery, unset the environment variable.  For example: <code>LMX_AUTOMATIC_SERVER_DISCOVERY_SERVER=</code>	When this environment variable is set in combination with <code>LMX_AUTOMATIC_SERVER_DISCOVERY</code> (described above), the client will start responding to automatic server discovery requests issued by other clients. This enables the client to relay information about another server, thereby acting as a proxy.  Only one Automatic Server Discovery server (either a license server or one client acting as a server) can be active at one time on one machine. This is handled automatically.  Example: You may have a license server over the Internet and several clients on a local network. In this situation, the first client can enable the broadcast server and the other clients on the same local network will get the server address from the client machine instead of manually setting the server address. For example:  <ol style="list-style-type: none"> <li>1. Client A has enabled the <code>LMX_AUTOMATIC_SERVER_DISCOVERY</code> and <code>LMX_AUTOMATIC_SERVER_DISCOVERY_SERVER</code> flags.</li> <li>2. Client A checks out a license for Vendor A from Server A. Server A is located on a remote network, where automatic server discovery is not working due to firewall restrictions.</li> <li>3. Client B requests automatic server discovery to check out the Vendor A software.</li> <li>4. Client A gets the automatic server discovery request from Client B, and manually sets the address of the server, and then Client B gets the server information from Client A.</li> </ol> You may specify a particular vendor using <code>VENDOR_AUTOMATIC_SERVER_DISCOVERY_SERVER</code> , where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_AUTOMATIC_SERVER_DISCOVERY_SERVER</code> is a generic environment variable used by all applications protected by LM-X.  You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a> .

<code>VENDOR_BORROW</code> or <code>LMX_BORROW</code>	<p>The desired number of hours to allow license borrowing, from 1 - 8760. or &lt;0 (e.g., -1) to allow early checkin of borrowed licenses.</p> <p>Examples: To set the borrow limit to 2 days:  <code>LMX_BORROW=48</code> To allow early checkin: <code>LMX_BORROW=-1</code></p>	<p>Setting this environment variable to a number greater than 0 sets the number of hours for license borrowing.</p> <p>Setting this environment variable to a number less than 0 enables early return of borrowed licenses.</p> <p>You may specify a particular vendor using <code>VENDOR_BORROW</code>, where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_BORROW</code> is a generic environment variable used by all applications protected by LM-X.</p> <p>You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a>.</p>
<code>VENDOR_BORROW_FORCE_RETURN</code> or <code>LMX_BORROW_FORCE_RETURN</code>	<p>A string that can be set to enable forcing a borrow return. Valid values are 0 or 1.</p> <p>For example: <code>LMX_BORROW_FORCE_RETURN = 1</code></p>	<p>When this environment variable is set to 1 and <code>LMX_BORROW=-1</code> (see above), the borrowed feature will be returned on the client side, even if there is no connection with the license server.</p> <p>You may specify a particular vendor using <code>VENDOR_BORROW_FORCE_RETURN</code>, where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_BORROW_FORCE_RETURN</code> is a generic environment variable used by all applications protected by LM-X.</p> <p>Caution: Use this variable carefully, because it can create an inconsistency between the client and the license server.</p>
<code>LMX_RANDOMIZE_PATH</code>	<p>A string that can be set to enable using a random path.</p> <p>For example: <code>LMX_RANDOMIZE_PATH=1</code></p>	<p>When this environment variable is set, LM-X chooses a random path from a list for each server request. If you have multiple license servers, you can set this variable to balance the load amongst the servers.</p> <p>You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a>.</p>
<code>VENDOR_EXTENDEDLOG</code> or <code>LMX_EXTENDEDLOG</code>	<p>The path to the extended client-side log.</p> <p>For example: <code>LMX_EXTENDEDLOG=C:\LM-X\My_LM-X_Log_Directory\client.log</code></p>	<p>This environment variable lets you set the path and filename for the extended client-side log, which contains information about all client activity.</p> <p>You may specify a particular vendor using <code>VENDOR_EXTENDEDLOG</code>, where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_EXTENDEDLOG</code> is a generic environment variable used by all applications protected by LM-X.</p> <p><b>Important: Using extended logging delays checkouts up to 15 seconds, because it gathers more information about environment than when using normal logging.</b></p> <p>You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a>.</p>
<code>VENDOR_CONNECTION_TIMEOUT</code> or <code>LMX_CONNECTION_TIMEOUT</code>	<p>The desired number of seconds, from 1 - unlimited. The default value is 30.</p> <p>For example: <code>LMX_CONNECTION_TIMEOUT=10</code></p>	<p>This environment variable lets you adjust the connection timeout setting. The connection timeout is set to 30 seconds by default unless you set it to a different value using this environment variable.</p> <p>Increasing the timeout value can be useful for highly loaded networks. For example, if you set <code>LMX_CONNECTION_TIMEOUT = 60</code>, the client can wait up to 60 seconds before timeout.</p> <p>You may specify a particular vendor using <code>VENDOR_CONNECTION_TIMEOUT</code>, where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_CONNECTION_TIMEOUT</code> is a generic environment variable used by all applications protected by LM-X.</p> <p>You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a>.</p>
<code>VENDOR_PROJECT</code> or <code>LMX_PROJECT</code>	<p>A string specifying a project name.</p> <p>For example: <code>LMX_PROJECT=Doorlatch_Design</code></p>	<p>This environment variable lets you set a project name for which all or vendor-specific LM-X licensed applications are being used. The project name is reported in <code>lmxendutil -licstat</code>. (See <a href="#">LM-X End-user utility</a>.)</p> <p>This lets you track for what purpose the application was used. For example, an application may be used for three different projects being run under different departments. Tracking which of the three projects the application was used for can help with accurate cost splitting amongst the projects.</p>
<code>VENDOR_QUEUE</code> or <code>LMX_QUEUE</code>	<p>A string that can be set to enable license queuing.</p> <p>For example: <code>LMX_QUEUE=1</code></p>	<p>This environment variable enables license queuing for all checkout requests.</p> <p>You may specify a particular vendor using <code>VENDOR_QUEUE</code>, where <code>VENDOR</code> is the name of the application vendor, as specified in the license file. <code>LMX_QUEUE</code> is a generic environment variable used by all applications protected by LM-X.</p> <p>You can set this environment variable in the same manner as described in <a href="#">Adding license files to the path manually</a>.</p>

TMPDIR	<p>A string specifying a path.</p> <p>For example: TMPDIR=/var/tmp</p>	<p>This system environment variable lets you specify the directory to be used for temporary files.</p> <p>You can set TMPDIR before running your LM-X licensed application if you know in advance that you will not have access to /tmp directory.</p> <p>TMPDIR affects UNIX platforms only.</p>
--------	--	---

## End-user tools

*Some information on this page refers to LM-X v4.8 and newer, which replaced the LM-X web-based UI with LM-X License Server Client, an independent window application. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

This chapter describes [installing](#) and using LM-X end-user tools, which include the following.

Tool	Windows File Name	Unix File Name
<a href="#">LM-X License Server</a>	lmx-serv.exe	lmx-serv
<a href="#">LM-X end-user utility</a>	lmxendutil.exe	lmxendutil
<a href="#">LM-X End-user Configuration Tool</a>	lmxconfigtool.exe	N/A (Windows only)
<a href="#">LM-X License Server Client</a>	LicserverClient.jar	LicserverClient.jar

# Installing end-user tools and LM-X License Server

*The information on this page refers to v4.4.3 and later, which introduced an installation program that installs both LM-X License Server and the end-user tools. For installation instructions applicable to earlier versions, see [Removed Features](#).*

## Installing end-user tools and LM-X License Server

An installation program lets you quickly and easily install [end-user tools](#), as well as the [LM-X License Server](#). The installer will allow you to [install the license server as a service](#) for Windows, Linux and Mac. (For other [platforms](#), you will need to start the license server as a daemon manually.)

**Note:** You can also [perform a silent installation](#).

To download the latest installation program, go to [http://www.x-formation.com/lm-x\\_license\\_manager/enduser\\_downloads.html](http://www.x-formation.com/lm-x_license_manager/enduser_downloads.html). For previous versions, contact your application vendor.

When you run the installation program, a wizard guides you through each installation step. The installation program optionally lets you install LM-X License Server as a service and helps you locate and copy the liblmxvendor library, which is required for starting the LM-X License Server. This library and any updates to it are supplied by your application vendor.

## Uninstalling LM-X License Server on Linux

To uninstall LM-X License Server on Linux:

1. Remove the directory where your SDK is installed (for example, /home/john/lmx-sdk-4.8.1).
2. If you installed the end-user tools, remove the directory where your [end-user tools](#) are located.
3. Remove init script from /etc/init.d/lmxserv{version} (for example, lmxserv481 for LM-X version 4.8.1).  
Please note that you need to switch to root privilege or use sudo to be able to remove init script.

Also see [Installing and uninstalling a license server on Windows](#) for detailed steps on how to install a license server as a service on Windows.

# LM-X End-user utility

*The information on this page refers to LM-X v4.9.5 or newer, which includes syntax changes for `licstat` and `licstatxml` commands. If you are using a previous version of LM-X, please see the documentation relevant to your version: [v4.4.3 documentation](#); [v4.4 documentation](#); [documentation for versions prior to v4.4](#).*

The LM-X End-user utility lets you get the HostID values for the computer system. For machine-locked licenses, application vendors will use HostID values to create licenses specifically for your system. The LM-X End-user utility also lets you:

- See who is currently using specific licenses on the license server, and see the borrow, grace and trial licenses currently checked out
- Remotely access the license server
- Remove users from the license server
- Read and verify a usage database and print the usage information to the screen

The LM-X End-user utility may be run by any user; you do not need administrator privileges to run the utility. The `lmxendutil` command usage is as follows. Optional parameters are enclosed in square brackets ( [ ] ). Variables are shown in *italic text*.

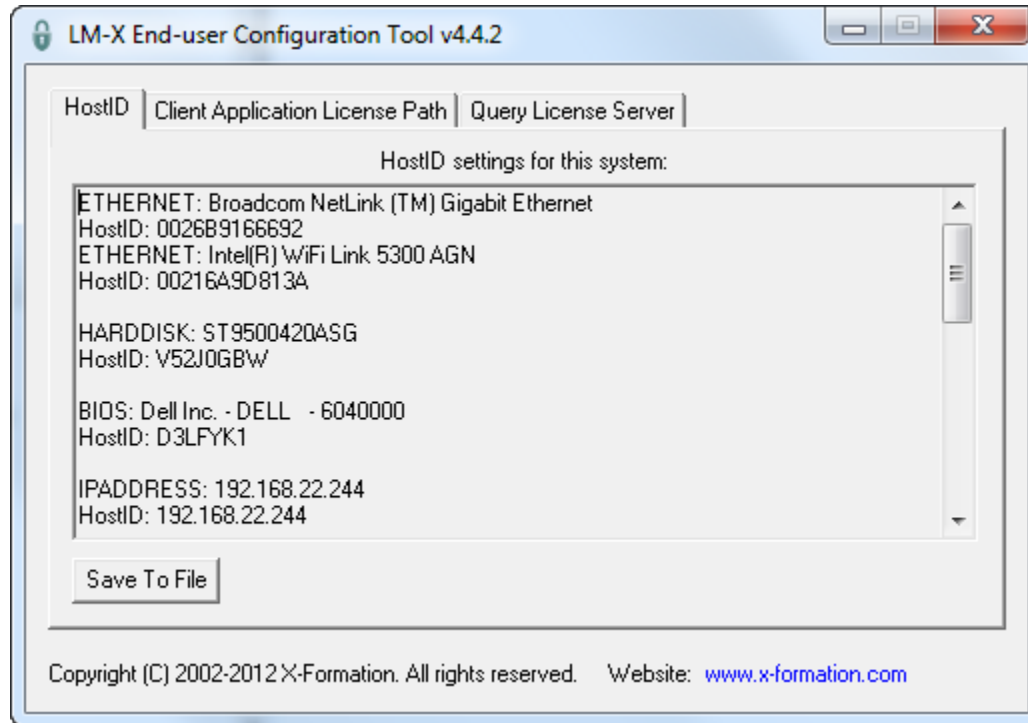
Command	Syntax	Description
-hostid	<code>lmxendutil -hostid</code>	Displays HostID values of your computer system.
-licstat	<code>lmxendutil -licstat [-vendor &lt;vendor_name&gt; (-host &lt;host&gt; -port &lt;port&gt;   -hal &lt;servers&gt;) -network -borrow -grace -trial -denials]</code>	<p>Displays statistics and lists which users are currently using which licenses on a specific license server</p> <p><b>Note:</b> The <code>lmxendutil -licstat</code> command does not use the environment variable <code>LMX_LICENSE_PATH</code> when querying the license server. A client application can find license servers on the network automatically, using <a href="#">automatic server discovery</a> or the <code>-host</code> option.</p> <p>When <code>-vendor</code> is used, the statistics will be limited to the specified vendor. If <code>-vendor</code> is not used, information such as grace, borrow and trial checkouts may not be returned, because the vendor is unknown.</p> <p>Information for the borrow, grace and trial licenses currently checked out is also returned unless one or more options are specified; for example, specifying <code>-borrow</code> will result in only borrow information being returned. If the <code>-denials</code> option is specified, this command additionally prints detailed information about denials for current and disconnected users for the past 24 hours, including username, hostname, IP address, login times, and denial times.</p> <p>See below for descriptions of optional parameters <code>-hal</code>, <code>-host</code>, and <code>-port</code>.</p>
-licstatxml	<code>lmxendutil -licstatxml [-vendor &lt;vendor_name&gt; (-host &lt;host&gt; -port &lt;port&gt;   -hal &lt;servers&gt;) -network -borrow -grace -trial -denials]</code>	Displays the same information as for <code>-licstat</code> , but in XML format.
-restartserver	<code>lmxendutil -restartserver [-host <i>host</i> -port <i>port</i> -password <i>password</i>]</code>	<p>Remotely restarts the license server.</p> <p>See below for descriptions of optional parameters <code>-host</code>, <code>-port</code> and <code>-password</code>.</p>
-shutdownserver	<code>lmxendutil -shutdownserver [-host <i>host</i> -port <i>port</i> -password <i>password</i>]</code>	<p>Remotely shuts down the license server.</p> <p>See below for descriptions of optional parameters <code>-host</code>, <code>-port</code> and <code>-password</code>.</p>
-removeuser	<code>lmxendutil -removeuser -clientusername <i>username</i> -clienthostname <i>host</i> [-host <i>host</i> -port <i>port</i> -password <i>password</i>]</code>	<p>Removes a user from the license server. Parameter descriptions are as follows:</p> <p><code>-clientusername <i>username</i></code> Removes a user with the specified username.</p> <p><code>-clienthostname <i>host</i></code> Removes a user at the specified hostname.</p> <p>See below for descriptions of optional parameters <code>-host</code>, <code>-port</code> and <code>-password</code>.</p>
-hal	<code>-hal <i>server1 server2 server3</i></code>	<p>Optional parameter for specifying HAL servers, used only when HAL is enabled. You must specify three servers in <code>port@host</code> format; for example:</p> <p><code>lmxendutil -licstat -vendor VENDORNAME -hal 6200@localhost 6300@localhost 6400@localhost</code>  <code>lmxendutil -licstatxml -vendor VENDORNAME -hal 6200@localhost 6300@localhost 6400@localhost</code></p>
-host	<code>-host <i>host</i></code>	Optional parameter that connects to the specified license server host. If you don't specify the <code>-host</code> option, <code>lmxendutil</code> command will perform autodiscovery.
-port	<code>-port <i>port</i></code>	Optional parameter that connects to license server on port ' <i>myport</i> '. If you do not enter this optional parameter, the default port is used.

-password	-password <i>password</i>	Optional parameter that uses the specified password. If you do not use this optional parameter explicitly, you will be prompted for the password (in this case, the password you enter is not displayed on the screen as you type).
-readusagedb	lmxendutil -readusagedb <i>usage.db</i>	Reads the specified usage database and performs a verification, then prints the usage information to the screen. (See <a href="#">Pay Per Use feature</a> .)

# LM-X End-user Configuration Tool

The End-user Configuration Tool is a Windows-only tool that lets you:

- See the HostID values of your computer system (in the HostID tab as shown in the example below)
- Edit the license path environment variable (see [Adding or removing license files from the path using the LM-X End-user Configuration tool](#) for more information)
- Query a license server



**Note:** After making changes to the license path, reboot to ensure that the changes take effect.



# LM-X License Server Client

LM-X License Manager uses LM-X License Server Client, an independent window application that helps you to monitor and manage your LM-X License Server.

Using LM-X License Server Client, you can:

- View information and statistics about the LM-X License Server.
- Restart or shut down the license server.
- View the LM-X server's HostIDs (unique machine values that can be used to lock a license file to a host), such as Ethernet, Hostname, IP Address, etc.
- View license usage statistics for the current license server, as well as for borrow, grace and trial licenses.
- View and edit the configuration file. (A password is required to access the configuration file.)
- View and search for entries in the log file.
- Select the license server for which you wish to manage statistics by entering the license server URL. (For more information, see [Using LM-X License Server Client](#).)

## Accessing LM-X License Server Client

If you are using Windows, we recommend using a shortcut in the Windows Start menu to access LM-X License Server Client. Otherwise, go to your end-user tools directory and directly run a LicserverClient.jar file.

You can access the host of the LM-X license server on the same port as the license server, which defaults to 6200 (for example, <http://www.lmx-server-host.com:6200>).

To access LM-X License Server Client, you must have JRE installed on your machine.

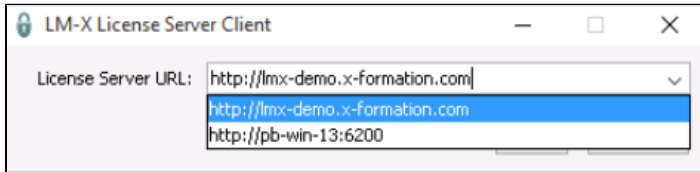
Once you have accessed the application, you can start [using LM-X License Server Client](#).

# Using LM-X License Server Client

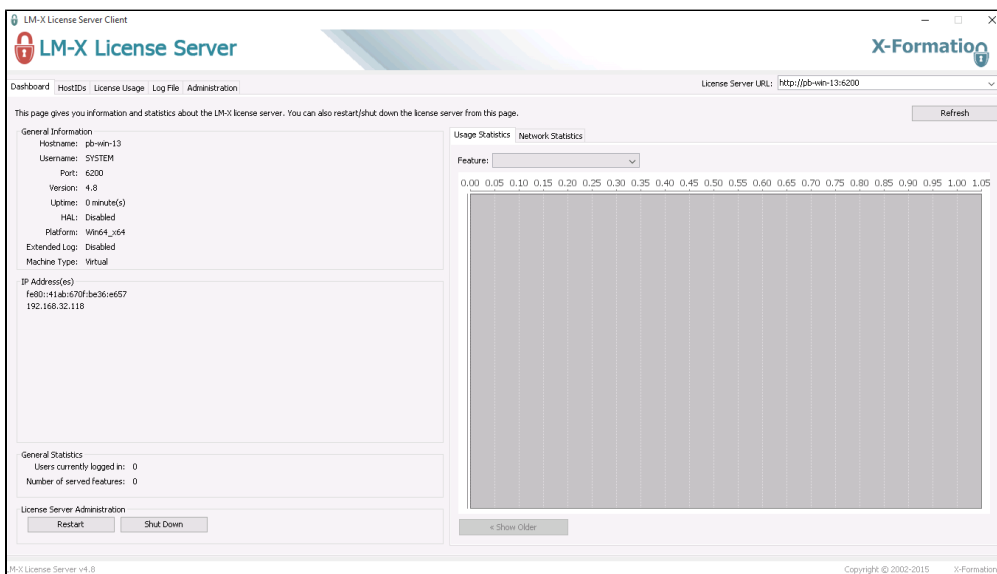
## Selecting a license server

The first time you access LM-X License Server Client, a small dialog with a combo box pops up that lets you either select a LM-X license server's IP address as the URL from the list or type it directly. You can choose from the following URLs:

- <http://lmx-demo.x-formation.com> (the demo server which you can use to test the application)  
and
- your local host of the LM-X license server, which defaults to 6200 (for example <http://lmx-server-host:6200>).

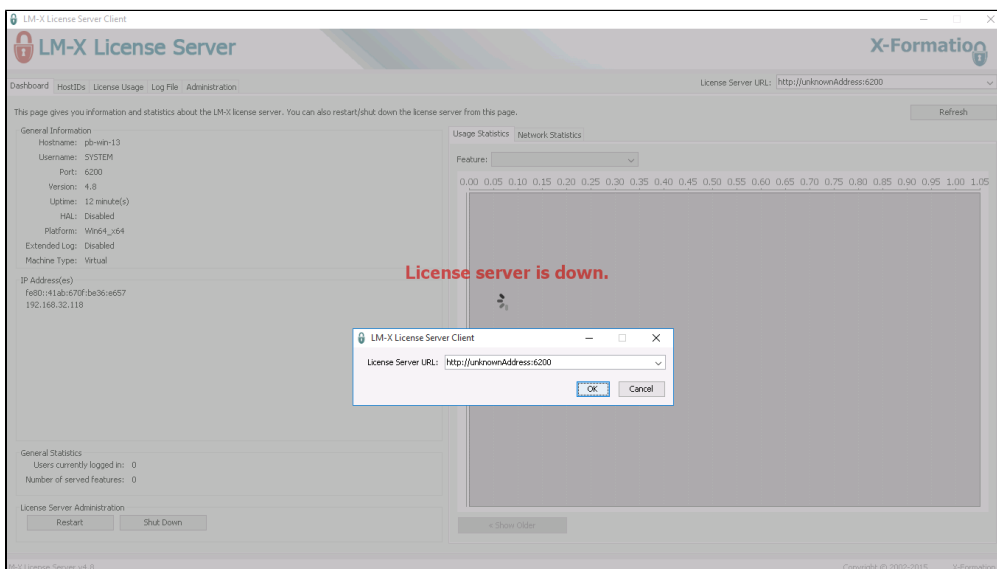


After you make your selection, you can easily view and manage the license server of your choosing, as shown below.

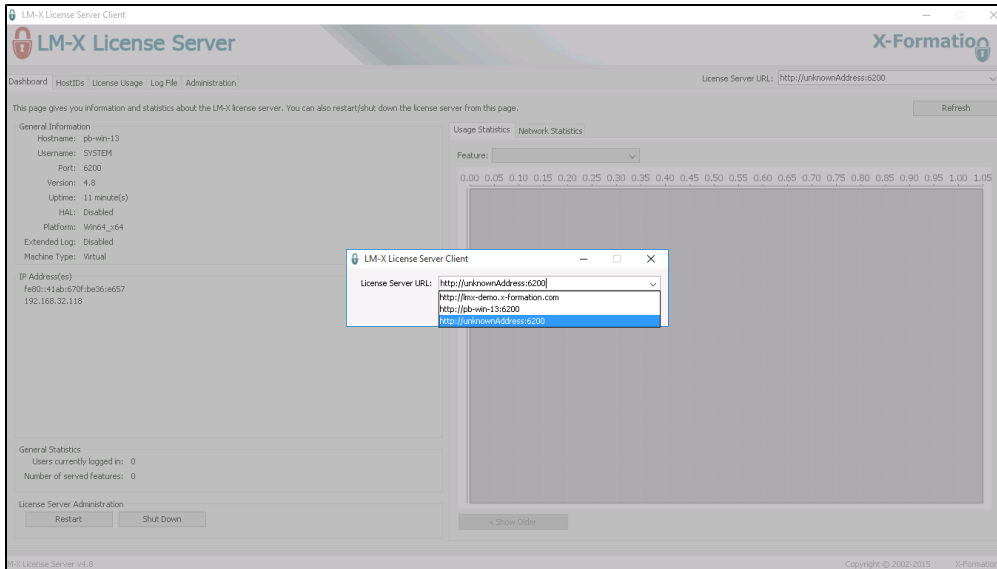


## Selecting a license server that is down

When you select a server which is down, then the main window turns grey and the message "License server is down" is displayed, as shown below.

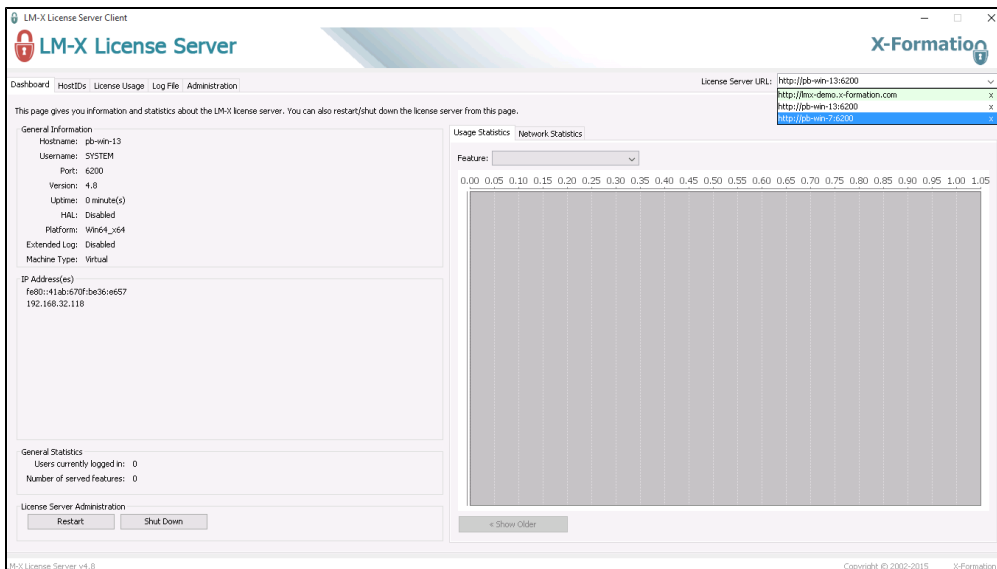


You are then prompted to choose from other servers you added earlier or add a new server by typing its URL in the dialog that appears, as shown below.



## Removing a license server

To remove a license server's IP address from the License Server URL combo box, click "X" to the right of the URL you wish to remove, as shown below.



After removing the URL, you will be automatically switched to the server above the one you have removed.

**Note:** You cannot delete all servers – when only one server is left, "X" disappears.

## Viewing server information and statistics

*The information on this page refers to LM-X version 4.1, which added new graphs to the Dashboard. If you are running LM-X version 4.0, please refer to [version 4.0 documentation](#) for information relevant to that version.*

The LM-X License Server Client Dashboard shows information about your LM-X license server, including:

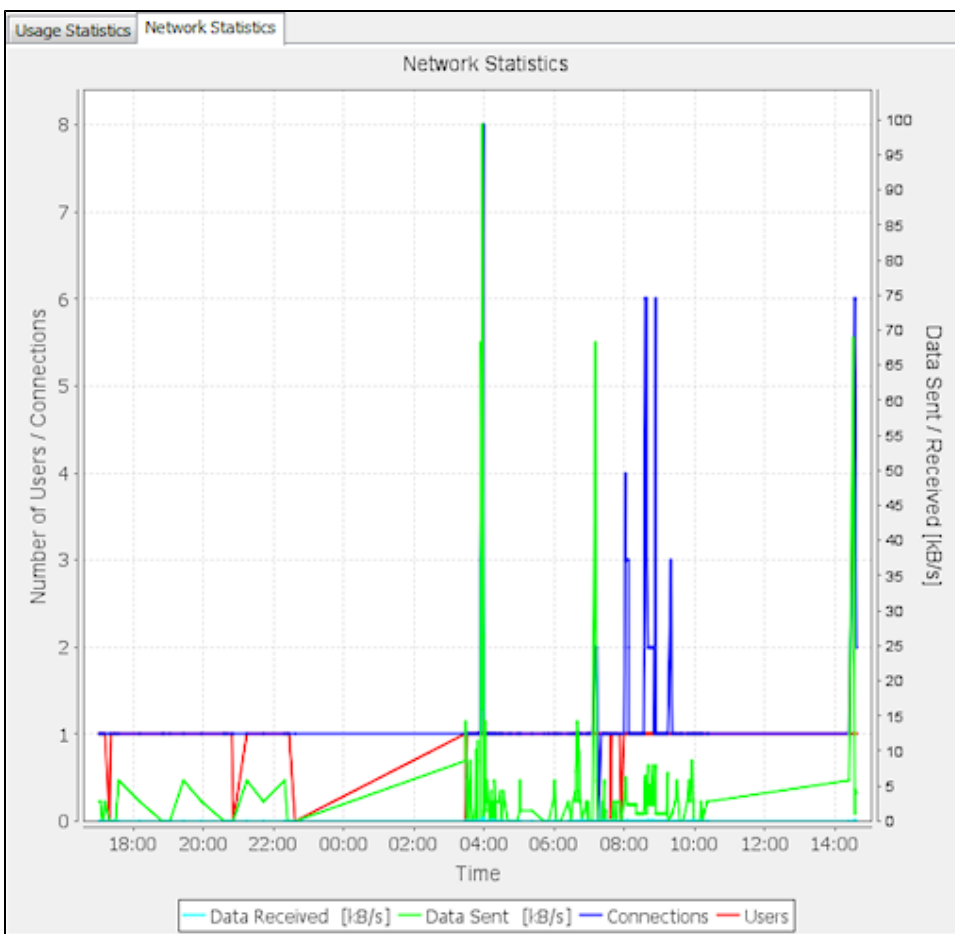
- General information, such as the hostname and port for the server, the LM-X version it is running, whether HAL and extended logging are enabled, etc.
- The IP addresses for the server.
- General statistics, including the number of users currently logged into the server and the number of features that are on the server.

In addition, the following graphs illustrate the license server data:

- The Usage Statistics graph shows usage and denials history for the past month for a selected feature on the server.
- The Network Statistics graph shows general statistics such as the server's number of users, connections, and data received/sent.

Using the right click menu in the graphs, you can:

- Modify chart properties, including settings for Title (renaming the chart and changing the appearance of the title text); Plot (renaming and changing the appearance of the axes and the lines in the plot, and setting the orientation of the plot); and Other (rendering options).
- Save the graph (currently limited to .PNG format).
- Print the graph.
- Zoom in/out on both axes or one axis (Domain, the horizontal line; or Range, the vertical line). For example, if you select Zoom Out > Domain Axis, the timeline on the graph will be less detailed and show a greater length of time. Zooming in shows greater detail over a shorter period of time.
- Zoom automatically to the extent of one or both axes of the graph. For example, the Network Statistics graph shown below has been zoomed automatically to both axes, so it shows all existing data (the data has been gathered for 20 hours and the upper limit of data sent/received is 100 kilobytes per second).



In addition to these shortcut menu zoom options, you can also click and drag to zoom to particular details in the graph.

## Restarting or shutting down the server

The LM-X License Server Client's Dashboard's License Server Administration area, located at the bottom of the Dashboard page, includes the ability to restart or shut down the license server. Restarting the license server normally takes a few seconds, but may take longer depending on the speed of the license server machine, number of licenses, and other factors.

## Viewing HostIDs

*The information on this page refers to LM-X version 4.1, which now allows saving HostID information to a text file. If you are running LM-X version 4.0, please refer to [version 4.0 documentation](#) for information relevant to that version.*

The HostIDs tab in the LM-X License Server Client lists all the HostIDs for the license server. The information for each HostID includes the Type (for example, ETHERNET, HOSTNAME, IPADDRESS, etc.), Description, and Value. The Description and Value are often identical, but some HostID types will have different entries. For example, an Ethernet HostID will have the Ethernet manufacturer under Description, and the actual Ethernet ID under Value.

You can use the Save button to save HostID information about the current machine the license server is running on to a text file. Saving this information to file gives you an easy way to supply your HostID information to your software vendor when requested.

# Viewing license usage statistics

*The information on this page refers to LM-X v5.0, which added the upgrade license type. If you are using an older version of LM-X, please refer to [version 4.1 documentation](#) for information relevant to that version.*

The License Usage tab in the LM-X License Server Client includes license usage statistics for features and users on the license server. Select **Users** or **Features** from the View options at the top right of the License Usage page to choose which statistics to see.

When you select **Feature** from the View options, the License Usage page includes the following information for each feature being served by the LM-X License Server:

- Feature name
- Software version number
- Software vendor name
- Key type (e.g., Exclusive, Additive, Upgrade, Token, etc.)
- Share code (e.g., Host, User, Virtual, etc.)
- License start and expire dates
- License type (e.g., Network)
- Total number of licenses for that feature on the license server
- Number of licenses in use
- Number of borrowed licenses

When you select **User** from the View options, the License Usage page lists the following statistics for each user on the LM-X License Server:

- User name
- User's machine hostname
- The IP address for the user's machine
- The features the user checked out
- The number of licenses the user has used
- The user's login and checkout time
- The state of the license (Borrowed, Checked Out, etc.)
- Borrow expiration

In addition to showing statistics for each user, you can use the Action column options to release all the licenses that are currently in use by a particular user.

The user statistics report may contain identical usernames and hosts under the following circumstances:

- If the user checked out multiple different features, each feature is displayed in a separate row.
- If multiple clients are started (for example, multiple instances of one application), you will see each client separately. (If you remove one client, the remaining clients will still remain valid.)



## Viewing the log file

The Log File tab in the [LM-X License Server Client](#) lets you see the LM-X License Server log file. You can filter results to show all entries, or limit the log to warnings or errors. You can search for specific entries in the log file using the "Search" box and the **Next** and **Previous** buttons. You can also use the **Download** button to download the log file.

# LM-X License Server

The LM-X License Server is a machine used to host network (floating, or shared) software licenses. Unlike node-locked licenses that are installed locally on individual users' machines, network licenses are able to be shared among many users. The LM-X License Server fulfills requests to run the network application if the requested license(s) are available. When the network license is released (for example, a user closes the application), the license is reclaimed by the license server and made available for other checkout requests.

When you start the LM-X License Server, it reads the associated configuration file (if one exists) to determine user settings such as the log file output path, whether certain users should be denied checkout of licenses, etc. (See [License server configuration file](#).) The following sections describe how to manage the LM-X License Server.

# Protocols

The license server runs over TCP protocol. LM-X supports IPv4/6 dual stack, which means that it is able to communicate in both IPv4 and IPv6 without having separate versions of the applications.

The license server also uses UDP protocol to allow applications to perform automatic server discovery on the network. Note: Automatic server discovery works only on local networks and will not work on WAN or VPN connections. Automatic server discovery is not guaranteed to work on all networks, particularly enterprise networks on which local firewalls or routers cut off UDP broadcast traffic.

When connecting client applications to IPv6 enabled servers, you must enclose the IP addresses in brackets [ ].

For example, to set environment variables for an IPv6 license server, you would enter:

LMX\_LICENSE\_PATH = @[1:2:3:4:5:6:7:8] or LMX\_LICENSE\_PATH = @[::1]

(For information about license file paths, see [Adding or removing license file paths.](#))

# License server configuration file

*The information on this page refers to LM-X v5.1.2 or newer, which added the ability to specify the duration for storing denial information. If you are using a previous version of LM-X, please see the documentation for [the previous version](#).*

The vendor may supply a license server configuration file, named `lmx-serv.cfg` by default. This configuration file is an ASCII text file, which can be opened and modified using any text editor. You may replace the existing information in the configuration file as needed.

For example, assume we are using a [floating license](#). This type of license allows any number of users to have the software installed, but only a certain number of users to use the software simultaneously. When all allotted licenses are in use, other users must wait until a license becomes available to use the software. If you want to use software licenses most efficiently and implement fair/desired distribution of licenses, then as an administrator of the LM-X server, you may set the following options in the LM-X License Server configuration file:

- **Permissions.** Setting permissions lets you allow or deny individual users/groups use of the license server according to your organization's specific needs. Permissions can be based on a set of rules that include permissions for normal checkouts as well as license borrowing.
- **Reservations.** You can reserve a specified number of licenses that can be used by individual users or groups. Reservations can also be done using a set of rules, allowing you to specify the reservation order. Some users or groups can be given higher priority than others.
- **Limitations.** You can limit the number of licenses that can be used by individual users or groups. Limitations can be done by a set of rules. In particular, limiting of users is done by a first match rule, so if a user belongs to more than one group specified in restrictions, the first restriction will apply to that user.

The configuration file includes instructions for using each setting in the file, which may include the following, depending on the options provided by your vendor. Some of the configuration settings can also be specified using the web-based UI, as described in [Administration](#).

Syntax	Description	Examples
<code>TCP_LISTEN_PORT = port number</code>	<p>The TCP port number the license server will listen on.</p> <ul style="list-style-type: none"> <li>• TCP port is used for data traffic protocol. The default TCP port is 6200.</li> <li>• UDP port is used for automatic server discovery protocol. The UDP port is fixed to 6200 and cannot be changed. See <a href="http://www.iana.org/assignments/port-numbers">http://www.iana.org/assignments/port-numbers</a> for more information.</li> </ul>	<code>TCP_LISTEN_PORT = 6200</code>
<code>TCP_BIND_ADDRESS = IP_address_1 / P_address_2</code>	<p>Limit which networks the license server allows for client connections.</p> <p>When this setting is specified, the license server will only accept clients that connect from a network that uses the specified IP addresses. You can specify only one address for each IP version (one for IPV4 and one for IPV6).</p> <p>This setting is useful when the license server is connected to more than one network (has more than one IP address) and you want to limit allowed connections based on which network the client is on.</p> <p>When this setting is unspecified, the license server accepts clients from all available networks.</p>	<code>TCP_BIND_ADDRESS = 192.168.21.321 8000:8000:8000:abcd:1234:12df:fd54</code>

<p>HAL_SERVERserver_number = [port]@hostname or HAL_SERVERserver_number = [port]@IP_address</p> <p><b>Note:</b> Port is optional.</p>	<p>High Availability Licensing (HAL) servers, which enable redundant servers, so if one server goes down, two others will still work. HAL consists of 3 specified servers, at least 2 of which must be up and running at all times.</p> <p>Each HAL_SERVER line indicates a license server that has HAL enabled by its license(s). Each HAL server has a specific role, and should be specified in terms of how many resources each server has:</p> <ul style="list-style-type: none"> <li>• HAL_SERVER1 is your master server, which allows both CHECKOUT and BORROW. HAL_SERVER1 should be your most powerful server.</li> <li>• HAL_SERVER2 is your first slave server, which allows CHECKOUT but denies BORROW in the event that your master server goes down. HAL_SERVER2 should be your second most powerful server.</li> <li>• HAL_SERVER3 is part of your configuration to ensure that everything works as expected, and does not allow any CHECKOUT or BORROW requests. HAL_SERVER3 should be your least powerful server.</li> </ul> <p><b>Important:</b> The HAL_SERVER list must be identical on all your servers for HAL to function properly</p>	<p>HAL_SERVER1 = 6200@server1 HAL_SERVER2 = 6200@server2 HAL_SERVER3 = 6200@server3</p>
LOG_FILE = path	<p>The log file path. Specifying the full path is preferred.</p> <p>If you do not specify this setting, the default is used: On Windows the default is lmx-serv.log, under the license server directory. On Unix, the default location for the log file is in the directory from which the license server was started</p>	<p>LOG_FILE = c:\program files\lmx-server.log LOG_FILE = /home/user1/lmx-serv.log</p>
LOG_FORMAT = NORMAL or EXTENDED	<p>The format for the log file.</p> <p>The default setting for the log file format is NORMAL.</p> <p>Specifying EXTENDED causes additional information to be included in the log file, such as license server HostIDs, whether the license server is a virtual machine, etc. Setting the log file format to EXTENDED is particularly useful for debugging purposes.</p>	<p>LOG_FORMAT= NORMAL LOG_FORMAT=EXTENDED</p>
LOG_EXCLUDE = message1, message2, etc.	<p>Exclude messages from the log. The following messages can be excluded: CHECKOUT, CHECKIN, STATUS, BORROW, BORROW_RETURN, REMOVE_USER, REMOVE_RESTART or REMOTE_SHUTDOWN.</p>	<p>LOG_EXCLUDE = CHECKOUT, CHECKIN, STATUS</p>
LOGFILE_ROTATE_INTERVAL = rotation_interval	<p>The interval for log file rotation.</p> <p>The value may be set to "day" (every day at midnight), "week" (every Monday at midnight), or "month" (the first day of every month at midnight).</p> <p>After rotation, the old log file will be named filename.log.rotation_date in the format yyyy-mm-dd.</p> <p>A message indicating the location of the rotated log file is added to the end of the old log file and the beginning of the new log file, as follows:</p> <p>"Log file was rotated and saved to filename."</p>	<p>LOGFILE_ROTATE_INTERVAL = day</p>
MIN_USER_REMOVE_TIME = time in seconds	<p>Minimum time, in seconds, that must elapse from the connection before a user can be removed using lmxendutil.</p> <p>The specified time must be equal to or greater than the number of seconds specified by your application vendor. Default minimum time is 120 seconds. If the time is set to -1, user removals will not be allowed.</p>	<p>MIN_USER_REMOVE_TIME = 120</p>

<code>LICENSE_FILE = path</code>	<p>The license file path.</p> <p>On Windows: If no file is set, the license server will look for <i>vendor.lic</i> in the same directory as the license server.</p> <p>On Unix: If no file is set, the license server will look for <i>/usr/x-formation/vendor.lic</i>.</p> <p>In both cases, the filenames must be lowercase. You can specify one or multiple paths as needed.</p>	<pre>LICENSE_FILE = d:\server\network.lic LICENSE_FILE = c:\extra_file.lic LICENSE_FILE = /home/user1/floating_license.lic LICENSE_FILE = /home/user1/floating_license2.lic</pre>
<code>USAGE_DATABASE = database path</code>	<p>Pay-per-use usage database (used for billing purposes). See <a href="#">Pay Per Use feature</a> for EXTENDED information, including database format and an example of data printout.</p>	<pre>USAGE_DATABASE = d:\server\usage.db USAGE_DATABASE = /home/user1/usage.db</pre>
<code>USAGE_LEVEL= detail level</code>	<p>Specify pay-per-use detail level.</p> <ul style="list-style-type: none"> <li>STANDARD includes basic usage information.</li> <li>EXTENDED includes user information in addition to the basic usage information.</li> </ul>	<code>USAGE_LEVEL = STANDARD</code>
<code>USAGE_WRITE_INTERVAL= number of actions</code>	<p>Specify the number of pay-per-use <a href="#">actions</a> (checkouts, checkins, etc.) after which pay-per-use records will be written to the pay-per-use database file. The default setting is 1000.</p>	<code>USAGE_WRITE_INTERVAL = 1000</code>
<code>REMOTE_ACCESS_PASSWORD = password</code>	<p>Remote administration password (used when remotely stopping and restarting the license server and removing users from it).</p> <p>The password is case-sensitive.</p>	<code>REMOTE_ACCESS_PASSWORD = MyPassword123</code>
<code>FAST_QUEUE = feature1, feature2, etc.</code> or <code>FAST_QUEUE = ALL</code>	<p>Fast queuing allows requests that can be fulfilled immediately to be fulfilled.</p> <p>For example, if a client is waiting for two licenses, and only one license is immediately available, another client that needs only one license can bypass the queue and take the single license without waiting.</p> <p>Default behavior of license queuing is to put the client at the end of the queue regardless whether the license request could be satisfied.</p>	<code>FAST_QUEUE = f2, d5, app2</code>

`ALLOW_IPADDR_ALL` = one or more IP addresses

`ALLOW_IPADDR_feature name` = one or more IP addresses (must be either specific A.B.C.D or with wildcards; e.g., A.B.\*)

`DENY_IPADDR_ALL` = one or more IP addresses

`DENY_IPADDR_feature name` = one or more IP addresses (must be either specific A.B.C.D or with wildcards; e.g., A.B.\*)

`ALLOW_HOST_ALL` = one or more hosts

`ALLOW_HOST_feature name` = one or more hosts

`DENY_HOST_ALL` = one or more hosts

`DENY_HOST_feature name` = one or more hosts

`ALLOW_USER_ALL` = one or more users

`ALLOW_USER_feature name` = one or more users

`DENY_USER_ALL` = one or more users

`DENY_USER_feature name` = one or more users

**Note:** For host, you can use a hostname or use "localhost" to specify the current machine. For IP address, you can specify a complete address (A.B.C.D) or use wildcards; e.g., A.B.\*).

Allow/deny specific clients from using the license server.

The allow/deny rules work as follows:

- Rules are attempted to be matched in the order they are written.
- If no rule matches the specific client, then that client is allowed.

The following example will deny all clients except that with hostname 'trusted'. This applies to all features.

```
ALLOW_HOST_ALL = trusted
DENY_IPADDR_ALL = *.*.*
```

The following example will allow clients on only 2 subnets, user Administrator and root from any host and deny everyone else. This applies to all features.

```
ALLOW_IPADDR_ALL = 192.168.1.* 192.168.2.*
ALLOW_USER_ALL = Administrator root
DENY_IPADDR_ALL = *.*.*
```

The following example will deny clients on localhost, deny the machines with hostname 'untrusted' and 'crackerjack', allow clients on the internal network, and deny everyone else. This applies to the feature f2.

```
DENY_HOST_f2 = localhost untrusted
crackerjack
ALLOW_IPADDR_f2 = 192.168.*.*
DENY_IPADDR_f2 = *.*.*
```

<p>ALLOW_BORROW_IPADDR_ALL = one or more hosts</p> <p>ALLOW_BORROW_IPADDR_feature name = one or more hosts</p> <p>DENY_BORROW_IPADDR_ALL = one or more hosts</p> <p>DENY_BORROW_IPADDR_feature name = one or more hosts</p> <p>ALLOW_BORROW_HOST_ALL = one or more hosts</p> <p>ALLOW_BORROW_HOST_feature name = one or more hosts</p> <p>DENY_BORROW_HOST_ALL = one or more hosts</p> <p>DENY_BORROW_HOST_feature name = one or more hosts</p> <p>ALLOW_BORROW_USER_ALL = one or more users</p> <p>ALLOW_BORROW_USER_feature name = one or more users</p> <p>DENY_BORROW_USER_ALL = one or more users</p> <p>DENY_BORROW_USER_feature name = one or more users</p> <p><b>Note:</b> For host, you can use a hostname or use "localhost" to specify the current machine. For IP address, you can specify a complete address (A.B.C.D) or use wildcards; e.g., A.B.C.*).</p>	<p>Allow/deny specific clients from borrowing licenses.</p>	<p>The following example will allow the specific users, and deny host and IP addresses on the list from borrowing any feature. Everyone else will be allowed.</p> <p>ALLOW_BORROW_USER_ALL = daisy harry tom DENY_BORROW_HOST_ALL = server1 machine5 DENY_BORROW_IPADDR_ALL = 192.168.3.* 192.168.4.*</p> <p>The following example will allow the specific users and deny everyone else from borrowing f2.</p> <p>ALLOW_BORROW_USER_f2 = lazyjack rabbit joeuser DENY_BORROW_IPADDR_f2 = *.*.*</p>
<p>LIMIT_USER_feature name_limit count = one or more users</p> <p>LIMIT_HOST_feature name_limit count = one or more hosts</p> <p>LIMIT_IPADDR_feature name_limit count = one or more hosts</p> <p><b>Note:</b> For host, you can use a hostname or use "localhost" to specify the current machine. For IP address, you can specify a complete address (A.B.C.D) or use wildcards; e.g., A.B.C.*).</p>	<p>Limit the number of licenses that can be used by individual users or groups to implement fair/desired distribution of licenses.</p> <p>Limiting of users is done by a first match rule, so if a user belongs to more than one group specified in restrictions, the first restriction will apply to that user.</p>	<p>LIMIT_USER_f2_5 = harry joe sam LIMIT_IPADDR_f3_3 = 192.168.2.* 192.168.4.*</p>
<p>RESERVE_USER_feature name_reserve count = one or more users</p> <p>RESERVE_HOST_feature name_reserve count = one or more hosts</p> <p>RESERVE_IPADDR_feature name_reserve count = one or more hosts</p> <p><b>Note:</b> For host, you can use a hostname or use "localhost" to specify the current machine. For IP address, you can specify a complete address (A.B.C.D) or use wildcards; e.g., A.B.C.*).</p>	<p>Reserve a number of licenses that can be used by individual users or groups to implement fair/desired distribution of licenses.</p> <p>Reservation of users is done by a first match rule, so if a user belongs to more than one group specified in the rules, the first rule will apply to that user.</p>	<p>RESERVE_USER_f2_5 = harry joe sam RESERVE_IPADDR_f3_3 = 192.168.2.* 192.168.4.*</p>



<p>BORROW_LIMIT_COUNT_ALL = <i>limit count</i> BORROW_LIMIT_COUNT_<i>feature name</i> = <i>limit count</i></p> <p>IGNORE_BORROW_LIMIT_COUNT_USER_<i>feature name</i> = <i>one or more users</i></p> <p>IGNORE_BORROW_LIMIT_COUNT_HOST_<i>feature name</i> = <i>one or more hosts</i></p> <p>IGNORE_BORROW_LIMIT_COUNT_IPADDR_<i>feature name</i> = <i>one or more IP addresses</i></p> <p><b>Note:</b> You can use IGNORE_BORROW_LIMIT_* flag to whitelist small and specific predicates blacklisted by broader BORROW_LIMIT_* predicate.</p>	<p>Limit/Do not limit the number of licenses that can be borrowed to prevent all licenses from being borrowed at the same time.</p>	<p>BORROW_LIMIT_COUNT_f2 = 1 BORROW_LIMIT_COUNT_ABCDEF = 5</p> <p>The following example will allow the user ADMIN, and deny host and IP addresses on the list from borrowing more than 100 features.</p> <p>BORROW_LIMIT_COUNT_F1 = 100</p> <p>IGNORE_BORROW_LIMIT_COUNT_USER_F1 = admin</p> <p>IGNORE_BORROW_LIMIT_COUNT_HOST_F1 = server</p> <p>IGNORE_BORROW_LIMIT_COUNT_IPADDR_F1 = 192.168.1.*</p>
<p>BORROW_LIMIT_HOURS_ALL = <i>limit hours</i> BORROW_LIMIT_HOURS_<i>feature name</i> = <i>limit hours</i></p> <p>IGNORE_BORROW_LIMIT_HOURS_USER_<i>feature name</i> = <i>one or more users</i></p> <p>IGNORE_BORROW_LIMIT_HOURS_HOST_<i>feature name</i> = <i>one or more hosts</i></p> <p>IGNORE_BORROW_LIMIT_HOURS_IPADDR_<i>feature name</i> = <i>one or more IP addresses</i></p> <p><b>Note:</b> You can use IGNORE_BORROW_LIMIT_* flag to whitelist small and specific predicates blacklisted by broader BORROW_LIMIT_* predicate.</p>	<p>Limit/Do not limit the number of hours licenses can be borrowed to prevent licenses from being borrowed for too long.</p>	<p>The following example will allow the specific users, and deny host and IP addresses on the list from borrowing any feature. Everyone else will be allowed.</p> <p>BORROW_LIMIT_HOURS_f2 = 1 BORROW_LIMIT_HOURS_ABCDEF = 5</p> <p>The following example will allow the user admin, and deny host and IP addresses on the list from borrowing more than 100 features.</p> <p>BORROW_LIMIT_HOURS_F1 = 10</p> <p>IGNORE_BORROW_LIMIT_HOURS_USER_F1 = admin</p> <p>IGNORE_BORROW_LIMIT_HOURS_HOST_F1 = server</p> <p>IGNORE_BORROW_LIMIT_HOURS_IPADDR_F1 = 192.168.1.*</p>
<p>FEATURE <i>featurename</i> {   <i>feature settings</i> }</p>	<p>Specify licenses directly within the configuration file to eliminate the need to have both a license file and configuration file for the license server. You can specify any features from one or more license files.</p> <p>The content must be specified within the <code>__START_LICENSE__</code> and <code>__END_LICENSE__</code> clauses.</p>	<p><code>__START_LICENSE__</code> FEATURE F1 {   VENDOR = XYZ } <code>__END_LICENSE__</code></p>
<p>GROUP_<i>name</i> = <i>member1 member2</i></p>	<p>Specify a group name and the group members to which you want to apply restrictions, limitations and reservations. Creating groups can make these features easier to use and help you to avoid/remove redundancies from the configuration file.</p> <p>You can create groups that contain users, host names and IP addresses. Groups can contain any other group, and there is no limit on the number of members that can be included in a group.</p> <p>The names of groups and group members are case-insensitive.</p>	<p>The following example creates a group named "hr" with one member, "anna," and another group, "employees," which contains three individual users (joe, mary, and sam), plus includes the group "hr" as a sub-group.</p> <p>GROUP_hr = anna GROUP_employees = joe mary sam hr</p> <p>After creating groups, you can apply permissions, reservations, and limitations described above to those groups in the same way you would individual users; for example:</p> <p>DENY_USER_f1 = employees RESERVE_USER_f2_2 = employees LIMIT_USER_f3_2 = hr</p>

DENIAL_STORE_PERIOD = <i>time_in_se</i> <i>conds</i>	<p>Specify how long the license server will keep denial information, in seconds.</p> <ul style="list-style-type: none"><li>• Valid values are between one minute and one year, expressed in seconds.</li><li>• The value is numeric only; do not use commas or periods in the entered value.</li><li>• The default duration is one day (86400 seconds).</li></ul>	<p>The following example sets the denial storage period to 1 day:</p> <p>DENIAL_STORE_PERIOD = 86400</p>
---	---	--

## lmx\_server\_conf.c file

A file named `lmx_server_conf.c` contains various flags and callbacks which can be used to extend or modify the behavior of the license server. For each callback you can write a function which meets the specified prototype. We recommend that you write your own enhancements in a new file and modify the makefile to include your enhancements during compilation of the license server.

Note that the settings specified using `lmx_server_conf.c` cannot be changed after compilation of the license server.

Some of the callback functions that can be specified using `lmx_server_conf.c` include the following:

Callback function	Description	Parameters
LMX_ServerShutdown	Callback for cleaning up user code when license server goes down.	SERVER_SHUTDOWN_MODE
	Callback for restarting the license server.	SERVER_RESTART_MODE

# License server log file

The license server can produce a log file that details activity such as client connections or disconnections, license checkout/checkin, and other server activity. For Unix operating systems, the log file may also contain [exit signals](#).

In the configuration file, you can control the following settings for the license server log:

- Specify normal or extended logging. When using extended logging note that:
  - Extended logging results in greater detail in the log file.
  - Extended logs can be [imported into License Statistics](#) to obtain denied request statistics.
- Specify the interval for log file rotation. Generally, data written to the log file is useful only for a limited time, so log rotation is recommended for removing old log data and reducing the storage requirements of the log file.
- Specify the desired output location for the log.

See [License server configuration file](#) for more details about the above log settings. You can also control these log settings using the web-based UI, as described in [Enabling and configuring license server logging](#).

Over time, the log file can grow to a substantial size depending on licensing activity, so it is best to write the log to a local file system rather than across a network.

If the log file is deleted, the license server will create a new log file on the next write.

# Running the license server from a command line

*The information on this page refers to v4.4.3 and later, which introduced an installation program that installs both LM-X License Server and the end-user tools and removed some commands from lmx-serv. For applicable to earlier versions, see [documentation for previous versions](#).*

The lmx-serv command will let you run the license server as a service in Windows or as a daemon in the background on Unix. However, it is recommended that you [use the provided installer](#) to install and start the license server instead of using lmx-serv.

The lmx-serv command usage is as follows.

## For Windows:

Command	Options
lmx-serv	<code>[-config <i>configfile</i> -licpath <i>licensefile</i> -logfile <i>logfile</i> -port <i>portnumber</i>]</code>

## For Unix:

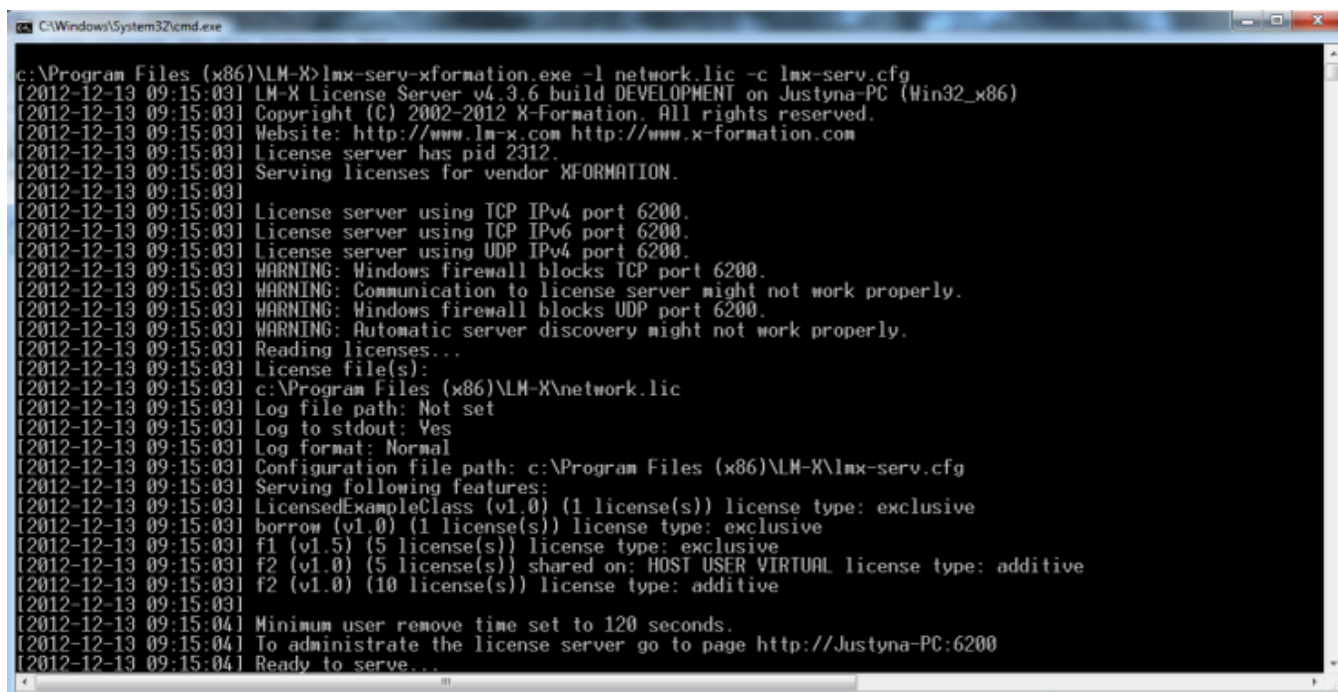
Command	Options
lmx-serv	<code>[-background -config <i>configfile</i> -licpath <i>licensefile</i> -logfile <i>logfile</i> -port <i>portnumber</i>]</code>

Where:

Command			Description
Long version	Short version	Applies to	
-background	-b	Unix	Run the license server as a daemon in the background.
-config	-c	All	Specify an optional path to an lmx-serv.cfg configuration file. Typing the full path is required. If the server is run without the -c parameter, it will use default settings.
-licpath	-l	All	Specify an optional license file path that will be read in addition to those specified within the lmx-serv.cfg configuration file. Alternatively, you may specify a directory in which the license server will look for all .lic files. You can specify multiple paths, separated by a semicolon ( ; ) for Windows or a colon ( : ) for Unix; for example, "-l C:\dir1;c:\dir2."  If no default license is defined in the configuration file and the -l parameter is not specified (or no license can be found in given location), the server will look for all .lic files in its directory.
-logfile	-lf	All	Specify an optional logfile path, which will override any logfile settings in the lmx-serv.cfg configuration file.
-port	-p	All	Specify an optional port number, which will override the port number set in the configuration file.
-help	-h	All	Print out usage information for these commands.

We recommend enclosing all switches (e.g., configuration file path) within double quotes ( " ") to avoid problems with white spaces.

The following example shows running the license server on Windows from a command line.



```

C:\Windows\System32\cmd.exe
c:\Program Files (x86)\LM-X>lmx-serv-xformation.exe -l network.lic -c lmx-serv.cfg
[2012-12-13 09:15:03] LM-X License Server v4.3.6 build DEVELOPMENT on Justyna-PC (Win32_x86)
[2012-12-13 09:15:03] Copyright (C) 2002-2012 X-Formation. All rights reserved.
[2012-12-13 09:15:03] Website: http://www.lm-x.com http://www.x-formation.com
[2012-12-13 09:15:03] License server has pid 2312.
[2012-12-13 09:15:03] Serving licenses for vendor XFORMATION.
[2012-12-13 09:15:03]
[2012-12-13 09:15:03] License server using TCP IPv4 port 6200.
[2012-12-13 09:15:03] License server using TCP IPv6 port 6200.
[2012-12-13 09:15:03] License server using UDP IPv4 port 6200.
[2012-12-13 09:15:03] WARNING: Windows firewall blocks TCP port 6200.
[2012-12-13 09:15:03] WARNING: Communication to license server might not work properly.
[2012-12-13 09:15:03] WARNING: Windows firewall blocks UDP port 6200.
[2012-12-13 09:15:03] WARNING: Automatic server discovery might not work properly.
[2012-12-13 09:15:03] Reading licenses...
[2012-12-13 09:15:03] License file(s):
[2012-12-13 09:15:03] c:\Program Files (x86)\LM-X\network.lic
[2012-12-13 09:15:03] Log file path: Not set
[2012-12-13 09:15:03] Log to stdout: Yes
[2012-12-13 09:15:03] Log format: Normal
[2012-12-13 09:15:03] Configuration file path: c:\Program Files (x86)\LM-X\lmx-serv.cfg
[2012-12-13 09:15:03] Serving following features:
[2012-12-13 09:15:03] LicensedExampleClass (v1.0) (1 license(s)) license type: exclusive
[2012-12-13 09:15:03] borrow (v1.0) (1 license(s)) license type: exclusive
[2012-12-13 09:15:03] f1 (v1.5) (5 license(s)) license type: exclusive
[2012-12-13 09:15:03] f2 (v1.0) (5 license(s)) shared on: HOST USER VIRTUAL license type: additive
[2012-12-13 09:15:03] f2 (v1.0) (10 license(s)) license type: additive
[2012-12-13 09:15:03]
[2012-12-13 09:15:04] Minimum user remove time set to 120 seconds.
[2012-12-13 09:15:04] To administrate the license server go to page http://Justyna-PC:6200
[2012-12-13 09:15:04] Ready to serve...

```

## Unix exit signals

When running LM-X on a Unix operating system, you may see an "exit signal" message in the license server log file. Exit signals are not LM-X errors, but rather events sent from the operating system to instruct an application to shut down in various ways. For example, the following excerpt from a Linux license server log file contains exit signal 15, which is the signal SIGTERM (termination).

```
[2015-02-08 15:48:15] WARNING: Unable to establish connection with  
HAL peer 6200@193.28.180.39. Please make sure the host is up!  
[2015-02-08 15:52:35] Shutting down due to exit signal 15
```

You can find information about exit signals at [http://www.comptechdoc.org/os/linux/programming/linux\\_pgsignals.html](http://www.comptechdoc.org/os/linux/programming/linux_pgsignals.html).

# Upgrading the license server

You must upgrade your LM-X License Server every time your software vendor upgrades to a new LM-X release.

To upgrade the server at your site:

1. Uninstall and remove the old [LM-X License Server](#).
2. Install the new license server obtained from your software vendor.

After upgrading LM-X to a newer version, your existing application and existing license files should continue to work with the new version, as described in [Version compatibility](#).



## Optional features

The following sections describe LM-X License Server features that you and/or your vendor may optionally enable, including [High Availability Licensing](#), [Pay Per Use feature](#), [license borrowing](#), [automatic server discovery](#), and [license queuing](#).

# Configuring LM-X License Manager for high network connection rates

By default, Windows can handle a maximum of 5000 open TCP connections. If your loads are higher, you may experience problems such as failed requests for new connections and poor performance.

To improve Windows' ability to handle the connections, change the registry as described below.

**(Important: Changes to the registry can cause serious problems, including inability to reboot your machine.** You may want to make a backup of the registry before continuing. For information on backing up the registry, visit [support.microsoft.com](http://support.microsoft.com) and type "registry editor" into the "Search" box. Instructions for working with the registry editor vary depending on your Windows version.)

1. From the Windows Start menu, type **Run** and press **Enter**.
2. From the Run dialog, type **regedit** and click **OK** to start the registry editor.
3. In the Registry Editor, open HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services.
4. For IPV4, open Tcpip, or for IPV6, open Tcpip6.
5. Select the registry subkey **Parameters**, and check the right panel for the registry entries MaxUserPort and TcpTimedWaitDelay.

If these entries do not exist, create them:

- a. Right-click on Parameters.
  - b. From the shortcut menu, select **New**, and then select **DWORD (32-bit) value**. A new registry entry, New Value #1, is added to the right side of the Registry Editor.
  - c. Change the name of the new entry to MaxUserPort.
  - d. Repeat steps a through c, changing the name of the second new entry to TcpTimedWaitDelay.
6. Change the value of the MaxUserPort and TcpTimedWaitDelay entries:
    - a. Select the entry.
    - b. Press **Enter**, or right-click on the entry and select **Modify**. This will open the Edit DWORD Value dialog.
    - c. Change the Value setting for MaxUserPort to 65534, and the Value setting for TcpTimedWaitDelay to 30.

# High Availability Licensing

High Availability Licensing (HAL) is an important feature for applications that require high availability. Activating HAL introduces fault tolerance, because the licensed applications no longer depend on a single point of failure.

## How High Availability Licensing works

HAL uses three license servers, each assigned a specific role. The first license server is the primary server and allows clients to both checkout and borrow licenses. The second license server can allow clients to checkout licenses only, in the event the first license server is down. The third license server denies all requests, but is required as part of the configuration to ensure high availability. To use HAL, your license must be HAL-enabled by your vendor. See [Enabling and configuring HAL](#) for information about using the web-based UI to set up HAL license servers.

HAL requires a stable network connection between the servers. Too many network problems will make the system unstable and license checkouts unreliable.

Note that a license server can serve either a HAL-enabled license or normal network license, not both. You should also note that HAL license servers do not communicate with one another except for information about which one is up and being active or passive. Information about other states, such as [borrowing a license](#) or [queuing licenses](#), is not exchanged between HAL license servers.

# How to install HAL license servers

The following steps describe how to easily install HAL (High Availability License) license servers on Windows.

1. Enable and configure HAL using the [LM-X License Server Client](#) (LicserverClient.jar).

HAL requires a set of 3 license servers. Each HAL license server has a fixed role, as described in the following table:

HAL License Server Number	Role
1	This HAL license server can allow clients to both checkout and borrow licenses, exactly like a normal license server.
2	In the event that HAL license server #1 is down, this server can allow clients to checkout licenses, but will deny borrow requests.
3	This HAL license server will deny any requests, but is required as a part of the configuration to ensure high availability.

You must specify the three servers in the license server configuration file for *each of the three servers* that will be used in your HAL configuration. To ensure the configuration files are identical, you may wish to edit one configuration file and copy the file to the other two servers. (The servers may have different settings throughout the configuration file as needed, but the HAL settings must be identical.)

2. Open the log file to verify the HAL license servers are started and working normally, indicated by the line "Ready to serve..." as shown in the following example. (This example is the log file for a master server; that is, the server specified as HAL\_SERVER1 in the configuration file.)

```
[2016-01-21 16:13:04] License server using TCP IPv4 port 6200.
[2016-01-21 16:13:04] License server using UDP IPv4 port 6200.
[2016-01-21 16:13:04] Logfile path: lmx-server.log
[2016-01-21 16:13:04] License file(s):
[2016-01-21 16:13:04] C:\LM-X\formation.lic
[2016-01-21 16:13:04] Serving following features:
[2016-01-21 16:13:04] f1 (v1.5) (5 licenses) license type: exclusive
[2016-01-21 16:13:04]
[2016-01-21 16:13:04] Minimum user remove time set to 120 seconds.
[2016-01-21 16:13:04] Ready to serve...
[2016-01-21 16:13:04] HAL: This license server is configured as a HAL MASTER.
[2016-01-21 16:13:04] HAL: Peer server: my_#2_server:6200
[2016-01-21 16:13:04] HAL: Peer server: my_#3_server:6200
[2016-01-21 16:13:04] HAL: CHECKOUT requests on this server are NOT ALLOWED!
[2016-01-21 16:13:04] HAL: BORROW requests on this server are NOT ALLOWED!
[2016-01-21 16:13:04] HAL: Connection with HAL peer my_#2_server:6200 is up!
[2016-01-21 16:13:04] HAL: Connection with HAL peer my_#3_server:6200 is up!
[2016-01-21 16:13:19] HAL: CHECKOUT requests on this server are ALLOWED!
[2016-01-21 16:13:19] HAL: BORROW requests on this server are ALLOWED!
```

Note that the log file for the servers specified as HAL\_SERVER1 and HAL\_SERVER2 in the configuration file (the servers that allow requests) may initially indicate that the server is not allowing requests. However, within 30 seconds, when the connection between the servers is detected, the log file will report that requests are allowed.

For more information, please refer to [Enabling and configuring HAL](#).

As for normal license servers, the log file will indicate the cause of any problems running the HAL servers. Note: you must disable or configure your firewall on each HAL server for HAL to function properly.

## Pay Per Use feature

The Pay Per Use feature captures license usage in a pay-per-use database which can then be sent to the vendor upon request. The database is authenticated, and its contents can be validated by the vendor.

For example, if you have an unrestricted use license that is paid for based on monthly usage, your vendor can request that you write the usage information to a pay-per-use database each month, and send the database file to them for billing purposes.

Usage databases use SQLite, a free open source database. You can find out more about SQLite at <http://www.sqlite.org>.

## Enabling and using usage databases

*The information on this page refers to LM-X v4.4 or newer, which added the `USAGE_WRITE_INTERVAL` setting. This setting is not available in previous versions of LM-X.*

The vendor enables Pay Per Use by editing the `USAGE_DATABASE` entry (disabled by default) in the license server configuration file. The configuration file contains a description and examples for using the `USAGE_DATABASE` setting. (For information on configuring Pay Per Use, see [License server configuration file](#) and [Enabling and configuring license server logging](#).)

You can edit the `USAGE_LEVEL` setting in the configuration file to specify whether to generate a STANDARD report that does not include user information, or an EXTENDED report that includes the username, hostname and IP address for each checkout and checkin request. The `USAGE_LEVEL` setting is set to STANDARD in the configuration file by default. In addition, you can set the `USAGE_WRITE_INTERVAL` option in the configuration to specify the number of pay-per-use [actions](#) (checkouts, checkins, etc.) that should occur before pay-per-use records will be written to the usage database file. Note that records will immediately be written to the database file when the server is shut down.

The usage database can be verified and printed to screen using the `lmxendutil -readusagedb` command. (See [LM-X End-user utility](#) for information about using this command.) This command reads the database, performs a verification, and prints the usage table content to the screen. You can review the content to ensure its validity by verifying the ID ranges; however, never attempt to manipulate the database.

## Database structure

The usage database contains several tables; however, the only table that may be accessed is named usage. The other tables are for authentication purposes only.

The contents of the usage table are described below.

Table name	Columns	Description
usage	(id integer primary key, time integer, action text, comment text, auth text)	This table contains usage information. Each row in the table contains a timestamp, an action that occurred, comments on additional information if applicable, and authentication information.

The time format used in the time integer column is "time\_t," which you can find out more about by going to: [http://en.wikipedia.org/wiki/Time\\_t](http://en.wikipedia.org/wiki/Time_t).

The action text column will contain one of the following:

Action	Description
STARTUP	Indicates the license server started up. The comment text column is empty.
SHUTDOWN	Indicates the license server was shut down. The comment text column is empty.
CHECKOUT	Indicates a checkout was performed. The comment text column contains additional information related to the checkout, including the feature name, a unique id of the specific feature (in case there are multiple licenses on the server of the same feature name), and a count. For example, (FEATURE: UNLIMITED UNIQUEID: 5E7100EE3F98F68F559AB09D57AA8BA154F77907A48D2487 COUNT: 1 SOFTLIMIT_EXCEEDED: FALSE).  SOFTLIMIT_EXCEEDED indicates whether the number of licenses specified for softlimit licensing has been exceeded.
CHECKIN	Indicates a checkin was performed. The comment text column contains additional information related to the checkin, in the same manner as for CHECKOUT.
USAGE	Specifies the current usage of a particular feature. This is the sum of all checkouts, checkins, and borrows that have occurred. The comment text column contains additional information related to the usage, in the same manner as for CHECKOUT. For example, (FEATURE: UNLIMITED UNIQUEID: 1ECA6DB5DDEAA15E565AE7C78AFBB6F67DC9F35593CCBFB0 COUNT: 2 SOFTLIMIT_EXCEEDED: FALSE USERNAME: Administrator HOSTNAME: amilo IPADDRESS: ::1)  The above example shows the syntax for USAGE_LEVEL set to EXTENDED, which includes USERNAME, HOSTNAME, and IPADDRESS (see <a href="#">Enabling and using usage databases</a> ).

## Usage table example

The following is a partial example of a usage table printout.

```
Timestamp: 2009-01-30 07:37:06 Action: CHECKOUT Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 3
Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 402
Timestamp: 2009-01-30 07:37:06 Action: CHECKOUT Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 3
Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 405
Timestamp: 2009-01-30 07:37:06 Action: CHECKIN Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 32
Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 373
.
.
.
Timestamp: 2009-01-30 07:37:08 Action: CHECKIN Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 61
Timestamp: 2009-01-30 07:37:08 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 0
Timestamp: 2009-01-30 07:37:08 Action: CHECKOUT Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 500
Timestamp: 2009-01-30 07:37:08 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 500
Timestamp: 2009-01-30 07:37:08 Action: CHECKIN Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 500
Timestamp: 2009-01-30 07:37:08 Action: USAGE Comment: FEATURE: TEST_MULTIUSER
UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 0
Timestamp: 2009-01-30 07:37:08 Action: SHUTDOWN Comment:
```



## Borrowing a license

LM-X License Manager supports license borrowing, which lets you use a network license without being connected to the license server. Borrowing is commonly used in cases such as taking a laptop computer home for the weekend or for traveling off-site, giving you access to the software when a connection to the network is not possible.

Your software vendor must allow license borrowing for it to be available for your use. In some cases, the vendor may supply an interface for borrowing or returning licenses. Otherwise, you can borrow and do an early return of licenses by setting an LM-X environment variable specifically for this purpose.

Use the following steps to borrow a license using an LM-X environment variable. (Instructions given for accessing environment variable settings are for Windows 7. Please see your OS documentation or your system administrator for instructions on editing environment variables for your specific OS.)

1. Open the Windows Control Panel and select **System and Security**.
2. Select **System** from the System and Security options.
3. Select **Advanced system settings** from the list of options in the left column of the System window.
4. From the System Properties dialog that appears, select **Environment Variables...**
5. Under System variables, select **New...**
6. For the Variable name, enter LMX\_BORROW or vendor\_BORROW (where vendor is the name of the software vendor for the software to be borrowed). Set the value to the number of hours for which you want to borrow the license. This number cannot be greater than the number of hours the software vendor allows.
7. Click **OK** to add the variable.
8. Click **OK** from the Environment Variables dialog to save your changes, and then click **OK** from the System Properties dialog.  
If you're running the application, be sure to restart it after setting the environment variable change to ensure that the settings are picked up. The license will be borrowed the next time you checkout the license and will be available for the specified duration, after which the application will no longer be available without connecting to the license server.
9. If you want to do an early return of the license at any time before its expiration, repeat the steps above to access the environment variables, then select LMX\_BORROW from the list of environment variables, and click **Edit**.
10. Change the value to -1 (or any negative value). This will return the license so it's once again available for other users to check out.

**Note:** You can query the server to see whether a license is borrowed at any time using the LM-X End-user Configuration Tool or LM-X End-user utility, lmxendutil.

In very rare cases, the client may not receive a borrowed license due to network problems that occur at the exact time a license is being borrowed. Because borrowing is not transaction based, the server may see the borrow as successful and will not allow the client to attempt another borrow. Instead, a second borrow request will fail with the message "Feature already borrowed on server." The client must then wait until the borrow time expires to successfully borrow a license if this network problem occurs.

## Automatic server discovery

LM-X License Manager's automatic server discovery allows a client application to find license servers on the network automatically. This feature eliminates the need for end users to enter server information and makes it easy for system administrators to move from one license server to another without notifying users. Automatic server discovery is performed only when the license server is unavailable or the feature couldn't be found on the known license servers. Otherwise, a local cache is used, making this feature unobtrusive and lightweight.

**Note:** Automatic server discovery works only on local networks and will not work on WAN or VPN connections.

If your software vendor allows automatic server discovery to be used for its licenses, you can enable it by setting an LM-X environment variable specifically for this purpose, as described below. (Instructions given for accessing environment variable settings are for Windows. Please see your OS documentation or your system administrator for instructions on editing environment variables for your specific OS.)

1. Open the Windows Control Panel and select **System and Security**.
2. Select **System** from the System and Security options.
3. Select **Advanced system settings** from the list of options in the left column of the System window.
4. From the System Properties dialog that appears, select **Environment Variables...**
5. Under System variables, select **New...**
6. For the Variable name, enter LMX\_AUTOMATIC\_SERVER\_DISCOVERY or vendor\_AUTOMATIC\_SERVER\_DISCOVERY (where vendor is the name of the software vendor for which to use automatic server discovery) and set the value to 1 (or any other integer) to enable it.
7. Click **OK** to add the variable.
8. Click **OK** from the Environment Variables dialog to save your changes, and then click **OK** from the System Properties dialog to close it.

## Queuing licenses

License queuing, which must be enabled by your vendor, helps you implement fair usage when licenses may not be immediately available. This feature is particularly useful for jobs scheduled for automatic execution.

Without queuing enabled, checkout requests are immediately denied if the required license(s) are unavailable. If queuing is enabled, its behavior is determined by the vendor; for example, applications may be placed on hold until the necessary license(s) become available. Contact your vendor for information about how license queuing has been implemented for your licenses. All license requests are appended to the end of the queue by default, regardless whether the request can be fulfilled immediately.

To enable license queuing, set the LM-X environment variable `LMX_QUEUE`, as described below (see also [Environment variables](#)). (Instructions given for accessing environment variable settings are for Windows 7. Please see your OS documentation or your system administrator for instructions on editing environment variables for your specific OS.)

1. Open the Windows Control Panel and select **System and Security**.
2. Select **System** from the System and Security options.
3. Select **Advanced system settings** from the list of options in the left column of the System window.
4. From the System Properties dialog that appears, select **Environment Variables...**
5. Under System variables, select **New...**
6. For the Variable name, enter `LMX_QUEUE` or `vendor_QUEUE` (where *vendor* is the name of the software vendor for which to use license queuing) and set the value to 1 (or any positive number) to enable it.
7. Click **OK** to add the variable.
8. Click **OK** from the Environment Variables dialog to save your changes, and then click **OK** from the System Properties dialog to close it.

Alternatively, you can enable fast queuing, which allows requests to be fulfilled immediately when possible. Fast queuing can allow smaller license requests to be processed more promptly and help ensure higher license utilization. However, because it might enable users to bypass the queue, it does not necessarily implement fairness.

For example, if a client is waiting for two licenses, and only one license is immediately available, another client that needs only one license can bypass the queue and take the single license without waiting.

To enable fast queuing, edit the `FAST_QUEUE` option in your license server configuration file. You can enable fast queuing for specified features or for all features; for example:

```
FAST_QUEUE = f1, f3
or
FAST_QUEUE = ALL
```

# Logs

LM-X includes the following logs.

Log	Description
License server log	<p>This log lists server actions that are helpful for debugging. You can configure this log for normal or extended detail, as well as set rotation intervals, name and path, in the <a href="#">license server configuration file</a>.</p> <p>Example of normal license server log:</p> <pre>[2010-10-07 10:53:46] LM-X License Server v3.6 build 7504 on Konradm-PC (Win32_x86) [2010-10-07 10:53:46] Copyright (C) 2002-2010 X-Formation. All rights reserved. [2010-10-07 10:53:46] Website: http://www.lm-x.com http://www.x-formation.com [2010-10-07 10:53:46] License server has pid 4676. [2010-10-07 10:53:46] Serving licenses for vendor XFORMATION. [2010-10-07 10:53:46] [2010-10-07 10:53:46] License server using TCP IPv4 port 6200. [2010-10-07 10:53:46] License server using TCP IPv6 port 6200. [2010-10-07 10:53:46] License server using UDP IPv4 port 6200. [2010-10-07 10:53:47] Reading licenses... [2010-10-07 10:53:47] License file(s): [2010-10-07 10:53:47] C:\x-formation\lm-x\win32_x86_vc2008\..\..\..\out/ win32_x86/win32_x86/..\examples/network/network.lic [2010-10-07 10:53:48] Log file path: C:\x-formation\lm-x\win32_x86_vc2008\ licserver-accept-examples.log [2010-10-07 10:53:48] Log to stdout: Yes [2010-10-07 10:53:48] Log format: Normal [2010-10-07 10:53:48] Configuration file path: C:\x-formation\lm-x\win32_x86_vc2008\ ..\..\lmx-serv-examples.cfg [2010-10-07 10:53:48] Serving following features:[2010-10-07 10:53:48] f1 (v1.5) (5 license(s)) license type: exclusive [2010-10-07 10:53:48] f2 (v1.0) (5 license(s)) shared on: HOST USER license type: additive [2010-10-07 10:53:48] f2 (v1.0) (10 license(s)) license type: additive [2010-10-07 10:53:48] borrow (v1.0) (1 license(s)) license type: exclusive [2010-10-07 10:53:48] LicensedExampleClass (v1.0) (1 license(s)) license type: exclusive</pre>
Extended client log	<p>This log lists client actions, entirely for diagnostic purposes. This log is disabled by default. You can enable it using the <a href="#">LMX_EXTENDED_LOG</a> environment variable as described in <a href="#">Environment variables</a>.</p> <p>This log will be generated or appended to every time a checkout request occurs.</p> <p>Example:</p> <pre>[2010-10-14 11:49:35] LM-X Extended Client Log [2010-10-14 11:49:36] [2010-10-14 11:49:36] Machine settings: [2010-10-14 11:49:36] Username: pkukielka [2010-10-14 11:49:36] Hostname: SUPPORT2-PC [2010-10-14 11:49:36] Vendor name: XFORMATION [2010-10-14 11:49:36] Working directory: c:\x-formation [2010-10-14 11:49:36] Platform: Win32_x86 [2010-10-14 11:49:36] Version: 6.1 [2010-10-14 11:49:36] Virtual machine: No [2010-10-14 11:49:36] Terminal server session: No [2010-10-14 11:49:36] [2010-10-14 11:49:36] Environment variables: [2010-10-14 11:49:36] LMX_AUTOMATIC_SERVER_DISCOVERY = &lt;not set&gt; [2010-10-14 11:49:36] LMX_QUEUE = &lt;not set&gt; [2010-10-14 11:49:36] LMX_PROJECT = &lt;not set&gt; [2010-10-14 11:49:36] LMX_BORROW = 1 [2010-10-14 11:49:36] LMX_LICENSE_PATH = &lt;not set&gt; [2010-10-14 11:49:36] LMX_CONNECTION_TIMEOUT = 30</pre>

lmxendutil - licstat output	<p>This report contains license usage statistics, lists which users are currently using which licenses on a specific license server, and gives information on the borrow, grace and trial licenses currently checked out. See <a href="#">LM-X End-user utility</a> for information on generating this report.</p> <p>Example:</p> <pre> LM-X End-user Utility v3.5 Copyright (C) 2002-2010 X-Formation. All rights reserved. Searching all license paths and performing automatic server discovery... +++++ LM-X License Server on 6400@192.168.22.103: Server version: v3.6 Uptime: 0 hour(s) 0 min(s) 4 sec(s) ----- Feature: two_licserver Version: 1.0 Vendor: XFORMATION Start date: NONE Expire date: NONE Key type: EXCLUSIVE License sharing: VIRTUAL 10 of 10 license(s) used: 10 license(s) used by hg@vista-x64 [::1] Login time: 2010-10-14 15:37 Checkout time: 2010-10-14 15:37 +++++ LM-X License Server on 6200@192.168.22.103: Server version: v3.6 Uptime: 0 hour(s) 0 min(s) 23 sec(s) ----- Feature: f3 Version: 1.5 Vendor: XFORMATION Start date: NONE Expire date: NONE Key type: EXCLUSIVE 0 of 5 license(s) used ----- Feature: f2 Version: 1.0 Vendor: XFORMATION Start date: NONE Expire date: NONE Key type: EXCLUSIVE License sharing: VIRTUAL 0 of 5 license(s) used ----- Feature: grace Version: 1.0 Vendor: XFORMATION Start date: NONE Expire date: NONE Key type: EXCLUSIVE License sharing: VIRTUAL 0 of 5 license(s) used </pre>
lmxendutil - licstatxml output	<p>This report contains license usage statistics and lists currently used licenses on a specific license server as XML. See <a href="#">LM-X End-user utility</a> for information on generating this report.</p> <p>Example:</p> <pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;LM-X_STAT_VERSION="3.6"&gt; &lt;LICENSE_PATH TYPE="NETWORK" HOST="6200@rafal-PC" SERVER_VERSION="3.6" UPTIME="0 \\hour(s) 0 min(s) 30 sec(s)" STATUS="SUCCESS"&gt; &lt;FEATURE NAME="f1" VERSION="1.5" VENDOR="XFORMATION" END="2018-01-01" \\USED_LICENSES="0" TOTAL_LICENSES="5" SHARE="VIRTUAL"/&gt; &lt;FEATURE NAME="f2" VERSION="1.0" VENDOR="XFORMATION" END="2018-01-01" \\USED_LICENSES="0" TOTAL_LICENSES="5" SHARE="HOST ,USER ,VIRTUAL"/&gt; &lt;FEATURE NAME="f2" VERSION="1.0" VENDOR="XFORMATION" END="2018-01-01" \\USED_LICENSES="0" TOTAL_LICENSES="10" SHARE="VIRTUAL"/&gt; &lt;FEATURE NAME="borrow" VERSION="1.0" VENDOR="XFORMATION" USED_LICENSES="0" \\TOTAL_LICENSES="1" SHARE="VIRTUAL"/&gt; &lt;FEATURE NAME="LicensedExampleClass" VERSION="1.0" VENDOR="XFORMATION" \\USED_LICENSES="0" TOTAL_LICENSES="1" SHARE="VIRTUAL"/&gt; &lt;FEATURE NAME="prod_a" VERSION="1.0" VENDOR="XFORMATION" USED_LICENSES="0" \\TOTAL_LICENSES="5" SHARE="VIRTUAL"/&gt; &lt;FEATURE NAME="prod_b" VERSION="1.0" VENDOR="XFORMATION" USED_LICENSES="0" \\TOTAL_LICENSES="5" SHARE="VIRTUAL"/&gt; </pre>

<p>Pay-per-use database file</p>	<p>This database file lists pay-per-use information that can be sent to your vendor for billing purposes. See <a href="#">Pay Per Use feature</a> for more information.</p> <p>Example:</p> <pre>Timestamp: 2009-01-30 07:37:06 Action: CHECKOUT Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 3 Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 402 Timestamp: 2009-01-30 07:37:06 Action: CHECKOUT Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 3 Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 405 Timestamp: 2009-01-30 07:37:06 Action: CHECKIN Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 32 Timestamp: 2009-01-30 07:37:06 Action: USAGE Comment: FEATURE: TEST_MULTIUSER UNIQUEID: C48E3CC9D201846D8FC3F2C0B22289323D64E026750ECBFE COUNT: 373</pre>
----------------------------------	---

## Error messages

For a complete list of error messages that may be encountered for LM-X License Manager, please refer to [Return codes](#).

## End-user solutions

The following sections offer solutions to potential problems you may encounter when using LM-X License Manager. You can also find additional topics in the [LM-X Enduser](#) section of the [X-Formation Knowledgebase](#).

- [Managing multiple programs running against multiple versions of LM-X](#)



# Managing multiple applications running against multiple versions of LM-X

To manage multiple applications running against multiple versions of LM-X:

- For [local licenses](#): you only need to ensure that the license path for each application is correct for its respective license.
- For [network licenses](#): we recommend that you use the tools provided by your application vendor. You can easily start up multiple license servers on the same machine.

If you are certain that multiple licenses are using the same version of LM-X, you can have a single license server host multiple licenses. The easiest way to test this is to add each license to the [license server configuration file](#) and make sure that the license server still starts up.

# LM-X End User's FAQ

This FAQ section addresses the most commonly-asked questions about LM-X License Server. We try to regularly update this section with new information to help you quickly and easily find the information you're looking for. If you have a question that we have not answered here, please [contact our support team](#).

You can find answers to your questions in the following sections:

Usage Information
<a href="#">How can I obtain LM-X License Manager dongle drivers?</a>
<a href="#">BIOS/HARDDISK serial numbers disappear on Windows</a>
<a href="#">Why am I getting communication errors between the client and LM-X license server?</a>
<a href="#">Does LM-X License Manager allow our users to travel with their licenses?</a>
<a href="#">Can end users administer and monitor their licenses?</a>
<a href="#">Can I combine multiple license files into one?</a>

# Usage Information

## How can I obtain LM-X License Manager dongle drivers?

At X-Formation we offer Sentinel HASP HL Pro dongles. For more information about downloading the latest Sentinel HASP drivers, go to [End User Tools Downloads](#).

**Note:** If an LM-X dongle is plugged into a machine that does not have the dongle drivers installed, the Windows "Found New Hardware" dialog will appear. You can follow the prompts in this dialog to install the drivers using Windows update.

You may also download the dongle drivers attached to this article. If desired, software vendors can ship the drivers to customers along with the dongle.

## BIOS/HARDDISK serial numbers disappear on Windows

If the serial numbers of BIOS and HARDDISK HostIDs disappear on your Windows machine, you should check if all the drivers for your motherboard are correctly installed.

Furthermore, one workaround to mitigate this problem is to use the HostID match rating setting, [HOSTID\\_MATCH\\_RATE](#).

For more information about this problem, see [Operating system issues](#).

## Why am I getting communication errors between the client and LM-X license server?

Communication errors between the client and LM-X license server may indicate a problem with the client machine.

See [Communication issues](#) for more information.

## Does LM-X License Manager allow our users to travel with their licenses?

LM-X License Manager supports a wide range of license policies, including [borrowing a license](#). For an overview of some of the license policies that LM-X can handle, see [Define a license policy](#).

## Can end users administer and monitor their licenses?

With LM-X License Manager, software vendors can provide end users with a license file they can use as a standalone license or with a license server. Our license monitoring software, [License Statistics](#), lets endusers monitor the use of floating or standalone licenses.

## Can I combine multiple license files into one?

Although it is usually possible to combine licenses, we recommend that in most cases individual licenses are kept separate. LM-X protected applications will search specified license paths for standalone licenses until a correct license is found, so there is generally no reason for combining licenses.

# Troubleshooting end-user problems

If you experience problems with LM-X, we may ask you to [enable extended logging](#), and send us the [log files](#) from the LM-X log directory.

Troubleshooting for specific issues is covered in the following sections:

[Enabling extended logging in LM-X](#)

[Communication issues](#)

[Locale issues](#)

[Operating system issues](#)

[System clock check issues](#)

[System security issues](#)

# Enabling extended logging in LM-X

If you experience problems with LM-X, we may ask you to enable extended logging, as described below.

## Enabling extended logging for client-side logs

*To enable client-side extended logging in Windows:*

1. Set the environment variable [LMX\\_EXTENDEDLOG](#); for example:  

```
$ set LMX_EXTENDEDLOG=C:\Users\john\lmx-extended.txt  
$ my-application.exe
```
2. Restart your licensed software.

*To enable client-side extended logging in Linux:*

1. Set the environment variable [LMX\\_EXTENDEDLOG](#); for example:  

```
$ export LMX_EXTENDEDLOG="/var/log/lmx-extended.log"  
$ ./my-application
```
2. Restart your licensed software.

## Enabling extended logging for license server

*To enable extended logging for the license server in Windows or Linux, either:*

- [Enable extended logging](#) in the [license server configuration file](#).  
or
- [Set the extended logging in the LM-X License Server Client](#).

For more information about license server logging, see [License server log file](#).

# Communication issues

## Failure to establish communication with the network

If communication between the client machine and the [LM-X License Server](#) cannot be established, you may see one or more of the following errors:

Error 3, LMX\_NO\_NETWORK: Unable to initialize network subsystem.  
Error 15, LMX\_NO\_NETWORK\_HOST: Unable to connect to license server.  
Error 16, LMX\_NETWORK\_DENY: Rejected actively from license server.

These error codes indicate a problem with the client machine; for example, there may be no network card, the license server may be down, or user permissions may be inadequate.

## Problem after establishing connection with the network

If communication between the client machine and the LM-X License Server fails, you may see the following error:

Error 47, LMX\_NETWORK\_RECEIVE\_ERROR: Error receiving message over network.

This error indicates a problem with the client machine failing to maintain a TCP connection; for example, a firewall may be blocking a TCP connection, or connection to the server may be slow.

# Locale issues

## "locale::facet::\_S\_create\_c\_locale name not valid" error on Ubuntu 14.04.

LM-X performs checks to verify that environment variables (for example LC\_ALL or LANG) are set. If they are not set, then LM-X sets them to "C". However, if the environment variable does not point to a valid locale, then LM-X does not perform any action, because it considers such variable to be set. When LM-X is unable to find a valid locale, you will see an error similar to the following:

```
locale::facet::_S_create_c_locale name not valid
```

This is not an LM-X error, but rather indicates a problem with your local machine caused by a misconfigured locale.

You can use the following command to output information about the current locale environment on your system:

```
locale
```

To check if current locales are supported by your operating system, run the following command:

```
locale -a
```

To install the locale, run:

```
sudo locale-gen name_of_the_locale
```

For more information about locales, please refer to [official Ubuntu documentation](#), or see the documentation relevant for your operating system.

# Operating system issues

## BIOS/HARDDISK serial numbers disappear on Windows

If the serial numbers of BIOS and HARDDISK HostIDs disappear on your Windows machine, you should check if all the drivers for your motherboard are correctly installed. This issue may occur after you have installed new firmware updates, or replaced generic Windows drivers with vendor-specific ones; for example for better OS performance, or to fix bugs.

This is not an LM-X License Manager error, but rather indicates that the drivers provided by your hardware vendor do not work properly after update. To resolve this problem, make sure your drivers are correctly installed. Contact your hardware vendor for details on how to resolve this issue.

### Mitigating the problem

One workaround to this problem is to use the HostID match rating setting, [HOSTID\\_MATCH\\_RATE](#) when [locking a license to multiple HostIDs](#). This setting lets you specify the percentage required for a successful match, from 0 (no matching required) to 100 (exact matching required). For example, SETTING HOSTID\_MATCH\_RATE="50" indicates that a 50% or better match must be achieved so that the license can work. This means that you allow your users to run the software if only two out of four HostIDs values are valid.

## "System time differs more than 24 hours" error

If there is a significant discrepancy between the [LM-X License Server](#) and license client, your lmx-server.log file will report an error message similar to the following:

```
[2013-09-24 11:44:07] CHECKOUT by pcuratolo@pcuratolo: FRONTEND
[2013-09-24 11:44:33] FAIL: Client pcuratolo@pcuratolo system time differs more than 24 hours. Please correct it!
[2013-09-24 11:44:33] CHECKIN by pcuratolo@pcuratolo: FRONTEND
[2013-09-24 11:44:33] FAIL: Client pcuratolo@pcuratolo system time differs more than 24 hours. Please correct it!
[2013-09-24 11:44:33] FAIL: New client pcuratolo@pcuratolo not allowed to connect.
```

This check is performed by LM-X to protect against deliberate attempts to set the system clock time in order to tamper with their license expiration time. To resolve this problem, you must ensure that server and client machine clocks are not more than 24 hours apart.

## LMX\_Free hangs when used within a dll file

When you have a dll, you typically have initialization code like the following:

```
BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }

    return TRUE;
}
```

Using this code to initialize and free LM-X handles can cause [LMX\\_Free](#) to hang when using the DETACH handlers, because the OS can close threads without passing the information to the LM-X library. You should instead initialize LM-X by explicitly calling code to initialize and free LM-X from the exported dll interface.



# System clock check issues

## "System clock has been set back" error

If you get a "system clock has been set back" error from LM-X License Manager, LM-X has detected that your system clock has been set back. Under some circumstances, the check can return a false positive. For example, you might have accidentally changed the clock to a future time.

To resolve this problem, you should set the system clock to the correct (present) time. If you believe the error is a mistake and your system clock seems to be correct, ask your application vendor for assistance.

For more information, see [System clock check](#).

## "WARN - License checkout not successful for session ..." error

LM-X License Manager may not work as expected if you get an error similar to the following:

```
2016-11-16 11:28:44,435 WARN - License checkout not successful for session Session ID/ Name. Error: LM-X Error:
(Internal: 98 Feature: Name of feature)
System clock has been set back from 1970-Jan-01 00:00:00
For further information go to http://www.x-formation.com
```

Assuming that LM-X client is embedded into an application running on IIS web server and, therefore, uses IIS\_IUSRS identity, you need to have read access permissions to the following folder:

```
C:\Windows\SysWOW64\config\systemprofile\AppData\Local\x-formation
```

To resolve this issue, set read access permissions for IIS\_IUSRS.

## Why is my system clock set to a distant date in the future?

The Windows file system is based on the [Gregorian calendar](#), which was instituted in 1582. Because of the constraints of the Gregorian calendar, you may get a "system clock has been set back" error indicating that the clock has been set back from a distant date in the future, for example:

```
System clock has been set back from 2022-Oct-16 17:30:49
```

ANSI C defines 32-bit `time_t`, which is interpreted as the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970 to 03:14:07 UTC on 19 January 2038.

LM-X License Manager uses `time_t`, because this is the only portable representation of time across operating systems. If a file from outside the span specified by `time_t` (for example 1750) is detected by Windows, the Windows API changes the time to the one that fits the range represented by the `time_t`, thereby resulting in LM-X informing the user that system clock check has been set back from a random date.

To fix this problem, reset the system clock using the `LmxResetSystemClock` tool to correct the time in the affected files. (See [System clock check](#) for more details.)

## Problem with verifying system clock

If there is a problem with using `lmxresetsystemclock.exe` to synchronize your local system's clock with a remote time server, you may see the following error:

```
LM-X Error: Could not verify system clock due to failed connection with remote time server
```

To resolve this problem, it is necessary to verify that UDP port 123 (NTP protocol) is not blocked by a firewall.

# System security issues

## "FAIL: CreateWindow > failed: 0" error

If there are problems with the socket stack and conflict between firewall or antivirus application and one of the hosts, you may see an error message similar to the following in the LM-X log file:

```
FAIL: CreateWindow > failed: 0
```

To fix this problem, you should verify your operating system security settings.

## LM-X Reference for FlexNet Users

This section provides a reference for users who are familiar with FlexNet/FLEXlm. You can use this reference to determine how FlexNet/FLEXlm-specific actions correspond with LM-X actions. This reference assumes you have experience using FlexNet/FLEXlm.

## Comparison of license files

LM-X licenses are text files, just like FlexNet/FLEXlm licenses. A typical LM-X license looks like the following example.

```
FEATURE my_app
\{
VENDOR = XFORMATION
VERSION = 1.0
END = 2015-03-27
LICENSEE = "COMPANY_ABC"
KEY = RNCLMCH56104622J8JAA5U0EROSHD9RA22E5HOFEQSKVSTA5RO \
VGK7M5884EEI9V1PRB97BBAIH5AN4PU73LQUQDSDM
\}
```

LM-X has the same concept of features and vendors that FlexNet/FLEXlm does. However, the LM-X format is easier to understand, because everything is marked by specific keywords.

For more information about LM-X license files, see [License files](#).

## Comparison of license paths

You use license paths to describe where licenses are found. In FlexNet/FLEXlm, the environment variable `LM_LICENSE_FILE` or `VENDOR_LICENSE_FILE` defines where the application should look for a license. With LM-X, you use the variable `LMX_LICENSE_PATH` or `VENDOR_LICENSE_PATH`.

The following tables show examples of specifying the path to a local license.

FlexNet/FLEXlm	LM-X
Windows: <code>LM_LICENSE_FILE=C:\application\license.dat</code>	Windows: <code>LMX_LICENSE_PATH=C:\application\myapplication.lic</code>
Unix: <code>LM_LICENSE_FILE=/usr/application/license.dat</code>	Unix: <code>LMX_LICENSE_PATH=/usr/application/myapplication.lic</code>

Examples of specifying the path to a network license:

FlexNet/FLEXlm	LM-X
Windows: <code>LM_LICENSE_FILE=port@host</code>	Windows: <code>LMX_LICENSE_PATH=port@host</code>
Unix: <code>LM_LICENSE_FILE=port@host</code>	Unix: <code>LMX_LICENSE_PATH=port@host</code>

## Comparison of license server setup

When setting up a network license, you must set up a license server. The table below specifies the files required for a network license setup and how they relate to FlexNet/FLEXlm files.

FlexNet/FLEXlm	LM-X
License server	License server
Windows: lmgrd.exe Unix: lmgrd	Windows: lmx-serv.exe Unix: lmx-serv
Vendor daemon	Vendor daemon
Windows: <i>vendord.exe</i> Unix: <i>vendord</i>	Windows: lmx-serv.exe Unix: lmx-serv
License file license.dat	License file networkapp.lic
Option file <i>vendord.opt</i>	Configuration file lmx-serv.cfg

Note that lmgrd and the vendor daemon are collected into one license server in LM-X.

Instead of specifying port numbers, SERVER lines and optional information in the license file and option file, you specify this information in the [license server configuration file](#), lmx-serv.cfg. Some settings, such as license file and log file paths and port number, may also be specified at the command line, as described in [Running the license server from a command line](#).

When you want to set up your network license server, make sure that you have the lmx-serv executable, lmx-serv.cfg, and your network license.

## The LM-X License Server

The LM-X License Server works in the same way as a FlexNet/FLEXlm license server. It serves licenses by listening on a specific TCP port and allowing you to check out licenses for network use or for borrowing, if enabled by your application vendor. Additionally, the server uses UDP for automatic server discovery. This enables applications to find the server without specifying a hostname.

Under Windows, you can run the license server in the console or as a service that runs in the background. Under Unix, you can run the license server in the foreground or background.

## Comparison of license server parameters

To see and change license server parameters, refer to the chart below.

Action to perform	FlexNet /FLEXlm	LM-X
See if the license server is up or who is using the license server	lmutil lmstat lmutil lmdiag	lmxendutil -licstat -host <i>host</i> -port <i>port</i>
See the hostid of the client or server machine	lmutil lmhostid	lmxendutil -hostid
Remotely shutdown the license server	lmutil lmdown	lmxendutil -shutdownserver -host <i>host</i> -port <i>port</i> -password <i>password</i>
Remotely restart the license server	lmutil lmreread	lmxendutil -restartserver -host <i>host</i> -port <i>port</i> -password <i>password</i>
Remove a user from the license server	lmutil lmremove	lmxendutil -removeuser -clientusername <i>username</i> -clienthostname <i>host</i> -host <i>host</i> -port <i>port</i> -password <i>password</i>





# Administration

The following sections describe the administrative functions of the LM-X web-based UI, which are located as sub-tabs under the Administration tab. Many settings in the Administration sub-tabs give you a user interface for modifying the contents of the configuration file; for example, setting values for HAL servers, the Pay Per Use database, logging, etc.

To ensure that the administrative functions are accessed only by authorized administrators, you must enter a valid password to open the Administration tab. The password is stored in the configuration file and can be seen when setting up the license server.

# Managing licenses

To manage licenses, go to the License Management sub-tab under the Administration tab. From this page, you can activate (download) a license from License Activation Center (LAC). In addition, you can upload license files; for example, a file that your vendor has sent to you via email or a license that you have previously activated and want to upload to the server.

- To activate a license, enter the activation key, and then click **Activate**. The activation key is added and automatically activated.
- To reactivate a license, click the **Reactivate** icon () in the Action column for the license you want to reactivate. The license is removed and activated again.
- To delete a license, click the **Delete** icon () in the Action column for the license you want to remove.
- To upload existing license files from LAC, click **Upload License File**, and then save the file to the desired location using the Upload File dialog that appears.

For general information about LAC, see [Introduction to License Activation Center](#). For information about activating licenses in LAC, see [License Activation Center End User Guide](#).

## Enabling and configuring HAL

To enable and configure High Availability Licensing (HAL), go to the High Availability Licensing sub-tab under the Administration tab.

1. To enable HAL, check the "Enable" checkbox.
2. Configure the HAL servers by entering the server information under the Configuration area.
3. To save your changes, click **Save**. You must restart the server for your changes to take effect. You can cancel any changes not yet saved by clicking **Cancel**.

For more information about HAL, see [High Availability Licensing](#).

# Enabling and configuring Pay Per Use

To enable and configure Pay Per Use, go to the Pay Per Use sub-tab under the Administration tab.

- To enable Pay Per Use, check the **Enable** checkbox.
- Under the Configuration area, you can:
  - Change the Pay Per Use database name if desired.
  - Specify the desired format (Normal or Extended).
  - If using Extended format, you can also enable "Usage Anonymization," which hashes the usernames in the database. After turning this option on, you cannot undo the anonymization of the usernames in the database. Therefore, although this setting can be enabled or disabled at any time, we recommend that it remains set either on or off to avoid having a partially anonymized database.
- To download the Pay Per Use database file (for example, to deliver it to your vendor), click **Download** and select the desired location and file name for the file using the Save dialog that appears.
- To save your changes, click **Save**. You must restart the server for your changes to take effect.
- You can cancel any changes not yet saved by clicking the **Cancel** button.

For more information about how to use and configure Pay Per Use, see [Pay Per Use feature](#).

# Enabling and configuring license server logging

To enable and configure license server logging, go to the Logging sub-tab under the Administration tab.

1. To enable Logging, check the **Enable** checkbox.
2. Under the Configuration area, you can specify the name for the log file, the format of the file (Normal or Extended), the rotation interval (None, Daily, Weekly or Monthly) to use for removing older data, and any LM-X messages to exclude from logging.

When using extended logging note that:

- Extended logging results in greater detail in the log file.
- Extended logs can be [imported into License Statistics](#) to obtain denied request statistics.

3. To save your changes, click **Save**. You must restart the server for your changes to take effect.

The following is an example of enabling extended logging.



The screenshot shows the LM-X License Server web interface. At the top is the logo and title 'LM-X License Server'. Below it is a navigation bar with tabs: Dashboard, HostIDs, License Usage, Log File, and Administration. The 'Administration' tab is selected, and within it, the 'Logging' sub-tab is active. The page content area has a sub-navigation bar with tabs: License Management, High Availability Licensing, Pay Per Use, Logging, Configuration File, and Misc. The 'Logging' sub-tab is selected. The main content area has a heading 'This page lets you configure logging for the LM-X license server.' followed by a checkbox labeled 'Enable' which is checked. Below this is a 'Configuration' section with the following fields: 'Filename:' with a text input containing '/usr/lmx/lmx-serv.log'; 'Format:' with a dropdown menu set to 'Extended'; 'Rotation Interval:' with a dropdown menu set to 'None'; and 'Exclude From Logging:' with a text area containing a list of log messages: CHECKOUT, CHECKIN, STATUS, BORROW, BORROW\_RETURN, REMOVE\_USER, REMOTE\_RESTART, REMOTE\_SHUTDOWN, and AUTOMATIC\_DISCOVERY.

For more information about license server logging, see [License server log file](#) and [License server configuration file](#).

## Editing the configuration file

The Configuration File sub-tab under the Administration tab lets you review and change the LM-X [License Server configuration file](#).

- After making changes to the configuration file, click **Save**. You must restart the server for your changes to take effect.
- You can cancel any changes not yet saved by clicking the **Cancel** button.

## Miscellaneous settings

The Misc sub-tab under the Administration tab lets you configure settings for TCP, remote access and user removal. From this page, you can:

- Specify the TCP port number in the TCP Listen Port field. (For more information about the TCP Listen Port, see [License server configuration file.](#))
- Enter a password for remote access to the license server in the Remote Access Password field. You must verify the password by entering it again in the Confirm Password field.
- Enable user removal by checking the Enable User Removal checkbox.
- Set the minimum time for user removal. The value for user removal must be no less than 120 (seconds). (For more information about user removal, see [License server configuration file.](#))

## Removed features

This section includes documentation for functionality that no longer applies to the current version of LM-X. If you are running an older version of LM-X, you may refer to these sections for information applicable to your LM-X version.

Removed or outdated functionality includes:

Feature	Removed in
<a href="#">Installing and uninstalling a license server on Windows</a>	v4.4.3
<a href="#">Installing a license server on Mac OS X</a>	v4.4.3
<a href="#">LMX_AtExit</a>	v4.6
<a href="#">Web-based UI</a>	v4.8

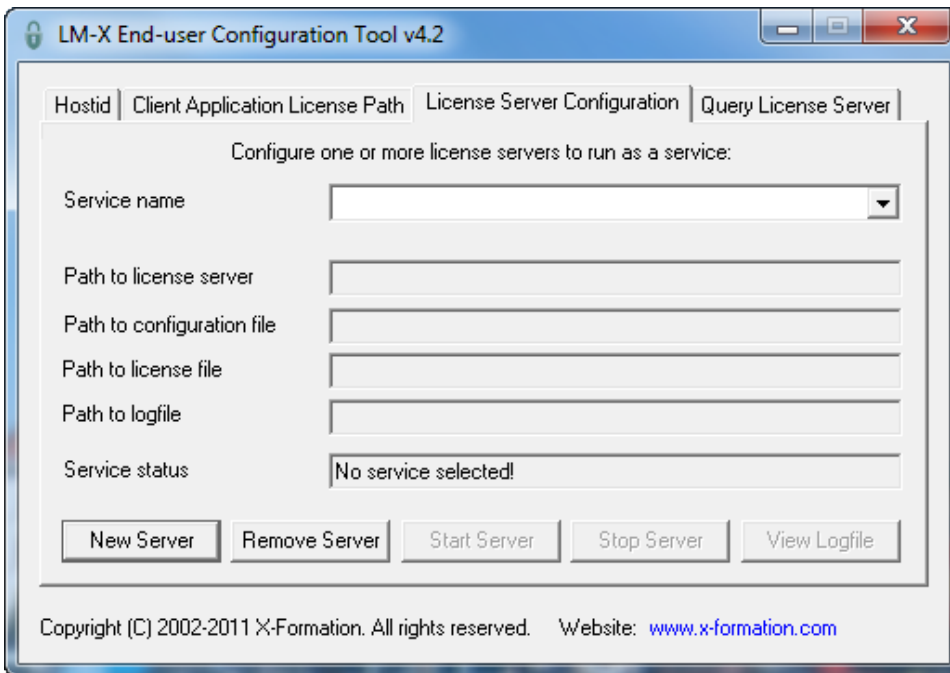


# Installing and uninstalling a license server on Windows

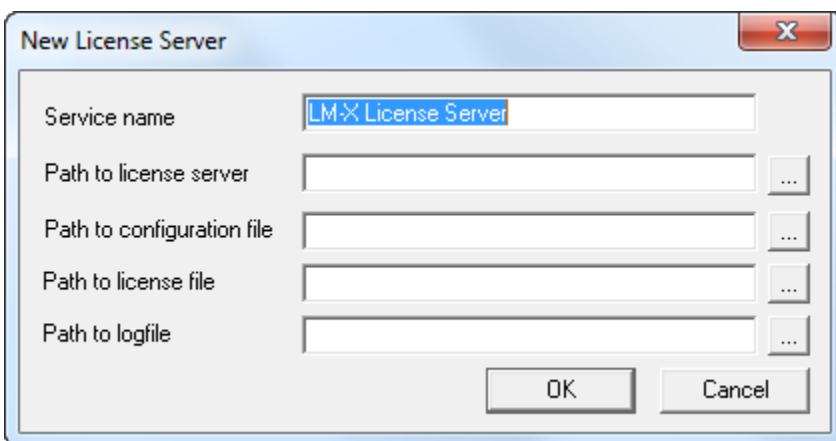
*The information on this page refers to versions prior to 4.4.3, which introduced an installation program that installs both the license server and the end-user tools. Manual installation is no longer required for Windows.*

The following steps describe how to easily install a license server as a service on Windows using the [LM-X End-user Configuration Tool](#) (Windows only). If you are using a HAL (High Availability License) server configuration, see [High Availability Licensing](#) for HAL-specific installation information.

1. Copy lmx-serv.exe and, if your vendor included a configuration file, lmx-serv.cfg into the directory where your server will reside. For example, C:\Program Files\Application.
2. Start the LM-X End-user Configuration Tool, lmxconfigtool.exe.
3. Click the **License Server Configuration** tab. This tab lets you manage license servers, including adding a new server, starting, stopping, and removing the server, monitoring its status and viewing its logfile.



4. In the License Server Configuration tab, click **New Server**.



5. In the New License Server dialog, specify the path to the license server location. For example, C:\program files\application\lmx-serv-vendor.exe. (You may browse for the file using the ... button.)
6. If your vendor included a configuration file, browse for and select the configuration file location. For example, C:\program files\application\lmx-serv.cfg. (You may browse for the file using the ... button.)
7. Additionally, you may add an optional license file and logfile to be used. Note that the logfile will override any settings for logfiles in the configuration file.
8. Click **OK** to install the license server as a service. (**Note:** The license server will not start until you reboot your computer or start it by clicking **Start Server**. You will do this later.)
9. If your vendor supplied a configuration file (lmx-serv.cfg by default), open it in a text editor, such as Notepad, and edit the entries for LOG\_FILE and LICENSE\_FILE to reflect the correct paths for your log file and license file. If you specified a license file in the dialog, it will be read in addition to any specified in the configuration file. If you specified a logfile in the dialog, it will override any settings in the configuration file. (You may also need to make additional settings in the configuration file depending on your specific needs. For example, if you are using HAL licensing, you must edit the HAL settings as specified in [How to install HAL license servers](#).)

10. In the License Server Configuration tab, click **Start Server**.
11. Click **View Logfile** to verify the license server is started and working normally, indicated by the line "Ready to serve..." as shown in the following example.

```
[2009-01-25 20:43:07] LM-X License Server v2.1 on amilo (Win32 x86)
[2009-01-25 20:43:07] Copyright (C) 2002-2009 X-Formation. All rights reserved.
[2009-01-25 20:43:07] Website: http://www.lm-x.com http://www.x-formation.com
[2009-01-25 20:43:07] License server has pid 17228.
[2009-01-25 20:43:07] Serving licenses for XFORMATION.
[2009-01-25 20:43:07] License server using TCP port 6200.
[2009-01-25 20:43:07] License server using UDP port 6200.
[2009-01-25 20:43:07] Logfile path: lmx-server.log
[2009-01-25 20:43:07] License file(s):
[2009-01-25 20:43:07] application.lic
[2009-01-25 20:43:07] Serving following features:
[2009-01-25 20:43:07] F3 (v1.5) (5 licenses) license type: exclusive
[2009-01-25 20:43:07] F2 (v1.0) (5 licenses) license type: exclusive
[2009-01-25 20:43:07] Ready to serve...
```

If there are any problems running the license server, the log file will indicate the cause of the problem. If you cannot find a log file, check to make sure that the correct configuration file was specified in step 6, and the configuration file contains the correct information. (Also see Limitations for installing LM-X License Server as a service, below.)

## Limitations for installing LM-X License Server as a service

When installing LM-X License Server as a service, you should note the following limitations:

- The display name of installed LM-X license server service must be unique; if [LM-X end-user tools](#) have already been used with the LM-X License Server as a service, you must exercise care to use a different display name in subsequent installations.
- Two distinct LM-X license servers cannot be run as a service using liblmxvendor.dll files from the same vendor; for example, Server A can run with liblmxvendor.dll issued by vendor "XF", and Server B with "XFORMATION", but Server C can run with neither "XF" or "XFORMATION".

## Uninstalling LM-X License Server

To uninstall LM-X License Server on Windows:

1. Uninstall the license server using the command line or [LM-X End-user Configuration Tool](#).
2. Remove [environment variables](#) (you may use the Client Application License Path tab in the [LM-X End-user Configuration Tool](#)).
3. Delete the remaining files such as [License server configuration file](#), utilities, etc.

# Installing a license server on Mac OS X

*The information on this page refers to versions prior to 4.4.3, which introduced an installation program that installs both the license server and the end-user tools. Manual installation is no longer required for Mac OS X.*

The attached script can be used to run a Mac OS X license server at system startup. This file must be placed in the /System/Library/LaunchDaemons/ directory.

All files used by the license server (licenses, logfile, configuration file, server executable) should be placed in /usr/xformation and must be accessible for the "daemon" user. General privileges can be set to 755 using chmod, as follows:

```
chmod -R 755 /usr/x-formation
```

The owner and group for the license server executable must have sufficient rights to run it, which you can set using the commands chown and chgrp.

**Download file**

[com.xformation.lmx.plist](#)

# LMX\_AtExit

*The information on this page refers to versions prior to 4.6, which removed the LMX\_AtExit function.*

The LMX\_AtExit function registers functions that are to be called at program termination.

## Prototype

```
LMX_STATUS LMX_AtExit
(
    AtExit_pfn Func
);
```

## Parameters

### Func

[in] Function to be called at cleanup.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

LMX\_AtExit is used to register cleanup functions that access the LM-X API.

**Note:** Cstdlib's atexit(), WinApi's \_onexit(), and static object destructors are not supported and will result in undefined behavior.

# Web-based UI

*The information on this page refers to versions prior to 4.8, which replaced web-based UI with LM-X License Server Client.*

LM-X License Manager includes a web-based UI that helps you to monitor and manage your LM-X License Server. The LM-X web-based UI runs on most popular browsers. For complete system requirements for the web-based UI, please see [System requirements for UIs](#).

Using the web-based UI, you can:

- View information and statistics about the LM-X License Server.
- Restart or shut down the license server.
- View the LM-X server's HostIDs (unique machine values that can be used to lock a license file to a host), such as Ethernet, Hostname, IP Address, etc.
- View license usage statistics for the current license server, as well as for borrow, grace and trial licenses.
- View and edit the configuration file. (A password is required to access the configuration file.)
- View and search for entries in the log file.

# LM-X License Manager Developer Documentation

This documentation for LM-X License Manager developers includes:

- [LM-X License Manager Quick Start](#): A quick start that helps you get started with LM-X License Manager, including installation instructions and an introduction to integrating LM-X with your application, distributing your software to end users, enabling license activation, and more.
- [LM-X Developers Manual](#): A guide that details the features and usage of LM-X License Manager.

# LM-X License Manager Quick Start

This guide gives you an introduction to protecting your application using LM-X License Manager. The topics included here are described in more detail in the *LM-X End Users Guide* and the *LM-X Developers Manual*, which also includes more advanced topics and the complete API specification.

The steps to complete before your LM-X protected application can be run by end users include:

1. [Download the LM-X SDK distribution.](#)
2. [Install the LM-X SDK.](#)
3. [Compile the LM-X SDK.](#)
4. [Define a license policy by creating an XML license template.](#)
5. [License your first application.](#)
6. [Distribute protected software to your end users.](#)
7. [Activate your license using License Activation Center.](#)

Each of these steps is described in the following sections.

## Download the LM-X SDK distribution

When you purchase LM-X License Manager or request an evaluation, you receive an Activation Key via email.

To download the newest release of the LM-X SDK:

1. Go to the [LM-X License Manager download page](#).
2. Enter your Activation Key obtained from X-Formation into the Activation Key field, and then click **Submit**.
3. Select the LM-X SDK version you are interested in and follow the Download instructions.

**Note:** The file(s) you need to download for your LM-X SDK installation depends on which platform(s) you've purchased support for. You must have a proper license to use the file(s) you download. Be sure to use the installation file that corresponds with the platform on which LM-X is installed. Failing to obtain a proper license will result in runtime errors when compiling the SDK.

See [Supported platforms](#) for more platform-specific information.



## Install the LM-X SDK

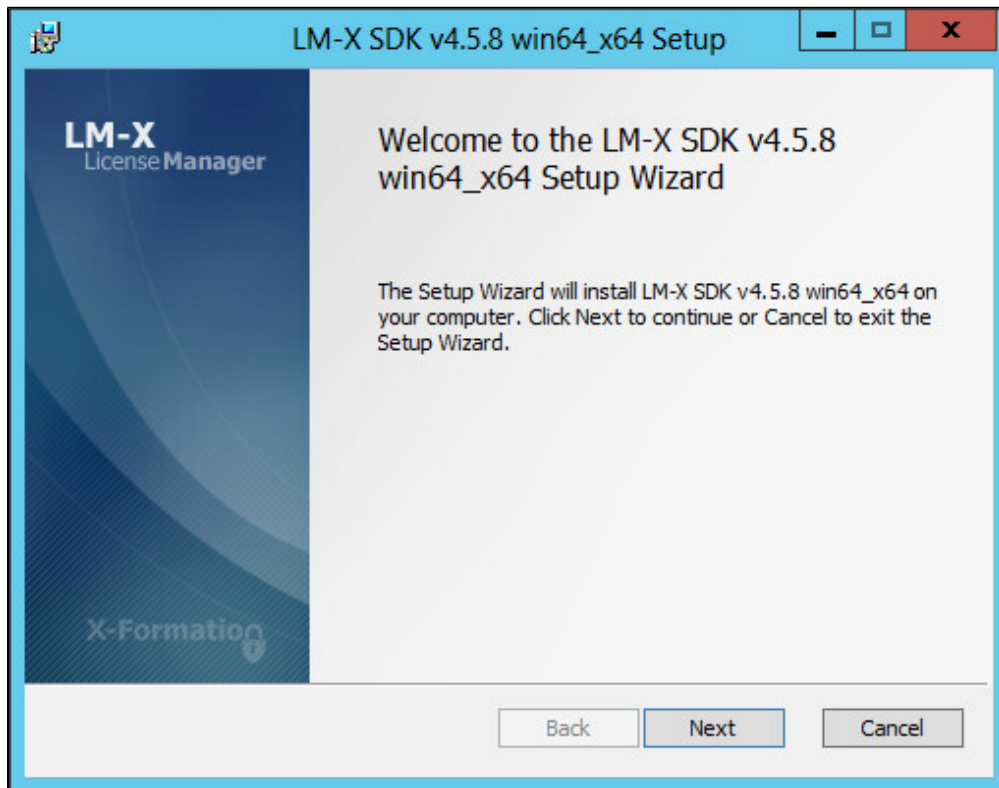
After the download is complete, you can install the LM-X SDK on [Windows](#) or [Unix](#).

## Install the LM-X SDK on Windows

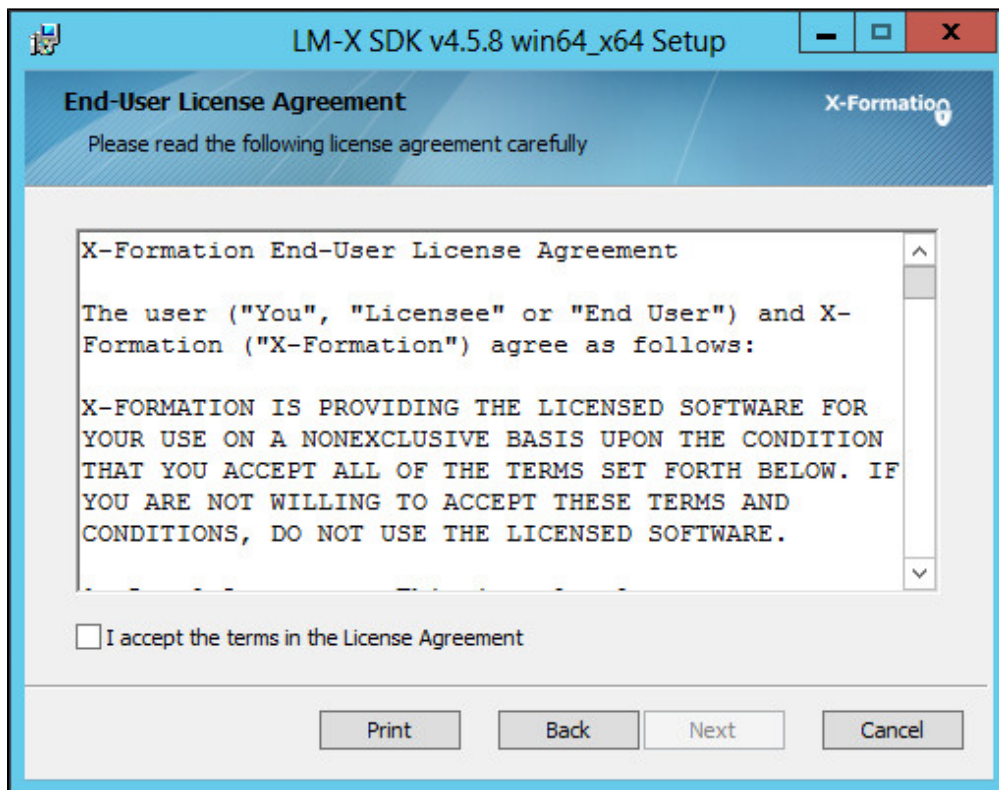
The LM-X License Manager installation wizard guides you through the installation procedure, so you can be up and running on a Windows machine in 5 minutes or less.

To install LM-X on a Windows machine:

Step 1. Start the installation wizard.



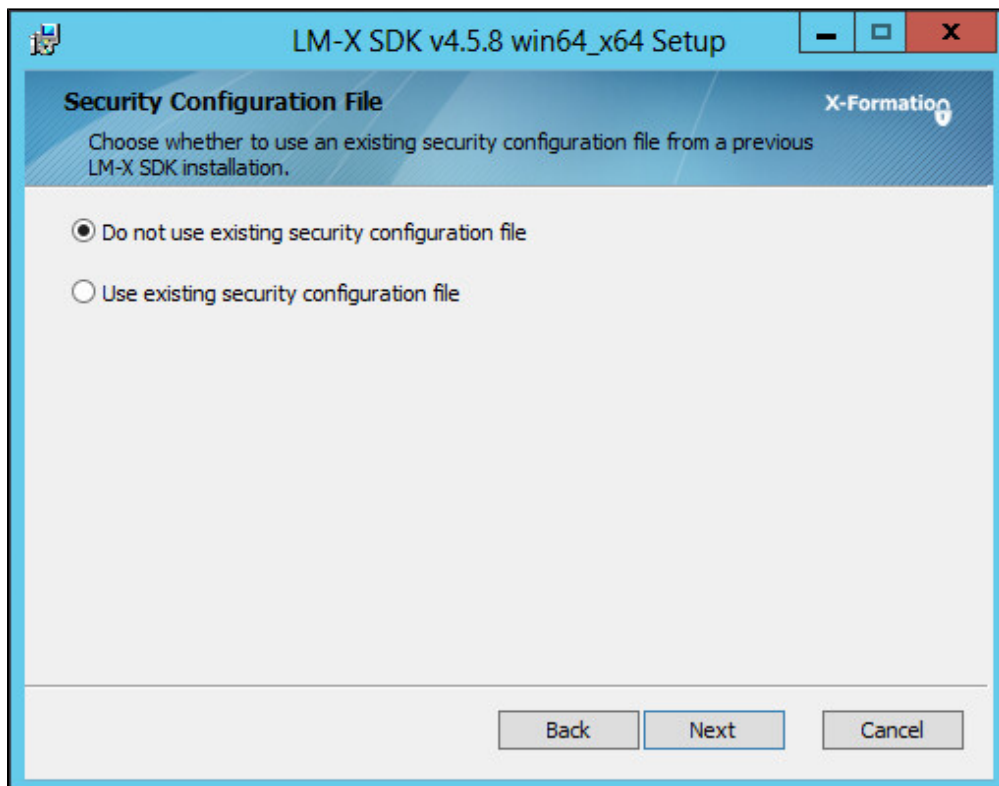
Step 2. Accept the X-Formation End User License Agreement.



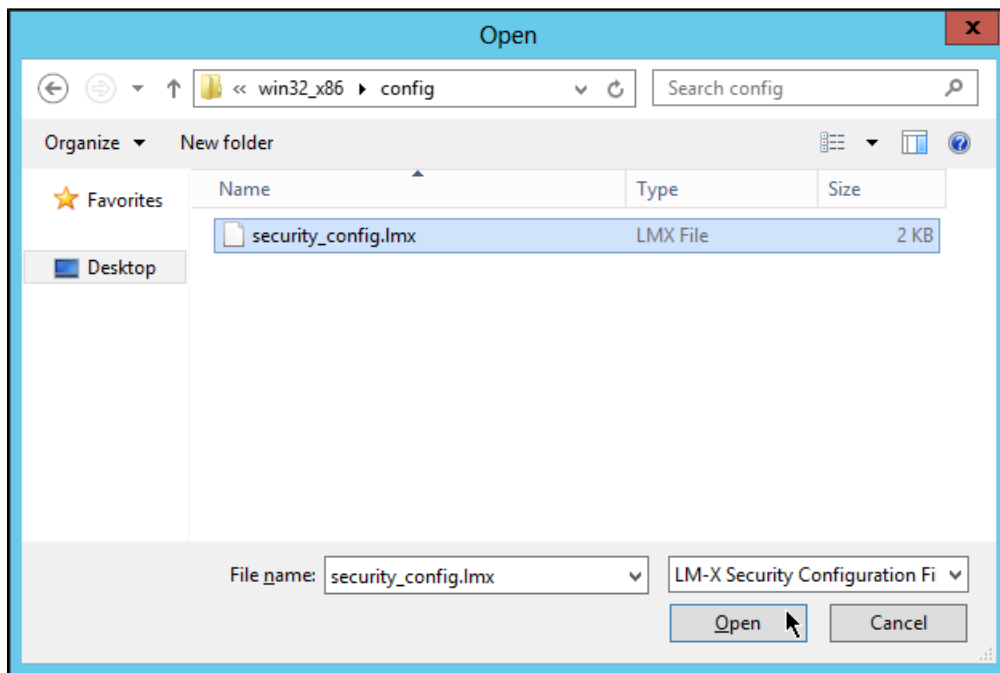
Step 3. Choose whether to use an existing LM-X security configuration file.

Note the following:

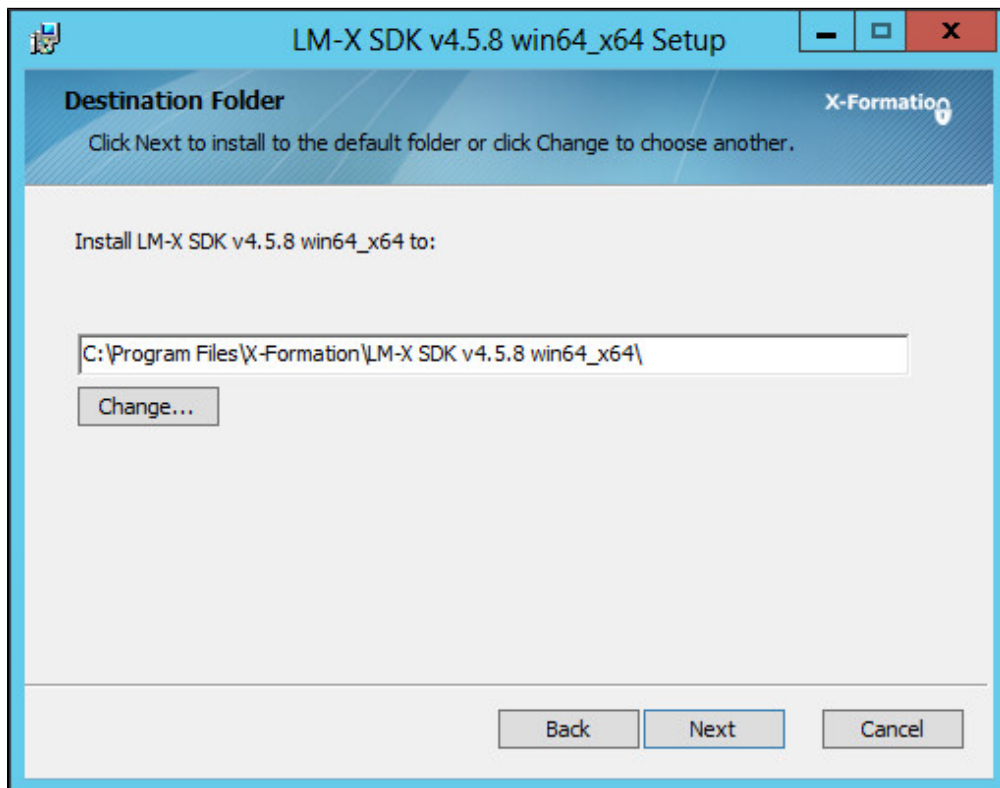
- If you are installing LM-X for the first time: Choose the first option.
- If you are upgrading LM-X to a newer version or compiling LM-X on multiple platforms: Choose an existing [LM-X security configuration file](#).



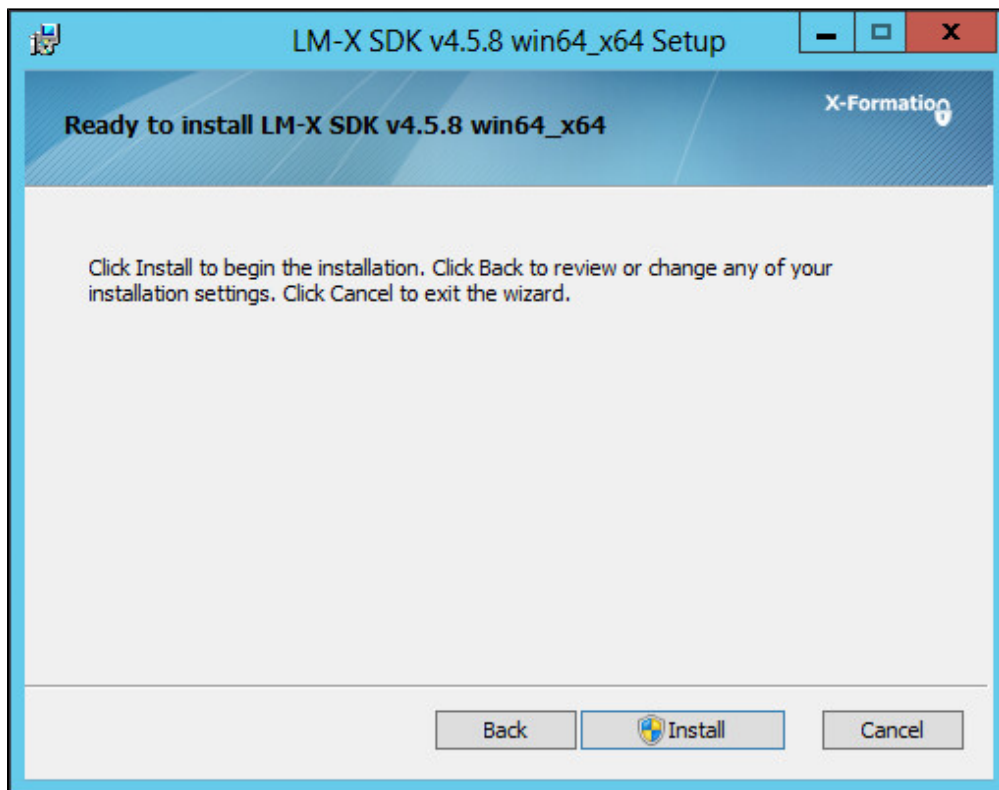
If you have chosen the second option above, use an existing security\_config.lmx file from a previous LM-X installation, as shown below.



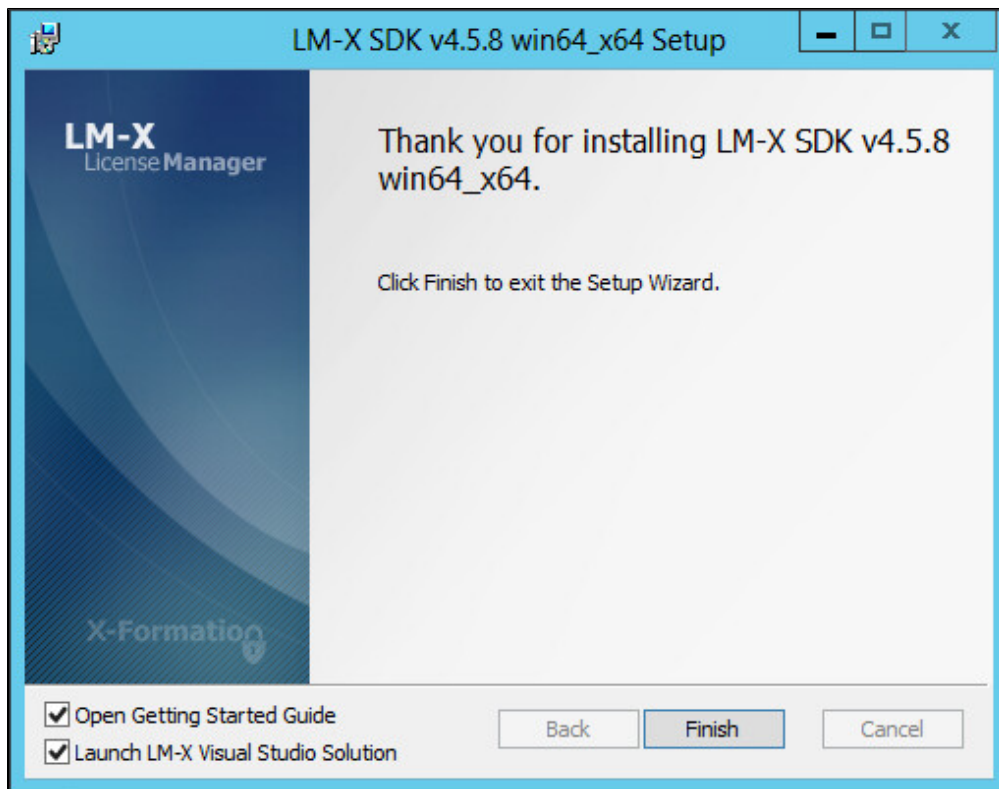
Step 4. Choose the proper installation directory and click Next.



Step 5. Click Install to begin the installation.



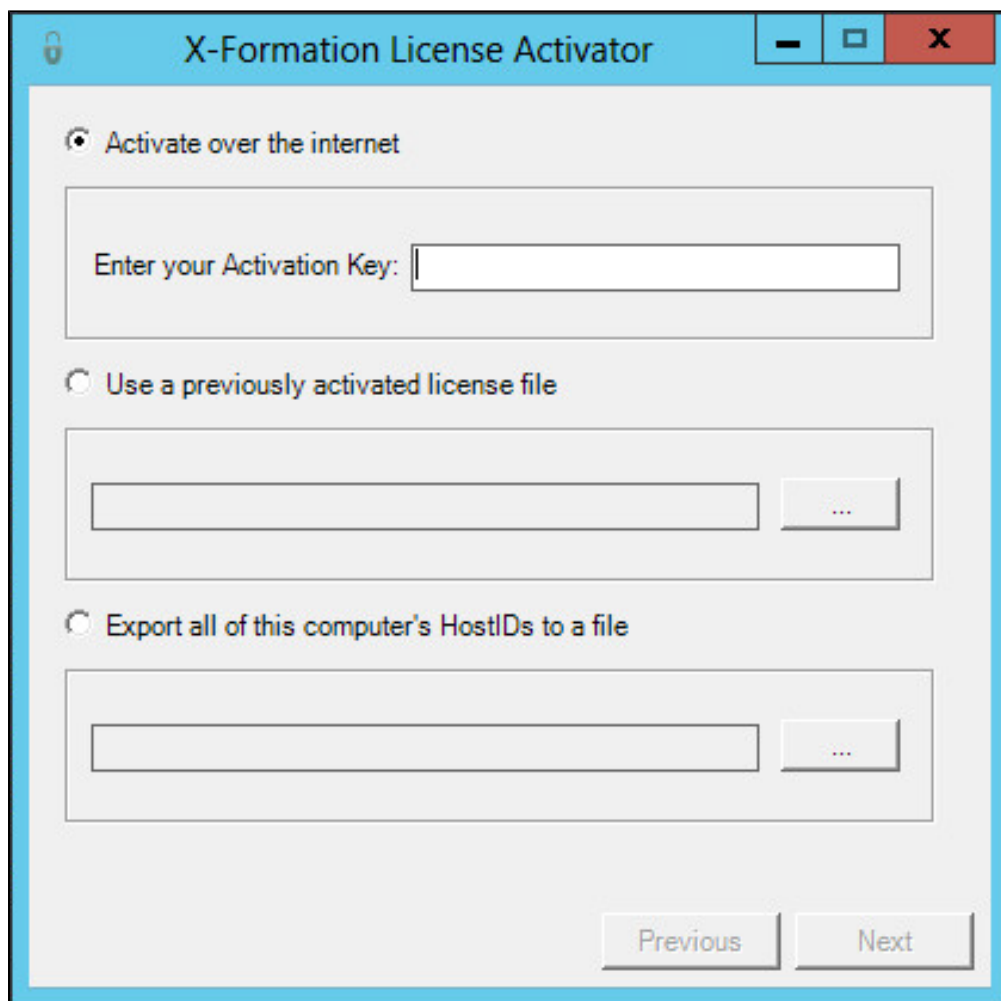
Step 6. Click Finish to finish the installation.



Step 7. Activate the license.

Note the following:

- If you are installing LM-X for the first time: Enter your Activation Key (sent to you via email).
- If you are upgrading LM-X: Select the option to browse for and select a previously activated license file.



The image shows a Windows-style application window titled "X-Formation License Activator". The window has a blue title bar with standard minimize, maximize, and close buttons. Inside the window, there are three radio button options for activation. The first option, "Activate over the internet", is selected. Below it is a text input field with the label "Enter your Activation Key:". The second option is "Use a previously activated license file", which has a text input field and a file selection button (three dots). The third option is "Export all of this computer's HostIDs to a file", which also has a text input field and a file selection button. At the bottom right of the window are two buttons labeled "Previous" and "Next".

**X-Formation License Activator**

☒ Activate over the internet

Enter your Activation Key:

☐ Use a previously activated license file

...

☐ Export all of this computer's HostIDs to a file

...

Previous Next

# Install the LM-X SDK on Unix

The following steps are designed to get LM-X License Manager up and running on a Unix machine in 5 minutes or less.

To install LM-X on a Unix machine:

Step 1. Run the LM-X SDK installer from your terminal.

```
$ sh lmx-sdk_v4.5.8_linux_x64.sh

Verifying archive integrity... All good.

Uncompressing LM-X License Manager SDK 100%

LM-X License Manager SDK 4.5.8 installer.

Copyright (C) 2002-2014 X-Formation. All rights reserved.
```

Step 2. Accept the X-Formation End User License Agreement.

```
-- You must accept the terms of the End User License Agreement (EULA) before installing and using LM-X License
Manager SDK.

-> X-Formation EULA [REJECT/accept/display]: accept
```

Step 3. Specify the proper installation directory.

```
-> Enter installation directory [/usr/lmx-sdk-4.5.8]: /home/john/lmx-sdk-4.5.8
```

Note the following:

- If you are installing LM-X License Manager for the first time: [The LM-X security configuration file](#) will be automatically created in the config directory of the SDK.
- If you are upgrading LM-X to a newer version or compiling LM-X on multiple platforms: Choose an existing LM-X security configuration file, as shown below.

```
-- If you have an existing LM-X security configuration file, you can specify it now.

-- If you are using LM-X License Manager SDK for the first time, specify 'n' (No).

-- The security configuration file will automatically be generated upon SDK compilation.

-> Do you want to use an existing security_config.lmx file? [Y/n]: Y

-- The security configuration file will be copied from the chosen path to /home/john/lmx-sdk-4.5.8/config
directory.

-> Enter path to search for security_config.lmx: [/home/john]: /home/john

Searching for security_config.lmx in /home/john... done.

[0] /home/john/lmx-sdk-4.5.7/config/security_config.lmx

[1] /home/john/lmx-sdk-4.5.6/config/security_config.lmx

-> Enter a number corresponding to security_config.lmx file path in the above list [0/1]: 0
```

Step 4. (MacOS only) Provide Developer ID Application Certificate.

Note the following:

- The Developer ID Application Certificate will be assigned to the `DEVELOPER_ID_CERTIFICATE` variable in the `platform.mk` file.
- If you did not provide a certificate during the installation process or you want to change the certificate, edit the `platform.mk` file and assign the correct certificate to the `DEVELOPER_ID_CERTIFICATE` variable, then recompile the LM-X SDK.
- See [MacOS on ARM support](#) for more information about providing a Developer ID Application Certificate.

```
-- If you have a Developer ID Application Certificate, you can provide it now.
-- It will be used to sign executables and libraries that are built during the installation process.
-> Do you want to provide Developer ID Application Certificate [Y/n]: y
-> Developer ID Application Certificate: DEVELOPER_ID_CERTIFICATE
```

## Step 5. Activate the license.

Note the following:

- If you are installing LM-X License Manager for the first time: Activate your license online using your Activation Key (sent to you via email).
- If you are upgrading LM-X: Select a previously activated license file.

```
-- You must have a valid license file to use LM-X License Manager SDK.
-- You can either use an existing license file (*.lic) or you can obtain a new license file by activating LM-X
License Manager SDK over the internet.
-- Please visit https://license.x-formation.com for details.
-> Do you want to activate your license over the Internet? [Y/n]: n
-> Do you want to choose an existing license file? [Y/n]: Y
-- The LM-X License Manager SDK license file will be copied from chosen path to /config directory.
-> Enter path to search for lmx.lic: [/home/john]: /home/john
-- Searching for lmx.lic in /home/john... done.
[0] /home/john/lmx-sdk-4.5.7/lmx.lic
[1] /home/john/lmx-sdk-4.5.6/lmx.lic
-> Enter a number corresponding to lmx.lic file path in the above list [0/1]: 0
-> Do you want to extract files to /home/john/lmx-sdk-4.5.8? [Y/n]: Y
-- Copying files... done.
-- Installation of LM-X License Manager SDK 4.5.8 completed successfully.
```

## Step 6. Decide whether you want to compile the LM-X SDK.

**Note:** If you used a previously activated license file in Step 4, [compile the LM-X SDK](#) now.

```
-> Do you want to compile your LM-X License Manager SDK now? [Y/n]: n
-- For detailed installation log see /tmp/lmx_sdk_installation.log.
-- To get started with LM-X License Manager SDK please see Getting started.
```

## Step 7. When needed, install the end-user tools.

**Note (MacOS only):** If a Developer ID Application Certificate was provided in Step 4, it will be used to sign the LM-X End-user Utility and LM-X License Server.

```
-> Do you want install LM-X Enduser Tools 4.5.8 now? [Y/n]: Y
```

(See [End-user tools](#) for more details.)



## Compile the LM-X SDK

After the installation is complete, you can compile the LM-X SDK on [Windows](#) or [Unix](#). Additionally, you can [extend or modify the behavior of the license server](#) during precompilation using [lmx\\_server\\_conf.c](#) file.

# Compile the LM-X SDK on Windows

*The information on this page refers to LM-X License Manager v4.7 and newer, which eliminated the need to use a separate mingw32 installer to compile the LM-X SDK using MinGW.*

The following steps are designed to compile LM-X SDK on a Windows machine in 5 minutes or less from a command line, using Visual Studio, and MinGW.

**Note:** By default, when you install the SDK, Visual Studio opens to let you compile the SDK. We recommend that you use the project files and use the Build, Clean and Rebuild UI actions within the IDE to compile and recompile the SDK. Alternatively, you can compile the SDK from a command line using nmake, as described in "Compiling the LM-X SDK from a command line", below.

When compiling the LM-X SDK under Windows, one of the following is required:

- Visual Studio (Express, Standard, Pro or better).
- MinGW

## Compiling the LM-X SDK using Visual Studio

To compile the LM-X SDK using Visual Studio:

Step 1. Copy your LM-X-SDK Imx.lic file to the config directory and specify a desired OPTION for the vendor.

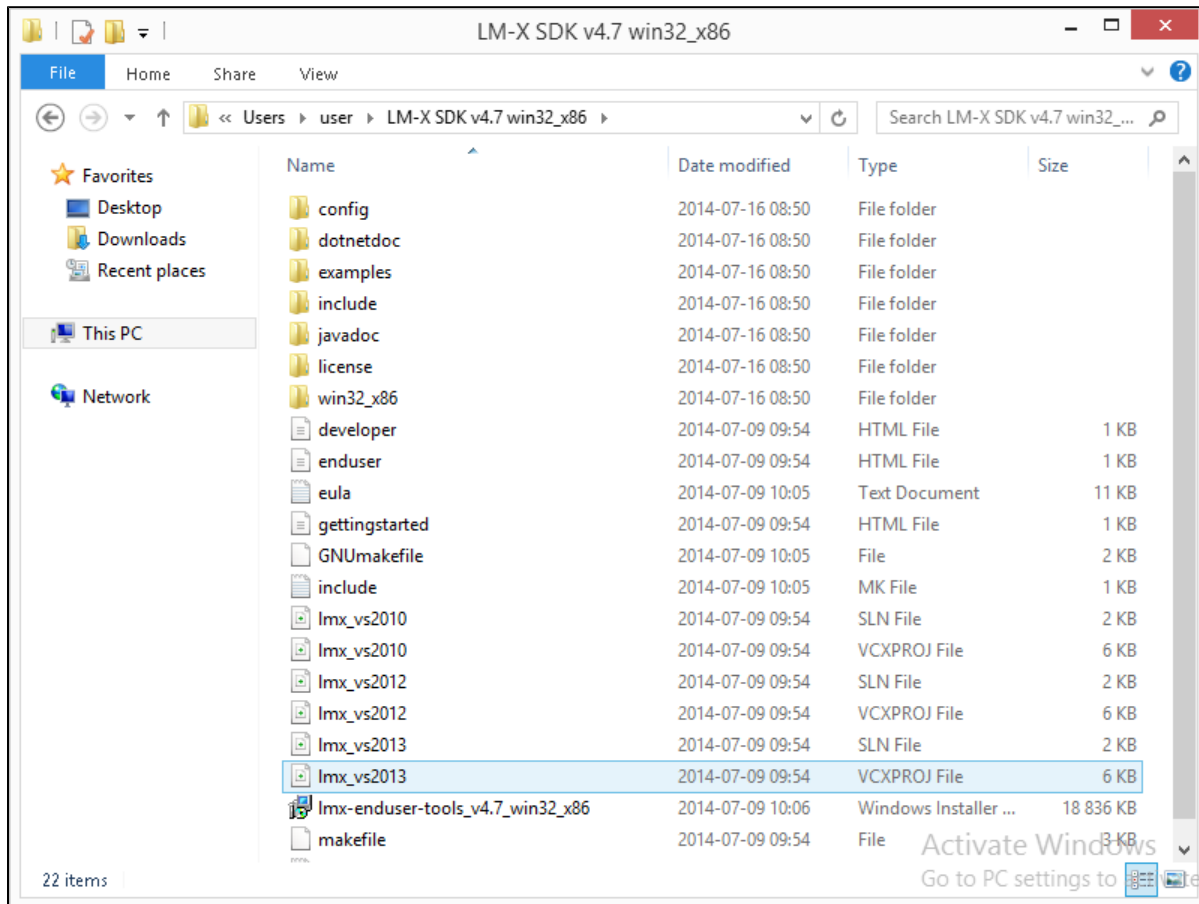
You can extend or modify the behavior of the license server during pre-compilation by editing the [Imx\\_server\\_conf.c](#) file.

**Note:** If you are [upgrading LM-X](#), remember to copy your [LM-X security configuration file](#) from a previous LM-X installation to the config directory. (In LM-X SDK versions older than 4.2, security\_config.Imx file was named after your vendorname.Imx.)

Step 2. Run Visual Studio.

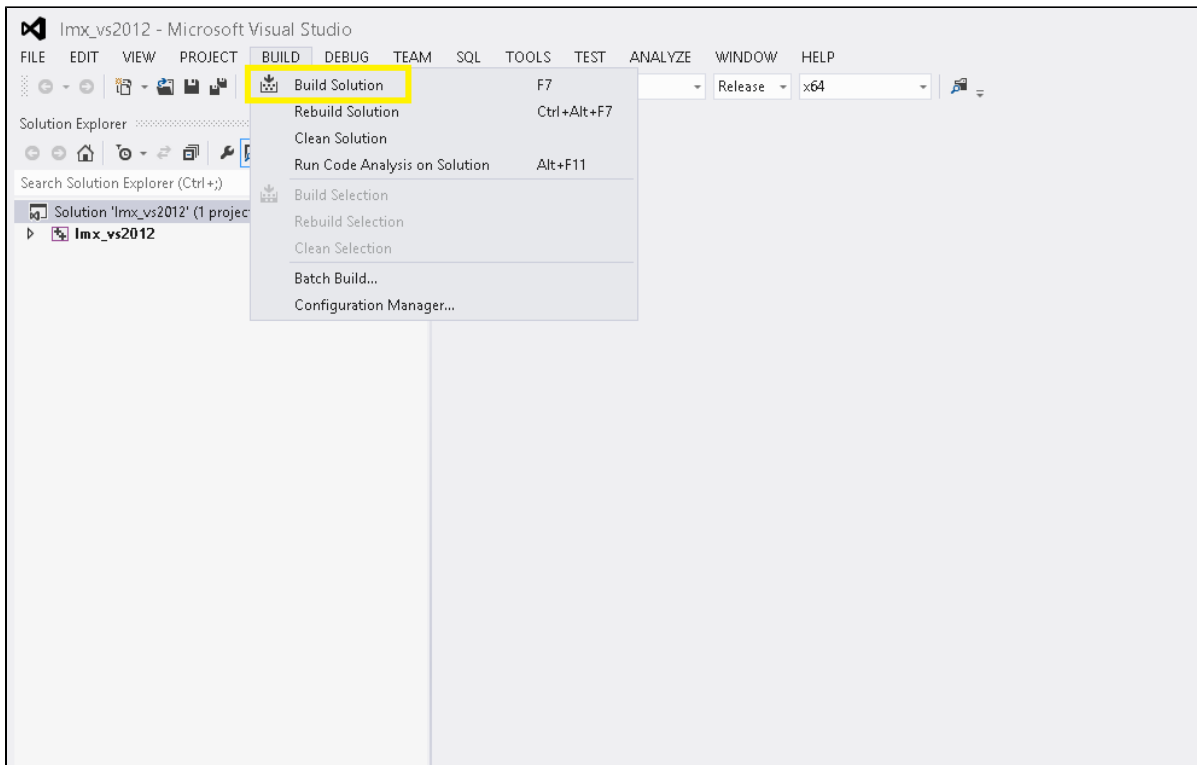
To run Visual Studio, either:

- Run Visual Studio by default by clicking **Finish** at the end of [installing the LM-X SDK on Windows](#)  
or
- Double-click on the proper Visual Studio solution file, as shown below:

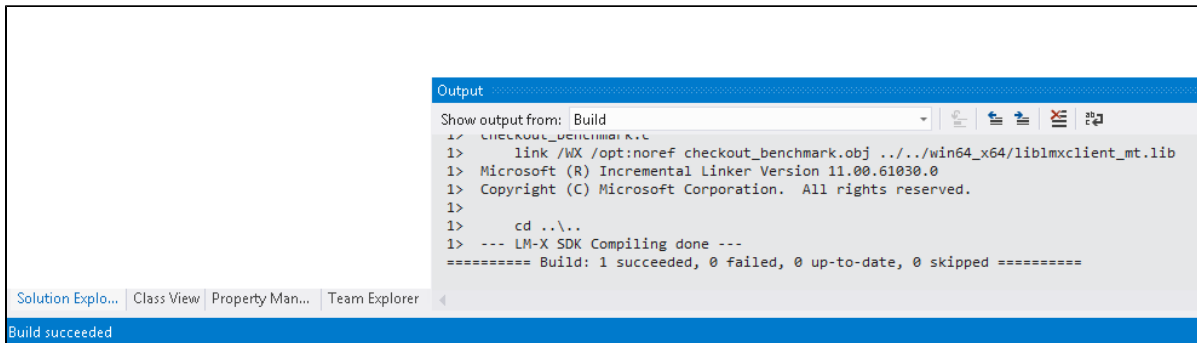


**Note:** Make sure the config directory includes the [license file](#).

Step 3. Click Build tab and from the list of options that appears, select "Build Solution".



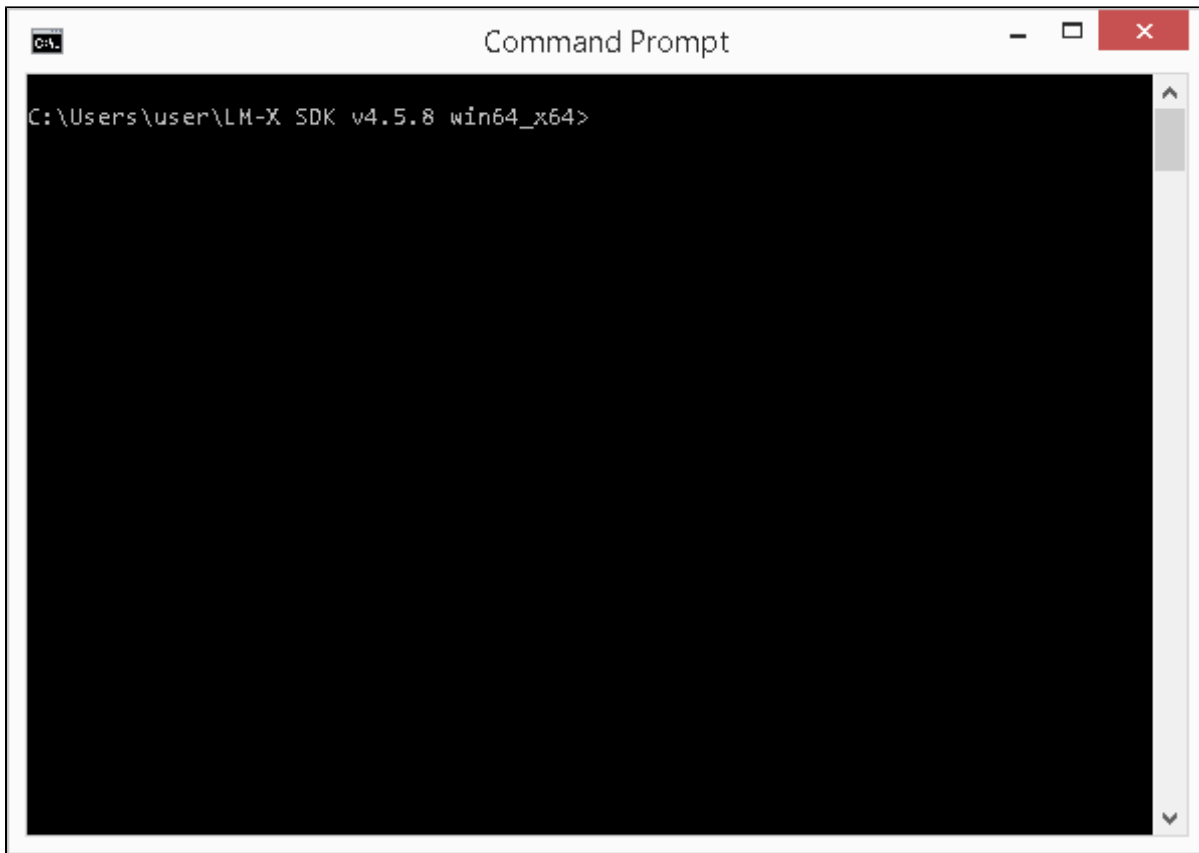
When the compilation is successfully completed, you will see a "Build succeeded" message on the Visual Studio status bar, as shown below.



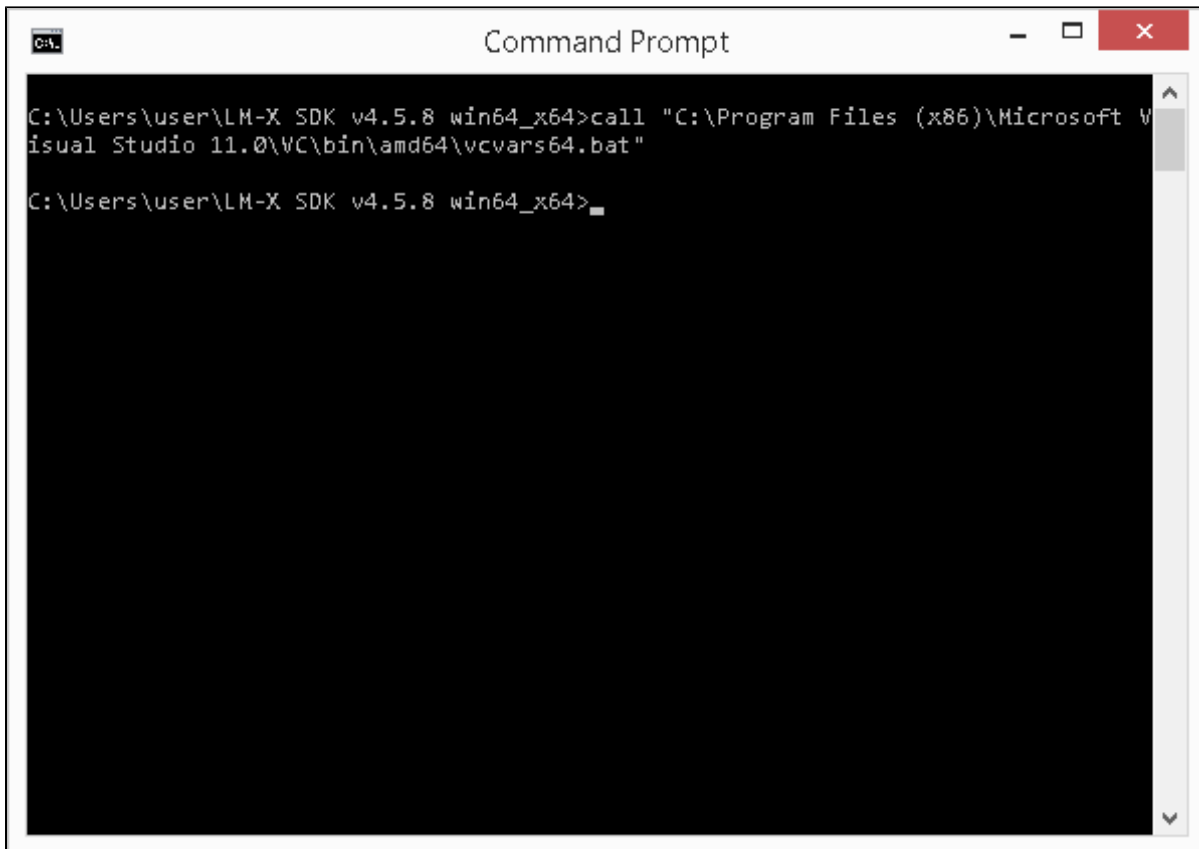
## Compiling the LM-X SDK from a command line using Visual Studio

To compile the LM-X SDK from a command line using Visual Studio:

Step 1. Run cmd.exe.



Step 2. Set the environment variable.



The following are examples of environment variables:

#### **Visual Studio 2010 32-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 10.0\VC\bin\vcvars32.bat
```

#### **Visual Studio 2012 32-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 11.0\VC\bin\vcvars32.bat
```

#### **Visual Studio 2013 32-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 12.0\VC\bin\vcvars32.bat
```

#### **Visual Studio 2010 64-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 10.0  
\VC\bin\amd64\vcvars64.bat
```

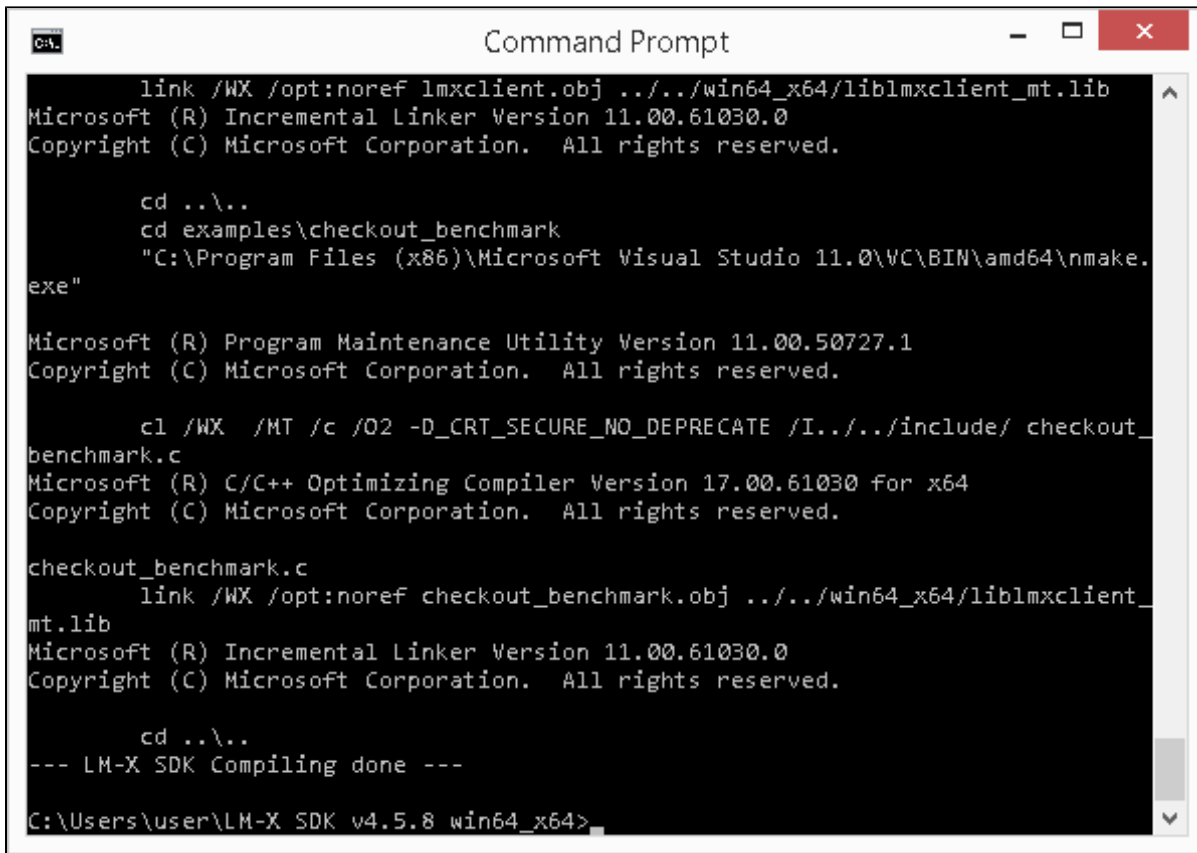
#### **Visual Studio 2012 64-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 11.0  
\VC\bin\amd64\vcvars64.bat
```

#### **Visual Studio 2013 64-bit**

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> call %PROGRAM_FILES%\Microsoft Visual Studio 12.0  
\VC\bin\amd64\vcvars64.bat
```

Step 3. From the root directory of the LM-X distribution, run nmake.



```

link /WX /opt:noref lmtxclient.obj ../../win64_x64/liblmtxclient_mt.lib
Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation. All rights reserved.

cd ../../
cd examples\checkout_benchmark
"C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\BIN\amd64\nmake.
exe"

Microsoft (R) Program Maintenance Utility Version 11.00.50727.1
Copyright (C) Microsoft Corporation. All rights reserved.

cl /WX /MT /c /O2 -D_CRT_SECURE_NO_DEPRECATED /I../../include/ checkout_
benchmark.c
Microsoft (R) C/C++ Optimizing Compiler Version 17.00.61030 for x64
Copyright (C) Microsoft Corporation. All rights reserved.

checkout_benchmark.c
link /WX /opt:noref checkout_benchmark.obj ../../win64_x64/liblmtxclient_
mt.lib
Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation. All rights reserved.

cd ../../
--- LM-X SDK Compiling done ---
C:\Users\user\LM-X SDK v4.5.8 win64_x64>

```

**Note:** You must run nmake from the SDK root directory. Running nmake from a subdirectory may produce error messages and fail.

## Cleaning the LM-X SDK using Visual Studio

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> nmake clean
```

**Note:** You may want to clean previously compiled files when rebuilding the SDK with a different license or security key.

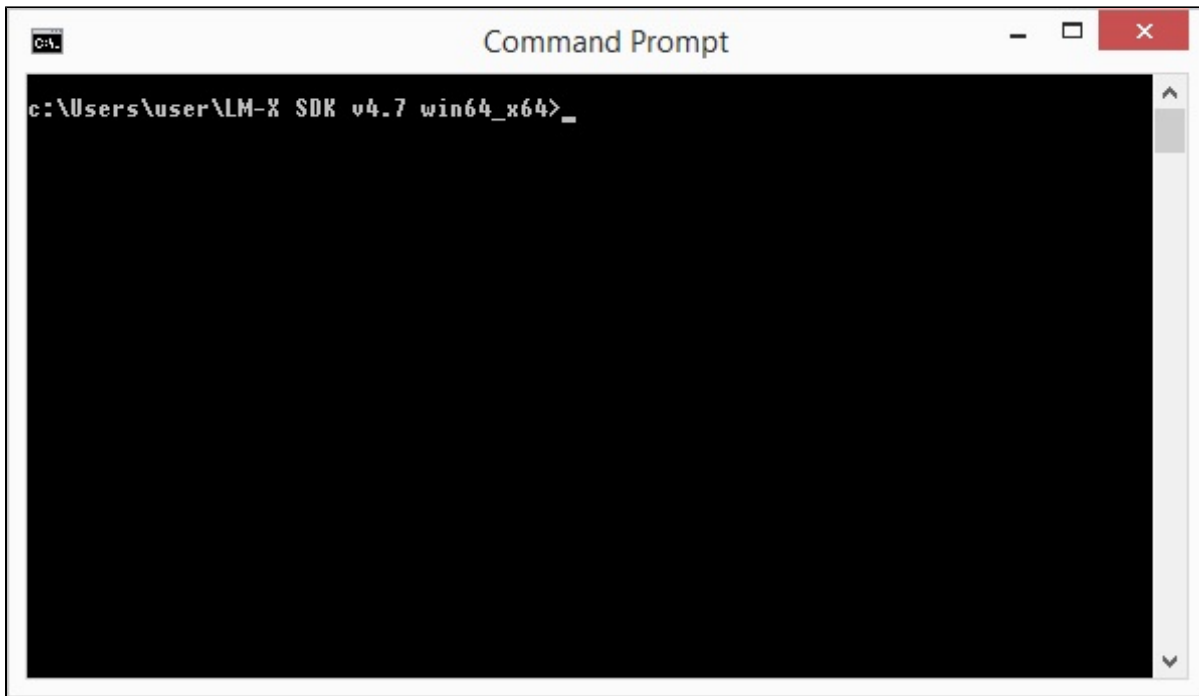
See [Installation issues](#) for information about problems and workarounds related to compiling the LM-X SDK.

## Compiling the LM-X SDK using MinGW

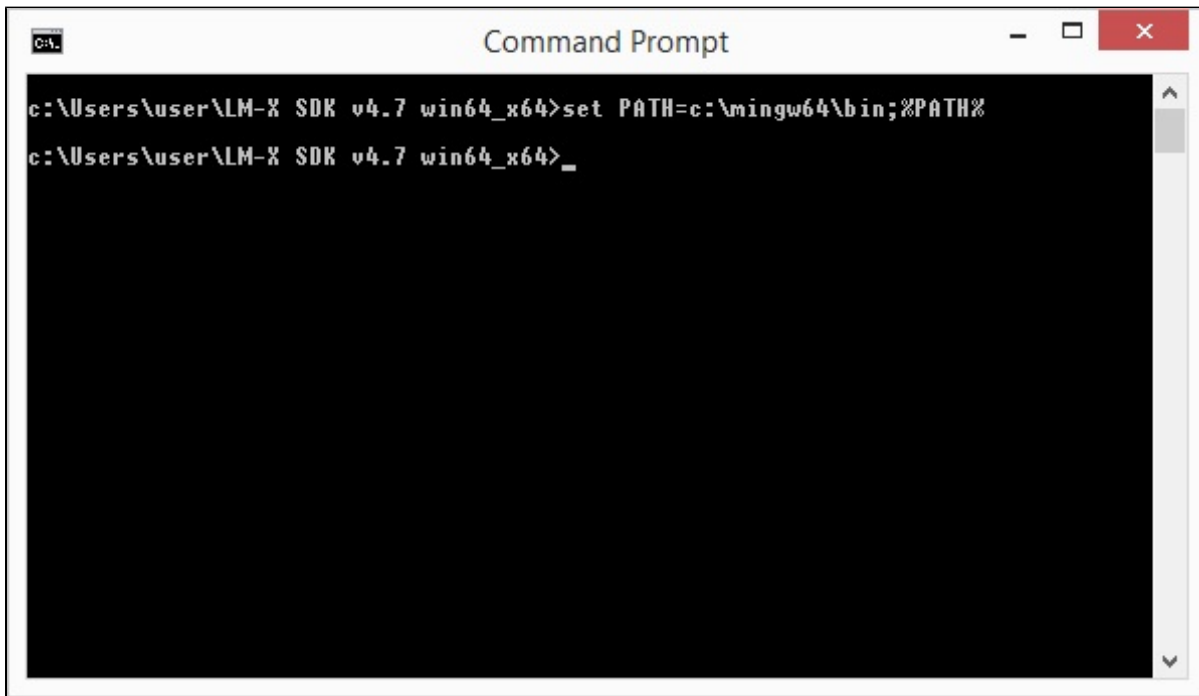
When compiling the LM-X SDK using MinGW, make sure [Cygwin](#) is installed on your machine. Also, please consider MinGW-specific requirements and limitations, as described in [Supported platforms](#).

To compile the LM-X SDK using MinGW:

Step 1. Run cmd.exe.

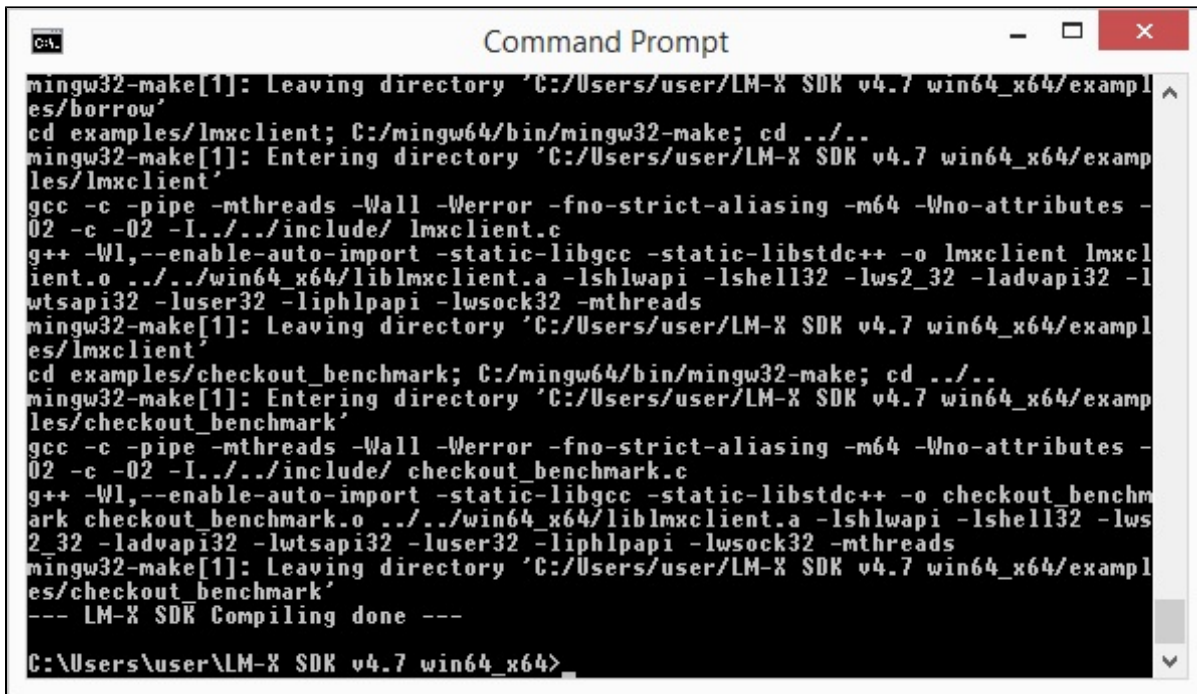


Step 2. Set the environment variable.



**Note:** The path for mingw32-make may vary depending on the machine and architecture being used.

Step 3. From the root directory of the LM-X distribution, run mingw32-make.



```

mingw32-make[1]: Leaving directory 'C:/Users/user/LM-X SDK v4.7 win64_x64/examp
es/borrow'
cd examples/lmxclient; C:/mingw64/bin/mingw32-make; cd ../..
mingw32-make[1]: Entering directory 'C:/Users/user/LM-X SDK v4.7 win64_x64/examp
les/lmxclient'
gcc -c -pipe -mthreads -Wall -Werror -fno-strict-aliasing -m64 -Wno-attributes -
O2 -c -O2 -I../include/ lmxclient.c
g++ -Wl,--enable-auto-import -static-libgcc -static-libstdc++ -o lmxclient lmxcl
ient.o ../win64_x64/liblmxclient.a -lshlwapi -lshell32 -lws2_32 -ladvapi32 -l
wtsapi32 -luser32 -liphlpapi -lwsock32 -mthreads
mingw32-make[1]: Leaving directory 'C:/Users/user/LM-X SDK v4.7 win64_x64/examp
les/lmxclient'
cd examples/checkout_benchmark; C:/mingw64/bin/mingw32-make; cd ../..
mingw32-make[1]: Entering directory 'C:/Users/user/LM-X SDK v4.7 win64_x64/examp
les/checkout_benchmark'
gcc -c -pipe -mthreads -Wall -Werror -fno-strict-aliasing -m64 -Wno-attributes -
O2 -c -O2 -I../include/ checkout_benchmark.c
g++ -Wl,--enable-auto-import -static-libgcc -static-libstdc++ -o checkout_benchm
ark checkout_benchmark.o ../win64_x64/liblmxclient.a -lshlwapi -lshell32 -lws
2_32 -ladvapi32 -lwsapi32 -luser32 -liphlpapi -lwsock32 -mthreads
mingw32-make[1]: Leaving directory 'C:/Users/user/LM-X SDK v4.7 win64_x64/examp
les/checkout_benchmark'
--- LM-X SDK Compiling done ---
C:\Users\user\LM-X SDK v4.7 win64_x64>

```

#### Cleaning the LM-X SDK using MinGW

```
C:\Users\user\LM-X SDK v4.5.8 win64_x64> mingw32-make clean
```

**Note:** You may want to clean previously compiled files when rebuilding the SDK with a different license or security key.

See [Installation issues](#) for information about problems and workarounds related to compiling the LM-X SDK.



# Compile the LM-X SDK on Unix

The following steps are designed to compile LM-X SDK on a Unix machine in 5 minutes or less from a command line.

**Note:** When compiling the LM-X SDK under Unix, GCC and make are required.

## Compiling the LM-X SDK from a command line

To compile the LM-X SDK from a command line:

Step 1. Copy your LM-X-SDK lmx.lic file to the config directory and specify a desired OPTION for the vendor.

You can extend or modify the behavior of the license server during pre-compilation by editing the [lmx\\_server\\_conf.c](#) file.

**Note:** If you are [upgrading LM-X](#), remember about copying your [LM-X security configuration file](#) from a previous LM-X installation to the config directory. (In LM-X SDK versions older than 4.2, security\_config.lmx file was named after your vendorname.lmx.)

Step 2. Run make.

```
/usr/local/lmx-sdk-4.6.1 $ make
```

**Note:** You can clean previously compiled files (for example, when rebuilding the SDK with a different license or security key).

Also note that you must run make from the SDK root directory. Running make from a subdirectory may produce error messages and fail.

```
/usr/local/lmx-sdk-4.6.1 $ make clean
```

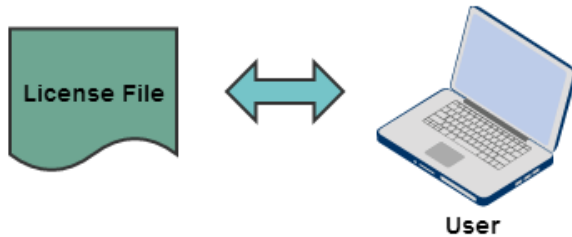
See [Installation issues](#) for information about problems and workarounds related to compiling the LM-X SDK.

## Define a license policy

Establish a license policy that suits your business. LM-X is configurable for a large variety of different license schemes, which usually consists of one or more features in your software. If your application has more than one differently licensed module, the license file will contain multiple features.

### Node-locked licenses

Designed for individual users/machines, [node-locked licenses](#) allow a single instance of an application to run on a specified host. LM-X lets you [determine which HostID type to use](#) to specify the machine you want to lock your licensed application to. For example, you can use [an Ethernet card](#), [BIOS](#), [Hasp HL dongle](#) or [harddisk HostID](#).

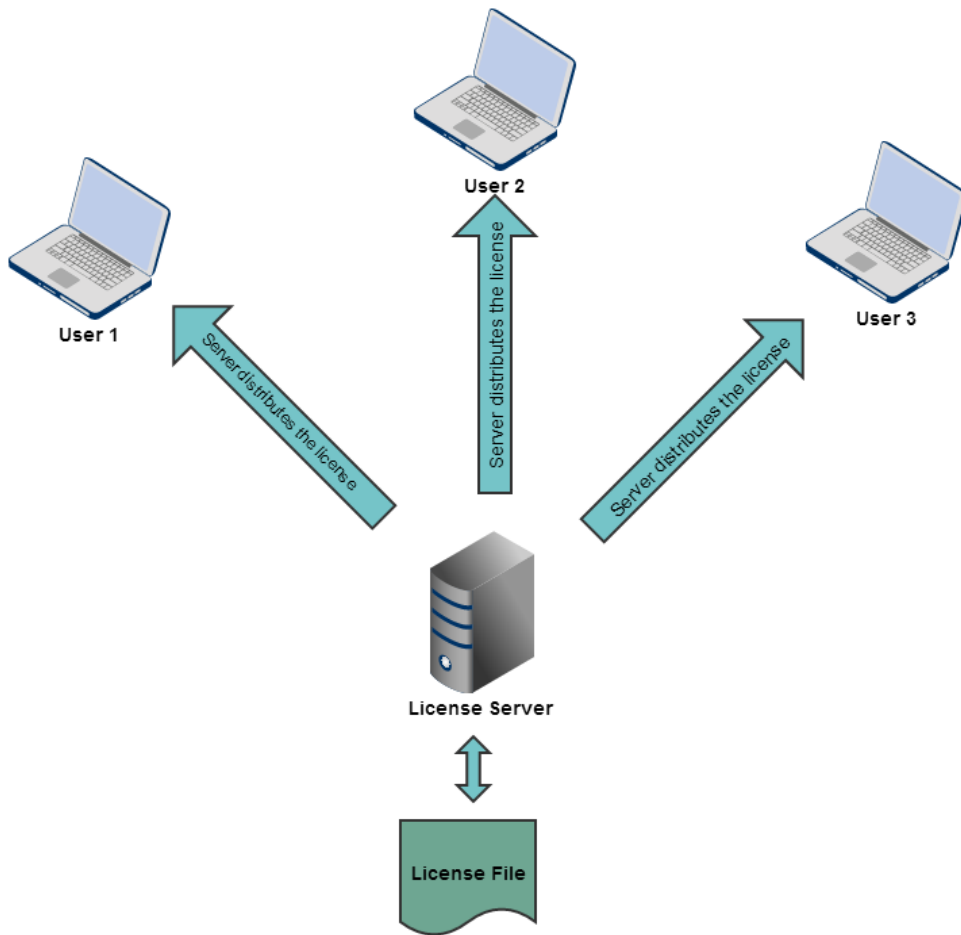


### Floating/network licenses

[Floating/network licenses](#) allow multiple users to install and use a limited number of licenses on multiple machines.

#### Example

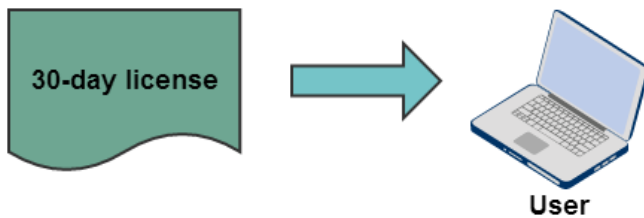
Suppose you have 10 workstations where Program X is used. In most cases there are fewer than 10 users concurrently using the application. Instead of purchasing 10 licenses, you can only purchase the number of floating licenses to support the number of concurrent users. Some highlights of the benefits of floating licenses include centralized license management and maximum utilization of each license.



[Determining whether to use node-locked or floating licenses](#) depends on your end user's needs and how you prefer to distribute your software.

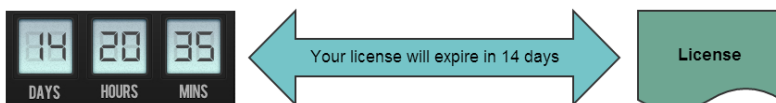
## Trial licenses

Trial licenses allow users to evaluate your software before purchase.



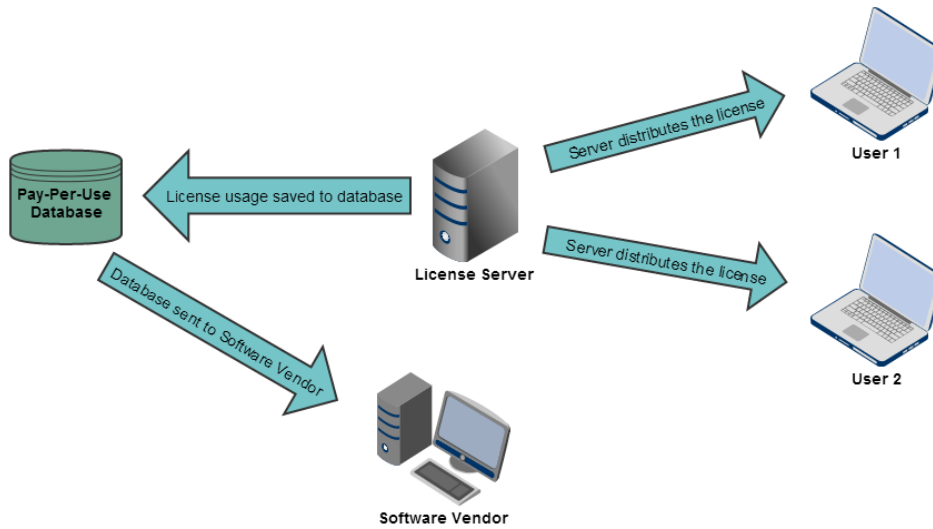
## Time-limited licenses

Time-limited licenses are used for restricting usage periods and for subscription-based licensing models.



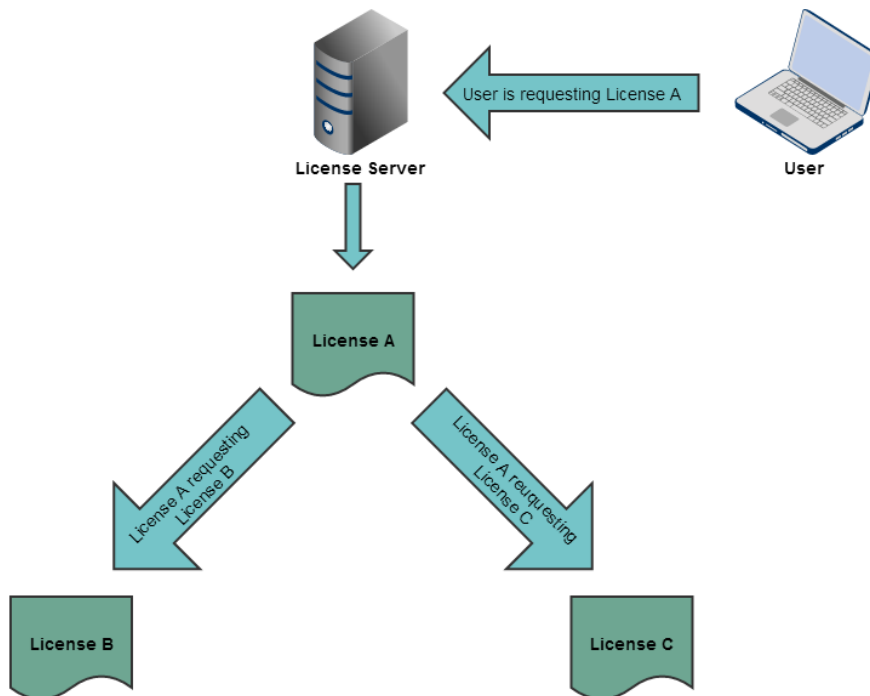
## Pay-per use licenses

Pay-per-use licenses are used for back-billing customers based on their actual usage.



## Token-based

Token-based licensing is used for bundling products in diverse ways, such as creating license pools and product suites.



# Example XML license template scenarios

Find the scenario(s) you are most interested in below, and use the example to help you create your own XML license template.

## Trial license XML template scenario

For this trial license scenario, let's assume you want your application to be:

- An evaluation for selected features that will work for a specified number of days or specified number of uses.
- Widely available, with no need for a license. (Trials do not require a license file and can be enabled by default when there is no license file found.)
- Easily converted to a licensed version by delivering a license file to the user.
- Prevented from being used after the trial period expires (the user is able to evaluate only once, and must then purchase a license to continue running the product).
- Available for re-evaluation when a new version of the software is released.
- Forbidden to run on virtual machines (by default).
- Protected against clock tampering (by default).

No XML license template is needed for trial licenses, and you only need to specify the length of the trial.

## Node-locked license XML template scenario

For this node-locked license scenario, let's assume you want your application to be:

- Subscription-based, paid annually.
- Locked to a particular machine (that is, a specific HostID).
- Forbidden to run on virtual machines (by default).
- Protected against clock tampering (by default).

In the following example we assume the license will expire on January 1, 2018. This type of license may require that the [user's HostID\(s\)](#) be defined as shown below.

```
<LICENSEFILE>
  <FEATURE NAME="MyFeatureName">
    <SETTING MAJOR_VERSION="1" />
    <SETTING MINOR_VERSION="0" />
    <SETTING END="2018-01-01" />
    <CLIENT_HOSTID>
      <SETTING IPADDRESS="123.123.123.123" />
      <SETTING USERNAME="john" />
    </CLIENT_HOSTID>
  </FEATURE>
</LICENSEFILE>
```

## Floating license XML template scenario

For this floating license scenario, let's assume you want your application (or particular application features) to be:

- Used only by authorized end users on a network. The number of concurrent users is counted.
- Subscription-based, paid annually.
- Forbidden to run on virtual machines (by default).
- Protected against clock tampering (by default).

In the following example we assume the license will expire on January 1, 2018. This type of license limits the number of concurrent users of the license to five.

```
<LICENSEFILE>
  <FEATURE NAME="MyFeatureName">
    <SETTING MAJOR_VERSION="1" />
    <SETTING MINOR_VERSION="0" />
    <SETTING COUNT="5" />
    <SETTING END="2018-01-01" />
  </FEATURE>
</LICENSEFILE>
```

# License your first application

Now that you have defined a license policy, you need a license template to integrate LM-X with your application.

## License file

To generate an XML license template:

1. Create a template of the license you are interested in.
2. Run the license generator provided with LM-X, `xmllicgen`, as described in [Generating licenses](#).

In the following example we assume the license will expire on January 1, 2018. This license defines one feature with version 1.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<LICENSEFILE>
  <FEATURE NAME="feature">
    <SETTING MAJOR_VERSION="1" />
    <SETTING MINOR_VERSION="0" />
    <SETTING END="2018-01-01" />
  </FEATURE>
</LICENSEFILE>
```

Run `xmllicgen` to convert the above LM-X license template to a license file.

### On Unix:

```
# xmllicgen license.xml
```

### On Windows:

```
# xmllicgen.exe license.xml
```

The resulting file will be `license.lic`. `Xmllicgen` will automatically replace your `.xml` extension with `.lic` and save the license next to your template.

## Source code

### Header file

First of all, you must include the required header, `lmx.h`, located in the include directory of your SDK.

```
#include "lmx.h"
```

If your project and LM-X SDK are in separate directories, remember to specify the include directory for your compiler. For example:

### GCC:

```
gcc -I /usr/lmx-sdk-4.6.1/include/
```

### MSVC:

```
cl.exe /I "C:\Program Files\X-Formation\LM-X SDK v4.6.1 win64_x64\include\"
```

## Commonly used functions

In the following examples we will present five commonly used [LM-X API](#) functions.

### LMX\_Init

The [LMX\\_Init](#) function initializes the protection system. This function creates a handle needed to use other LM-X API functions. It returns `LMX_STATUS` variable that indicates initialization status.

```

LMX_HANDLE LmxHandle;

if (LMX_Init(&LmxHandle) != LMX_SUCCESS)
{
    printf("Unable to initialize!\n");
    return 1;
}

```

### LMX\_SetOption

The [LMX\\_SetOption](#) function sets up flags that change licensing behavior prior to license checkout. This function returns LMX\_STATUS variable that indicates the status of initialization.

In the following example LMX\_SetOption is used to set the license path to the current path.

```

LMX_SetOption(LmxHandle, LMX_OPT_LICENSE_PATH, ".");

```

### LMX\_Checkout

The [LMX\\_Checkout](#) function is one of the most important LMX API functions, because it checks out one or more licenses for a specific feature. This function requires that the feature name, version and the count of the features be defined as shown below.

```

if (LMX_Checkout(LmxHandle, "feature", 1, 0, 1) != LMX_SUCCESS)
{
    printf("Unable to checkout\n");
    LMX_Free(LmxHandle);
    return 1;
}

```

### LMX\_Checkin

The [LMX\\_Checkin](#) function returns the licenses for a single checked out feature or all checked out features.

```

LMX_Checkin(LmxHandle, "feature", LMX_ALL_LICENSES);

```

### LMX\_Free

The [LMX\\_Free](#) function, which has an inverse effect to [LMX\\_Init](#), frees any allocated memory used by the licensing system and closes any open connection to a license server.

```

LMX_Free(LmxHandle);

```

## Making it work

The following example illustrates a complete, compiled example that includes source code necessary to license your application using LM-X License Manager.

```
#include <stdio.h>

#include "lmx.h"

int main()
{
    LMX_HANDLE LmxHandle;

    if (LMX_Init(&LmxHandle) != LMX_SUCCESS)
    {
        printf("Unable to initialize!\n");
        return 1;
    }

    // Look for licenses in current directory.
    LMX_SetOption(LmxHandle, LMX_OPT_LICENSE_PATH, ".");
    if (LMX_Checkout(LmxHandle, "feature", 1, 0, 1) != LMX_SUCCESS)
    {
        printf("Unable to checkout!\n");
        LMX_Free(LmxHandle);
        return 1;
    }
    // Here you are safe to run your licensed features
    printf("Here you can run your features\n");

    LMX_Checkin(LmxHandle, "feature", LMX_ALL_LICENSES);
    LMX_Free(LmxHandle);
    return 0;
}
```

If you decide to copy and paste the above code block, save it as an `example.c`.

Compilation

Let's assume you saved your program source code as an `example.c` and installed your LM-X SDK in the default directory.

To compile your first program run the following:

GCC:

```
gcc -c -pthread -fPIC -Wall -Werror -fno-strict-aliasing -m64 -Wfatal-errors -Wno-unused-local-typedefs -Wno-
vla -Wno-attributes -O2 -c -O2 -I/usr/lmx-sdk-4.6.1/include/ example.c

gcc -static-libgcc -o example example.o /usr/lmx-sdk-4.6.1/linux_x64/liblmxclient.a -pthread -lrt -ldl
```

MSVC:

```
cl /WX /MT /c /O2 -D_CRT_SECURE_NO_DEPRECATED /I "C:\Program Files\X-Formation\LM-X SDK v4.6.1
win64_x64\include\" example.c

link /WX /opt:noref example.obj "C:\Program Files\X-Formation\LM-X SDK v4.6.1
win64_x64\win64_x64\liblmxclient_mt.lib"
```

Running your application

Your current directory should include the files listed in the table below.

File	Description
license.xml	Your license template.
license.lic	Ready to use license, generated with xmllicgen.
example.c	Your first program source code.
example/example.exe	Your program executable.



Now run your first LM-X licensed application.

# Distribute your LM-X licensed software to end users

After protection is implemented in your application, you can distribute it to your end users.

**Note:** When you distribute LM-X licensed software to your end users, you should bear in mind that all files shipped by X-Formation are digitally signed. However, since liblmxvendor.dll is a vendor-specific file, you should consider signing it yourself.

You must include the following in your distribution to the end user:

- The LM-X End-user Tools installation program (lmx\_enduser\_tools\_version.msi for Windows or lmx\_enduser\_tools\_version.sh for Unix)
- A license server configuration file (lmx-serv.cfg)
- The vendor library (liblmxvendor.dll for Windows or liblmxvendor.so/dylib for Unix)
- liblmxnet.dll when .NET wrapper was used
- lmxjava.dll (Windows) or liblmxjava.so/dylib (Unix) when Java wrapper was used

**Important:** When distributing .NET applications: To ensure that your .NET applications work for all Windows versions (not required for Windows 8 only), your installer *must* bundle and install the appropriate Microsoft Visual C++ Redistributable package (and in some cases, the .NET framework) along with your .NET application. This requirement is set by .NET runtime and cannot be done automatically by the LM-X SDK, so you must do this manually.

The installation program you send to the end-user will install:

- LM-X End-user Utility (lmxendutil), which is a command line tool that lets you get the HostID values for the computer system.
- LM-X End-user Configuration Tool (lmxconfigtool), which provides the same abilities as lmxendutil but uses a Windows GUI.
- The license server (optional). To use floating licenses, end users require a license server together with the liblmxvendor.dll library and the license server configuration file (lmx-serv.cfg)

The LM-X End-user Utility and End-user Configuration Tool can be used for license server configuration by letting you:

- Get the HostID values for the computer system.
- See who is currently using specific licenses on the license server.
- Remotely access the license server.
- Remove users from the license server.
- Read and verify a usage database and print the usage information to the screen.

If the license file needs activation, follow the steps for [license activation](#) in the next section.

# Activate your license using License Activation Center

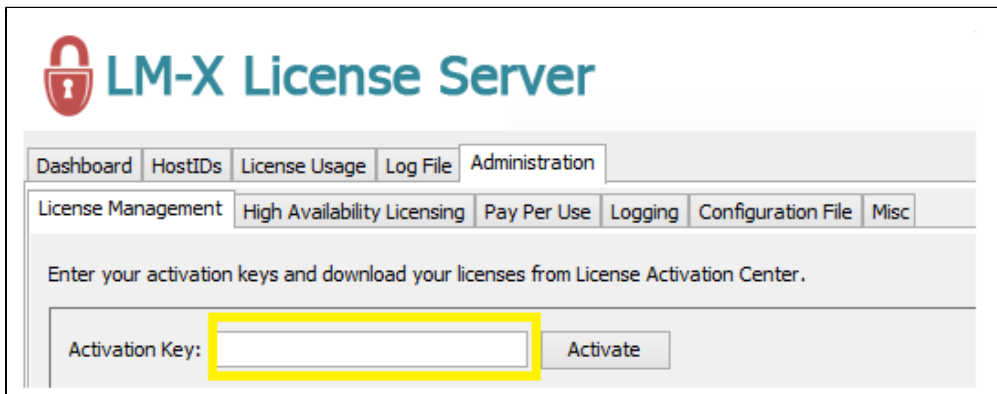
License activation is necessary when you distribute an application that will be locked to a particular end user's machine, or when the expiration date depends on the activation date. Otherwise, license activation is not usually necessary.

## Fully automating license activation

We highly recommend using LM-X together with [License Activation Center](#) (LAC) for the ultimate solution to license activation, particularly for managing greater numbers of activations. LAC provides an efficient license activation solution that lets you quickly and easily generate licenses and activation keys for end users. LAC gives end users a means to activate their own licenses, without your intervention.

### LM-X with LAC

A common way of integrating LM-X with LAC is by uploading [xmllicgen](#) to LAC as a license generator. LAC is responsible for handling information about existing licenses, including license generation and delivery, whereas LM-X is capable of using a generated license. You can completely automate license generation and delivery using the LAC REST and [SOAP APIs](#) that let you use LAC on any machine. In addition, the [LM-X web-based UI](#), which is integrated with LAC, lets you activate your license online by entering your activation key in the License Management sub-tab under the Administration tab, as shown below.



The screenshot displays the LM-X License Server web interface. At the top, there is a red padlock icon followed by the text "LM-X License Server". Below this, a navigation bar contains tabs: "Dashboard", "HostIDs", "License Usage", "Log File", and "Administration". Under the "Administration" tab, there is a sub-tab labeled "License Management". Other sub-tabs include "High Availability Licensing", "Pay Per Use", "Logging", "Configuration File", and "Misc". The main content area under "License Management" contains the text "Enter your activation keys and download your licenses from License Activation Center." Below this text, there is a form with the label "Activation Key:" followed by a text input field. The input field is highlighted with a yellow border. To the right of the input field is an "Activate" button.

You can also activate your license manually, or partially automate your license activation as described in [Different approaches to license activation](#).

# Different approaches to license activation

Apart from a fully automated license activation using [License Activation Center](#) (LAC), LM-X gives you the ability to activate your licenses in two ways:

- Semi-automated, using LM-X API
- Manual

## Partially automating license activation

You can partially automate your license activation by using the LM-X API to integrate HostID retrieval into your application. (See [LMX\\_HostidSimple](#) and [LMX\\_Hostid](#).) This enables you to obtain the information in a more automated way. You can also automate the execution of `xmllicgen` by integrating it with any custom license generation tools. License delivery would be the same as for manual license activation.

## Manually activating licenses

Manually activating licenses is feasible for issuing fewer numbers of licenses, but is difficult to manage for larger numbers of license activations.

To manually activate a license, you need to work individually with your end users as follows:

1. You ask the end user to run the LM-X End-user utility or End-user Configuration tool to obtain their HostID(s).
2. The end user sends you their HostID(s).
3. You enter the HostID tag(s) in the license template.
4. You generate a license using `xmllicgen`.
5. You send a license to the customer.

# LM-X Developers Manual

This manual describes how software application developers can use LM-X License Manager to license their software to end users.

## Getting started with LM-X

This chapter provides introductory information regarding LM-X, including an LM-X introduction, an overview of how LM-X works, a description of the distribution files, and a list of user community resources.

# Introduction to LM-X

LM-X License Manager lets you control your license policies externally from your application, specifying your licensing options in a separate license file. Although you can use LM-X with very little or no custom configuration, you have complete flexibility to license your application in any way you choose, with the ability to alter licensing policies on a per-customer basis, letting you focus on your business and fulfilling the needs of your customers.

Some highlights of LM-X benefits include:

- The ability to license your software application on a concurrent-usage or per-computer basis. You can further restrict the use of your licensed application to a single, specified computer or a set number of users on a network.
- A range of different licensing styles to fit your needs, including common solutions such as node-locked (stand-alone), floating (network), trial, time-limited, and perpetual licenses, to more complex solutions, such as feature-based, token-based, pay per use, and more.
- Flexibility in customization of licenses, including vendor-set fields for extensions or additional restrictions.
- Options that let you control the level of security on access to your application, such as HostID locking, expiration date, system clock check, protection against use on virtual machines, [dongle support](#), and more.
- Strong license verification using known public key cryptography such as RSA and symmetric cryptography with AES.
- Transparent connection handling between protected applications and license servers. LM-X will notify the application if a license server connection becomes unavailable, so the application can either ask LM-X to attempt reconnection to the license server or fail, as appropriate.
- The ability to separately license individual components of your program, so that customers can purchase only what they need.
- For your end users, LM-X is designed to be simple and trouble free, and includes features that make your application more attractive to end users, such as license borrowing, grace licenses, redundant servers (HAL), automatic server discovery, to name only a few.

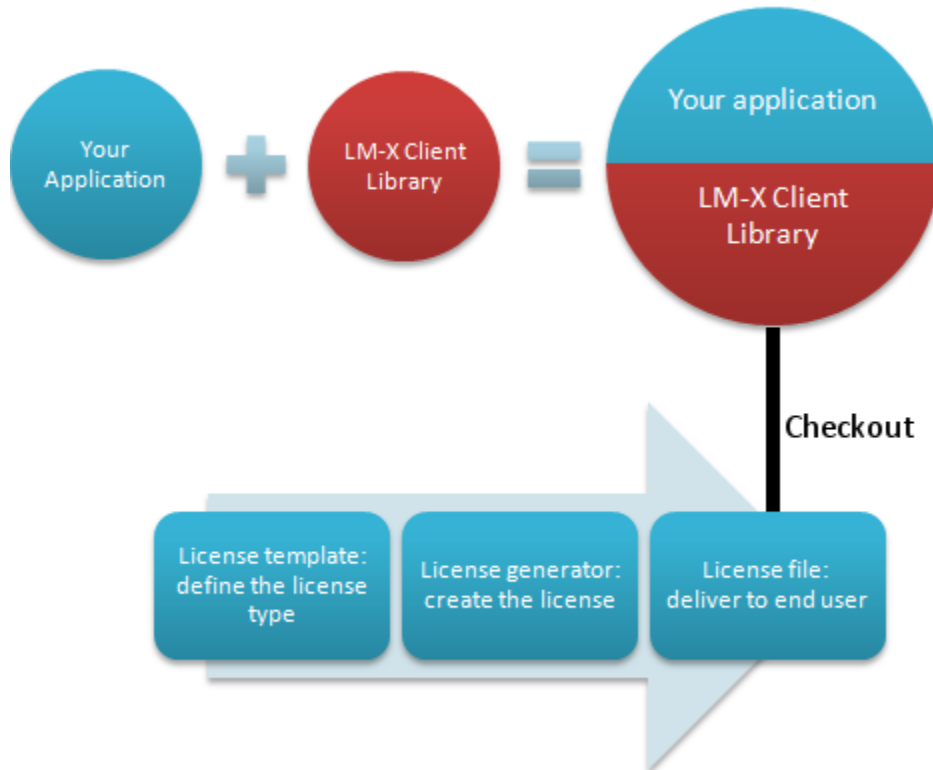
For further information on what you need to know in order to implement the most common license solutions, see [Application licensing strategies](#).

## How LM-X works

LM-X consists of a client library you use together with your application and a license generator you use to create license files. A license file contains text that defines the license agreement between you (the software vendor) and the end user. For example, the license defines whether the application is node-locked or floating, can run on any computer or only on a specific one, whether the license will expire on a specific date, and so on.

When you integrate the LM-X client library, it checks for a valid license. LM-X will automatically do all the checks for you, including available features, date and version, and checks against usage on unauthorized machines or users.

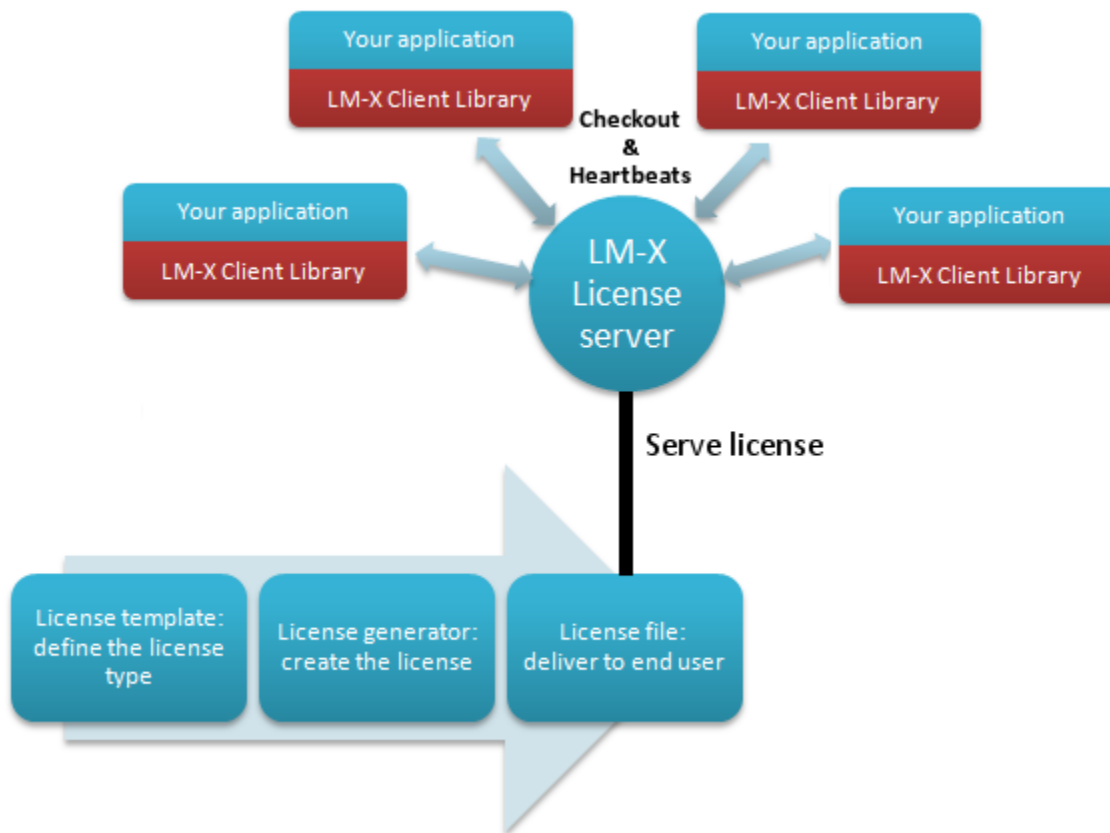
A node-locked licensing scheme is illustrated below.



For network licensing, LM-X additionally requires an external license server. The license server is software that is typically [set up as a service](#) (on Windows systems) or a daemon (on Unix systems) on the end user's network. The license server handles requests from users on a network to use the application, keeps track of floating licenses, and ensures there is no license overuse.

LM-X will check for a valid license file on the license server before granting permission to run the application. Periodic heartbeats occur between the license server and protected applications in the manner you specify (see [Heartbeats](#)). LM-X handles all the communication, based on TCP/IP protocol, which is transparent in the end users' network.





To help you determine what you need to put in place for the most common licensing strategies, see the following section, See [Application licensing strategies](#).

# LM-X distribution content

*The information on this page refers to LM-X v4.9.3 or newer, which added a new debug version (liblmxnetd.dll) of the .NET library file. If you are using a previous version of LM-X, please see the [documentation for previous versions](#).*

LM-X consists of several libraries and tools, described in this section. See [Getting Started With LM-X License Manager](#) for instructions on getting a license server and sample applications running in just a few minutes.

The following files are for developers and are **not for redistribution**.

Component		Description
Windows Filename	Unix Filename	
lmx-sdk_version_platform.msi	lmx-sdk_version_platform.sh	LM-X SDK installer.
lmxdev.exe	lmxdev	Tool for generating LM-X security configuration files and needed header files to compile the license generator and client application.
liblmxclient_md.lib liblmxclient_mdd.lib liblmxclient_mt.lib (or liblmxclient.lib) liblmxclient_mtd.lib	liblmxclient.a	Library for licensing, used by the client application.  Note that the Windows libraries liblmxclient_mt.lib and liblmxclient.lib are identical.
libhasphsupport.lib	libhasphsupport.a	Support library needed when dongles are used with LM-X.  (Not available on all platforms.)
lmx.h	lmx.h	Include file for the client library.
xmlicgen.exe	xmlicgen	The XML license generator.
examples	examples	This directory contains many examples on how to use LM-X.

The following files are for end users and may be **redistributed freely**.

Component		Description
Windows Filename	Unix Filename	
lmx-enduser-tools_version.msi	lmx-enduser-tools_version.sh	LM-X end-user tools and license server installer.
lmx-serv.exe	lmx-serv	License server. This executable is the same for all vendors.
no_security_config\lmx-serv.exe	no_security_config\lmx-serv	Pre-compiled license server, used to prepare license server for platform for which you have no physical machine.
lmx-serv.cfg	lmx-serv.cfg	Configuration file for license server.
liblmxvendor.dll	liblmxvendor.so or liblmxvendor.dylib	Library containing vendor-specific code such as callbacks, some LM-X server settings, and embedded security information.  Must be shipped to end users along with the license server (lmx-serv).
liblmx.dll	liblmx.so or liblmx.dylib	Client library.
liblmxnet.dll	N/A (Windows only)	Managed class library that is a .NET wrapper for the LM-X client API.
liblmxnetd.dll	N/A (Windows only)	Debug version of liblmxnet.dll (above).
lmxjava.dll	liblmxjava.so or liblmxjava.dylib	Library that is a Java wrapper for the LM-X client API.
liblmxjava.jar	liblmxjava.jar	Java archive with library that is a Java wrapper for the LM-X client API.
no_security_config\liblmx.dll	no_security_config\liblmx.so or liblmx.dylib	Pre-compiled client library, to be used when you want to use a different compiler than LM-X is compiled with.
lmxendutil.exe	lmxendutil	Command line utility for end users.

Imxconfigtool.exe	N/A (Windows only)	GUI utility for end users.
Imxresetsystemclock.exe	Imxresetsystemclock	Tool for resetting the end user's system clock.
Imxresettrial.exe	Imxresettrial	Tool for resetting the end user's trial license information.

See [Distributing applications to end users](#) for additional details about files that are distributed to end users.

# Application licensing strategies

*The information on this page refers to LM-X v5.0, which added the upgrade license type. If you are using an older version of LM-X, please refer to [documentation for earlier versions](#).*

LM-X License Manager handles the most simple to the most complex licensing needs. If you require a simple licensing solution, you can use LM-X default settings with few or no modifications.

If your licensing needs are more sophisticated, LM-X includes many advanced features to handle your needs, as described in [Feature descriptions](#). LM-X offers outstanding flexibility to adjust the license model to your business strategy in two main ways: a universal and wide selection of license file settings (tags), and a license file structure with independent sets of tags that define features. (See [Protecting your application](#) for details on license files.)

You should consider the following as part of your licensing strategy, depending on the type of license you want to use:

- For node-locked licenses, you should consider whether the license will work on terminal servers or virtual machines (not enabled by default) and whether to enable a trial license if a normal license is not in place.
- For floating licenses, you should consider policies for dealing with heartbeats and network disconnects (for example, how you will handle saving files and quitting the application in case the network is down).
- For trial licenses, you do not need to change your procedures. To enable trial licenses, you simply set a predefined trial length as described in [Trial licenses](#). A trial license begins when the application is run for the first time. Trial licenses have anti-reuse and anti-clock tampering mechanisms that prevent usage after the trial length has expired. You may also consider whether to allow the trial to work on terminal servers or virtual machines.

Example source code for each of these basic license types are included in the examples directory and are described in [Getting Started with LM-X License Manager](#).

The following tables show the most common feature settings for node locked and floating licenses, whether they are required or optional, and whether they are set by default.

## Settings for both node-locked and floating licenses

Setting	Required for node-locked	Required for floating	Usage	Set by default
<a href="#">KEY</a>	Yes	Yes	License key set by the license generator, which should not be modified.	Yes
<a href="#">VENDOR</a>	Yes	Yes	Your assigned vendor ID, which should not be modified.	Yes
<a href="#">MAJOR_VERSION</a>	Yes	Yes	The major version number of the feature.	Yes (Defaults to 1)
<a href="#">MINOR_VERSION</a>	Yes	Yes	The minor version number of the feature.	Yes (Defaults to 0)
<a href="#">END</a>	No	No	Specifies the license expiration date.	No
<a href="#">LICENSEE</a>	No	No	Specifies the customer name to which the license has been issued.	Yes
<a href="#">SYSTEMCLOCKCHECK</a>	No	No	Enables/disables the system clock check performed by client application and license server.	Yes
<a href="#">CLIENT_HOSTID</a>	No	No	Locks the license to a specific machine.  Note: This setting applies to both floating and node-locked licenses, because a license may be locked to both a server (using SERVER_HOSTID) and one or more clients.	No
<a href="#">HOSTID_MATCH_RATE</a>	No	No	Specifies the percentage of HostIDs that must match for successful HostID verification.	Yes (Defaults to 100%)

Using or changing one more of the above settings can change a licensing model. For instance:

To use this licensing model:	Modify the license file settings to:
Feature-limited demo license	Contain only the features you want to license.
Temporary license	Set END to the date on which you want the temporary license to end.
Permanent license	Do not set END.
Time-limited	Set END to a specific date.
Annual subscription	Set END to the current date + one year.
Copy protection license	Use CLIENT_HOSTID.

Shared license	Use SHARED=USER HOST CUSTOM.
----------------	------------------------------

## Settings for floating licenses only

Setting	Required	Usage	Set by Default
<a href="#">KEYTYPE</a>	Yes	Specifies whether a network license is exclusive, additive, upgrade or token-based.	Yes (Defaults to exclusive)
<a href="#">COUNT</a>	Yes	Specifies the number of licenses that can be issued simultaneously for the particular feature floating on the network.	No
<a href="#">SERVER_HOSTID</a>	No (highly recommended)	Locks the license to a license server machine.	No
<a href="#">BORROW</a>	No	Enables leasing of licenses to computers that can be disconnected from the network.	Yes
<a href="#">GRACE</a>	No	Enables leasing of licenses to computers that are disconnected from the network unexpectedly.	Yes

The following table describes licensing options that are commonly used, and whether they apply to node-locked or floating licenses.

License Option	Node-locked	Floating	Usage	Default Behavior
<a href="#">Trial licenses</a>	Yes	Yes	Lets end users use an application for a predetermined length of time without requiring a license.	Disabled by default. When enabled, trial licenses are not allowed on virtual machines or terminal servers by default.
<a href="#">Licensing for virtual machines and cloud computing</a>	Yes	Yes	Lets you explicitly allow trusted and enterprise customers to use your application in a virtual environment.	By default, LM-X denies all checkouts for local licenses in virtual environments, such as VMware or Virtual Server, and refuses to load licenses on license servers to prevent potential license overuse.
<a href="#">Automatic server discovery</a>	No	Yes	Allows a client application to find license servers on the network automatically.	By default, automatic server discovery is disabled on the client side.

## Controlling license behavior

You can easily add options that change licensing behavior prior to license checkout using [LMX\\_SetOption](#). These options let you control behavior for setting the license path, heartbeats, trial expiration, and many other settings.

For end-user settings such as automatic server discovery, license borrow, license queuing, etc., LM-X includes environment variables that end users can set as needed.

The environment variables are unset by default, and can be set on demand by end users. If you want to prohibit the use of one or more of the environment variables, you can unset them upon startup of your application using [LMX\\_Putenv](#).

The section [Environment variables](#) in the [LM-X End Users Guide](#) gives complete information about each environment variable, including the format of the environment variable paths.

## LM-X community resources and references

As part of the LM-X user community, you have access to the following resources and additional references to help answer your questions and give us your feedback:

- Search our [Knowledgebase](#), which is updated regularly with new information to help you quickly and easily find the answers you're looking for, including:
  - [Answers to commonly asked questions](#) about LM-X product features, installation, usage and related topics.
  - [LM-X training topics](#), such as how to choose the right HostID for your needs; how to provide demo versions of your software; how to rehost or blacklist a license; how to use dates for version numbers and more.
- Log into our [issue tracking system](#) to report software issues, give us your enhancement requests, track progress on reported issues, and more.
- Visit our [website](#) for the latest news, product information, support links, and [LM-X License Manager end user downloads](#).
- Log into our support portal to get [X-Formation product downloads and license information](#).
- Visit the [X-Formation blog](#) for the latest news about X-Formation products and services, industry news, etc.

## Protecting your application

This chapter describes using LM-X to protect your application, including licensing strategies, how license files are structured, and details about LM-X options you can use in your licenses.



# License file content

A license file is a text file that is produced using a license generator. A license file consists of blocks, which are referred to as "features." Features contain information regarding the specific modules in your program that are licensed. Each feature in a license has its own key, where all security information is encrypted.

Many different license models can be created by using particular LM-X license file settings (tags). Each feature has an independent set of such settings.

An LM-X license file has the following general format and limitations:

- License blocks (features) start with the keyword **FEATURE**, followed by an identifier (see [FEATURE settings](#)).
- A license may include unlimited features.
- Each block is enclosed in braces (**{}**).
- Inside each block are a series of directives, which are parameter/value pairs separated by an equal sign (=). See [Feature descriptions](#) for detailed information about each feature directive.
- Each line can contain only one directive.
- Custom fields should be enclosed in quotes (see [Features](#) for details of required and optional fields).
- Identifiers, directives, and their values are case-sensitive.
- Comment lines must be prefixed with a pound sign (#). For example:  
# my comment
- Whitespace characters (spaces and tabs) may be used freely for formatting.

The following sections describe the license file content in greater detail.

# Features

*The information on this page refers to LM-X v5.0 and newer, which added the upgrade license type. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

Features contain information regarding the specific modules in your program that are licensed. These blocks are linked directly to a specific vendor by the VENDOR directive inside the feature. This vendor id is determined at purchase time of the SDK and is specific to a single company.

The possible directives inside a feature are shown below. Required directives are marked with an asterisk (\*); all other directives are optional. Note that strings have a maximum length of 8192 characters (8 KB). Large strings will lead to a noticeable speed decrease in loading licenses.

```
*VENDOR = (vendor_id; generated by license generator)
*KEY = (generated by license generator)

MAJOR_VERSION = 0 - 9999
MINOR_VERSION = 0 - 9999
LICENSEE = "string"
START = YYYY-MM-DD
END = YYYY-MM-DD
MAINTENANCE_START = YYYY-MM-DD
MAINTENANCE_END = YYYY-MM-DD
ISSUED = YYYY-MM-DD
SN = "string"
DATA = "string"
COMMENT = "string"
OPTIONS = "string"
PLATFORMS = "platform strings"
TIME_ZONES = -12 to 13
REPLACES = "FEATURE=FeatureName or KEY=LicenseKey"

COUNT = 1 ? 2147483647 or UNLIMITED
TOKEN_DEPENDENCY = "FEATURE=FeatureName VERSION=MAJOR_VERSION.MINOR_VERSION
COUNT=1 - 2147483647"
KEYTYPE = EXCLUSIVE or ADDITIVE or UPGRADE or TOKEN
SOFTLIMIT = 5
HAL_SERVERS = 3
BORROW = 1 ? 8760
GRACE = 1 ? 168
HOLD = 1 - 1440
USERBASED = 1 - 2147483647 or ALL
HOSTBASED = 1 - 2147483647 or ALL

SHARE = HOST|USER|CUSTOM (network licenses only) and/or VIRTUAL (network or
local licenses) and/or TERMINALSERVER (local licenses only) or SINGLE (local
licenses only)
SYSTEMCLOCKCHECK = LOCAL|INTERNET|FALSE
HOSTID_MATCH_RATE = 0 - 100
```

For detailed information about each directive, see [Feature descriptions](#). For information about including features in your license, see [FEATURE settings](#).

# License file examples

*The information on this page refers to LM-X v5.0, which added the upgrade license type. If you are using an older version of LM-X, please refer to [documentation for earlier versions](#).*

The following shows an example of local, additive, upgrade and token license file content. Note that the network licenses include the **COUNT** directive.

```
# Local example
FEATURE f1
{
  VENDOR=MYCOMPANY VERSION=1.4 START=01-01-2021 END=01-06-2021
  SHARE=TERMINALSERVER LICENSEE="Our_user"
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}

# Additive example
FEATURE f2
{
  VENDOR=MYCOMPANY KEYTYPE=ADDITIVE COUNT=6 VERSION=1.0
  SHARE=HOST BORROW=168 LICENSEE="Our_user"
  KEY=4i]mYsfN30C6ShBYszCq2WVicpTZxQwkfKJTohkzglwNkle...
}

# Upgrade example (requires f3 exclusive license)
FEATURE f3
{
  VENDOR=MYCOMPANY KEYTYPE=UPGRADE COUNT=6
  KEY=szCq2WVicpTZxQw9Uuz12b2B3v]vcsFBx7mYsfN30C6kw71...
}

# Token example
FEATURE f4
{
  VENDOR=MYCOMPANY KEYTYPE=TOKEN VERSION=1.0 END=01-01-2022
  TOKEN_DEPENDENCY="FEATURE=f2 VERSION=1.0 COUNT=3"
  KEY=Uuz12b2B3v]vcsFBx7qEQG1SwXCz8AWVicpTZxQw9Uuzj1k...
}
```

## Feature descriptions

Descriptions of each directive that can be included in a feature are given in the following sections. For information about including features in your license, see [FEATURE settings](#).

## VENDOR

The VENDOR directive contains the vendor ID that X-Formation assigned to you upon your purchase of LM-X. This directive is set by the license generator, and is checked against to ensure that the content of the block has not been modified.

# KEY

The KEY directive is set by the license generator, and is checked against to ensure that the content of the block has not been modified. This directive can also hold custom data embedded for usage with the protected application.

To add custom data, use the [KEYCOMMENT](#) [setting](#). Note that the [OPTIONS](#) and [COMMENT](#) settings are also validated in the key.

## MAJOR\_VERSION

The MAJOR\_VERSION directive indicates the major version number of the feature. Valid values range from 0 to 9999. If unspecified, the version number defaults to 1.

## MINOR\_VERSION

The MINOR\_VERSION directive indicates the minor version number of the feature. Valid values range from 0 to 9999. If unspecified, the minor version number defaults to 0.



# **LICENSEE**

The LICENSEE directive indicates the user or company to whom the license was issued. When this field is used, the entry is enclosed in quotation marks. You should use printable characters and not use quotes (") within the string.

# START

The START directive specifies the date on which the feature becomes valid. The date is ISO-8601 compliant, in the format YYYY-MM-DD. The START time is 00:00:00 local time (for example, if software is sent from the United States to China, it will start at 00:00:00 China time).

# END

The END directive specifies the date on which the feature expires. The date is ISO-8601 compliant, in the format YYYY-MM-DD. The END time is 23:59:59 local time (for example, if software is sent from the United States to China, it will expire at 23:59:59 China time).

## MAINTENANCE\_START

The MAINTENANCE\_START directive specifies the date on which your license maintenance plan begins. LM-X does not use this keyword; it is only used to make your licensing terms more clear. The date is ISO-8601 compliant, in the format YYYY-MM-DD.

## MAINTENANCE\_END

The MAINTENANCE\_END directive specifies the date on which your license maintenance plan expires. LM-X does not use this keyword; it is only used to make your licensing terms more clear. The date is ISO-8601 compliant, in the format YYYY-MM-DD.

## ISSUED

The ISSUED directive specifies the date on which the license was created. LM-X does not use this keyword; it is only used to make your licensing terms more clear. The date is ISO-8601 compliant, in the format YYYY-MM-DD.

## SN

The SN directive stores a custom serial number for the license.

When this directive is used, the entry is enclosed in quotation marks. You should use printable characters and not use quotes (") within the string.

# DATA

The DATA directive stores additional information regarding the license.

This directive is not evaluated during the generation of the KEY checksum; accordingly, you can change values in the DATA field without needing to generate a new key.

With this directive, you can use entries enclosed in quotation marks. You should use printable characters within the string.



# COMMENT

The COMMENT directive stores additional information regarding the license. With this directive, you can use entries enclosed in quotation marks. You should use printable characters within the string. The information is validated in the key (see [KEY](#)), so it is protected from tampering.

**Note:** This directive is the same as the [OPTIONS](#) directive.

# OPTIONS

The OPTIONS directive stores additional licensing options for the license. With this directive, you can use entries enclosed in quotation marks. You should use printable characters within the string. The information is validated in the key (see [KEY](#)), so it is protected from tampering.

**Note:** This directive is the same as the [COMMENT](#) directive.

# REPLACES

The REPLACES directive replaces one license with another license. Replacing an old license with a new license ensures that customers can no longer use the old license on the same machine simultaneously.

A license can be replaced by specifying the feature name or the license key. Multiple features can be specified using a comma-separated list.

For node-locked licenses, when the REPLACES tag is set, the client must specify the path for both the license that will be replaced and the replacement license. For network licenses, the license server will find the licenses upon startup.

# COUNT

The COUNT directive specifies the number of licenses that can be issued simultaneously for the particular feature floating on the network. The count may be up to 2147483647 (the 32-bit integer limit) or may be set to UNLIMITED.

If the COUNT directive is omitted, the license is considered to be for local usage. If the COUNT directive is specified, the feature *must* be hosted on a license server.

# TOKEN\_DEPENDENCY

The TOKEN\_DEPENDENCY directive specifies the real license (or another token-based license) that a token-based license is dependent upon.

If the TOKEN\_DEPENDENCY directive is specified, the feature *must* be hosted on a license server. The TOKEN\_DEPENDENCY directive *must* be used together with KEYTYPE = TOKEN and cannot be used with the [COUNT](#) directive.

## Usage and recommendations

See [Token-based licensing](#) for further information about implementing this feature and examples of usage.

# KEYTYPE

*The information on this page refers to v5.0 and later, which introduced the upgrade license type. For documentation applicable to earlier versions, see [documentation for previous versions](#).*

The KEYTYPE directive specifies whether a network license is exclusive, additive, [upgrade](#) or [token-based](#).

- By default, all network licenses are exclusive. Exclusive licenses are taken into account before other license types. If two exclusive licenses of the same feature name are present, the first is used and the second is discarded.
- Additive licenses coexist with exclusive licenses and other additive network licenses of the same feature name, and are seen as a separate license pool of the same feature.
- [Upgrade licenses](#) are similar to additive licenses, but rather than co-existing with an exclusive license of the same feature name, they are applied to the corresponding exclusive license during license server startup, transparent to the client. Upgrade licenses increase the license count so that only one feature exists.
- [Token-based licensing](#) is always additive, and lets you specify that a particular feature should use one or more other licenses to fulfill checkout requests.

## Usage and recommendations

The KEYTYPE directive can be used only with network licenses; accordingly, the [COUNT](#) directive or [TOKEN\\_DEPENDENCY](#) directive must be set if KEYTYPE is set.

- To prepare an [upgrade license](#), create a feature with the same name as the original license, set the number of licenses using the [COUNT](#) directive, and set KEYTYPE="UPGRADE" in the XML license template file. (See [Upgrade licensing](#) for more information about how to define an upgrade license.)
- To prepare an additive license, set KEYTYPE="ADDITIVE" in the XML license template file.
- To prepare a token-based license, the "KEYTYPE=TOKEN" directive must be used together with the [TOKEN\\_DEPENDENCY](#) directive. Token-based licenses are always additive. (See [Token-based licensing](#) for more information about how to define a token-based license.)

## Additive and upgrade licenses

To add licenses for software features that a customer buys after the initial purchase of a license without changing the original license, you can use either additive or upgrade licenses.

While upgrade licenses increase the license count for an existing feature, additive licenses are treated as a separate license pool of the same feature; therefore, you must checkout licenses in the proper order when using additive licenses.

It is important to understand that multiple additive licenses (even when combined with an exclusive license) do not behave exactly the same as one license with a combined license count (as opposed to [upgrade licenses](#)).

If a client tries to acquire multiple additive licenses, the maximum number of licenses that can be acquired per [LMX\\_Checkout](#) call will be equal to what one license holds. The license server will not perform checkouts across multiple additive licenses.

For example: A license server has one exclusive and two additive licenses of feature "ABC" available, with the license counts 1, 2 and 3. When a client requests one "ABC" license, the exclusive license with count 1 will be used. If this same client later attempts to acquire three more "ABC" licenses, only the additive license with the license count of 3 will be used rather than using one license from the first additive license and two from the third additive license.

To avoid problems, you should checkout all needed additive licenses at one time, or preferably, use [upgrade licenses](#) instead, which eliminates such potential problems completely.

# SOFTLIMIT

The SOFTLIMIT directive enables segmenting the license count into two pools. Checkout requests that exceed the specified softlimit will respond differently to the application than those within the specified softlimit.

The value specified in the SOFTLIMIT directive must be less than the value specified in the [COUNT](#) directive.

This directive can be used only with network licenses; accordingly, the COUNT directive *must* be set if SOFTLIMIT is set.

## Usage and recommendations

Softlimit licensing lets you segment a network license pool into two ranges: normal licenses (the specified softlimit number) and overdraft licenses (licenses exceeding the specified softlimit and up to the specified license count).

When the number of licenses in use exceeds the softlimit, LMX\_Checkout() will start to return LMX\_SOFTLIMIT instead of LMX\_SUCCESS. This information can be used to initiate a response, such as alerting end users when they are in an overdraft state and should be careful about their usage.

For example, with 5 softlimit licenses and a network license count of 10, checkout requests 1 through 5 will return LMX\_SUCCESS. Checkout requests 6 through 10 will return LMX\_SOFTLIMIT, indicating an overdraft state. Checkout requests over 10 will be denied.

One use of softlimit licensing might be to make a distinction for billing purposes, where a certain number of licenses are sold at full price, and the rest are sold at a reduced price. You would then use the pay-per-use log to verify the actual usage.

Softlimit licensing may also be used to implement a more lenient license model, which permits temporary overdraft licenses only in the event of license server failure.

For example, say the softlimit is 10, and the count is unlimited. If all 10 licenses are in use, further checkout requests will be denied by default. However, if the license server is restarted, new users could obtain licenses before the original 10 users invoke their heartbeat. With softlimit, you can allow temporary overuse to ensure that when the original users invoke the heartbeat, they regain their licenses. Further requests are then denied until the license count is once again less than the softlimit. This allows users who were using the server before the failure to continue their work without interruption.

# HAL\_SERVERS

The HAL\_SERVERS (redundant servers) directive enables high availability licensing (HAL). HAL lets end users specify backup (redundant) license servers that will continue to enable license hosting in the event that the primary license server goes down.

When HAL is enabled, the end user must have a set of three license servers. Whenever two of the servers (majority rule) are up, the servers will issue licenses.

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if HAL is set.

## Usage and recommendations

High availability licensing (HAL) is an important feature for applications that require high availability. Activating HAL introduces fault tolerance, because the applications no longer depend on a single point of failure.

Each of the three HAL license servers has a fixed role, which is specified in the LM-X security configuration file. Assigning fixed roles to HAL license servers simplifies maintenance as well as helping to protect against intentional or unintentional license overuse.

The roles are:

HAL License Server Number	Role
1	This HAL license server can allow clients to both checkout and borrow licenses, exactly like a normal license server.
2	In the event that HAL license server #1 is down, this server can allow clients to checkout licenses, but will deny borrow requests.
3	This HAL license server will deny any requests, but is required as a part of the configuration to ensure high availability.

If you want to lock the license to the license server HostID, you *must* provide values for all three license servers to ensure proper functionality. This means that you should issue end users a single license that works on all three machines. However, locking the license to a HostID is optional and you may choose not to lock the license to any HostID, as described in [Specifying HostIDs to lock the license to](#).

HAL requires a stable network connection to prevent hiccups in traffic between the servers, which can potentially cause license checkout problems. Too many network problems will make the system unstable and people will not get reliable access to the software. For this reason, enabling HAL over WAN is recommended only if the connection is fast and stable.

Note that a license server can serve either HAL-enabled licenses or normal network licenses. To enable HAL, you must specify the HAL server information as described in [FEATURE settings](#).

We recommend that end users specify all three servers in their license path when using this feature if automatic server discovery is not used. Doing so will let the license client choose the appropriate secondary license server in the event that the primary license server goes down.



# BORROW

The BORROW directive enables leasing of licenses to computers that can be disconnected from the network. The parameter associated with BORROW tells how many hours a license can be leased. Valid values range from 1 to 8760 hours (1 year).

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if BORROW is set.

## Usage and recommendations

The license borrowing feature lets users run your software on a computer that is not continuously connected to the network, such as a laptop, by allowing a floating network license to be leased for a time-limited period.

Borrowing works by making a floating network license available for a certain amount of time to the requesting computer, and making it unavailable on the license server for this time. Through a borrow request, the client can obtain the time-limited license, which will work like a local uncounted license.

When performing borrowing, a temporary license file is written on the client side, which will be used for subsequent license requests, with the checked out license(s) marked unavailable for the same period of time on the license server.

At the time of the borrow request, the client must be connected to the network. After a successful borrow request, the client will make use of its temporary license and can be disconnected from the network. Note that the license is borrowed for only the currently logged in user of the client machine.

Borrowing is enabled by default. You may wish to disable borrowing to increase checkout speed (see [Optimizing license checkout speed](#)) or to increase security. Although fairly secure, borrowing is by nature less secure than other mechanisms in LM-X. You should recognize that there is a trade-off between the convenience offered by implementing borrowing and the overall strength of your licensing solution.

To use borrowing for your applications, you should set the permitted lease time length at license creation, as described in [FEATURE settings](#). Licensed applications must then enable borrowing by calling `LMX_SetOption()`. (See `LMX_OPT_ALLOW_BORROW` in `LMX_SetOption`.) The maximum number of hours that may be set is 8760, or one year; however, setting the shortest reasonable lease times is recommended for best security.

The client computer can borrow a license or return the license early using a normal checkout with the proper environment variable settings, as described in [Controlling license behavior](#).

If a feature is borrowed for more than one hour, and the client reestablishes a connection with the license server, the client will refresh the borrow time. For example:

1. A client borrows Feature A from Server 1 with a borrow time of 24 hours.
2. 12 hours later, the client checks in Feature A, but does not do a borrow return.
3. If the client later performs a checkout of Feature A from Server 1 and establishes a connection with the server, the checkout will succeed and the borrow hours for Feature A will once again be 24 hours. However, if the client cannot establish a connection with the server, the borrow time will be only 12 hours (the remaining hours left after originally borrowing the feature).

**Note:** The borrow time is not refreshed within the first hour. Therefore, a new checkout done within the first 60 minutes does not guarantee another borrow time of 24 hours.

Temporary data is written on both the client and license server to save information about borrow leases. If the data is deleted at the client, the borrowed license will no longer function; however, if the data is deleted at the server, the license will again become available.

Therefore, be aware that if an end-user site moves a license server from one machine to another, previously leased licenses will continue to work without being counted by the license server, and this may lead to over-licensing.

As an alternative to allowing the borrow time to run out, you can send the customer a new license that is functionally identical but has a different license key.

# GRACE

The GRACE directive enables leasing of licenses to computers that are disconnected from the network unexpectedly. The parameter associated with GRACE defines the number of grace hours a license may be used before a network connection to the license server is required again. Valid values range from 1 to 168 hours (7 days).

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if GRACE is set.

## Usage and recommendations

Grace licenses allow users to continue working with an application when the license server goes down. In emergency situations, when no one is available to restart the license server, a time-limited grace license lets the user continue to run the feature. Note that the grace license is available only for the currently logged-in user of the client machine.

Grace licenses work very similarly to license borrowing, described in [BORROW](#). A temporary license is written at the client side upon each successful license checkout from a license server. This license is written in an inactivated state, so it does not have a fixed expiration date set.

If the server becomes unavailable, heartbeats will fail and eventually let you cease the execution of the application, as described in [Heartbeats](#). Upon next checkout, the local grace license is activated in place of the network license for the permitted number of hours. If the license server comes up before the grace license expires, future checkout requests will acquire a network license as usual, but will not reset the expire time of the grace license.

To prevent fraud and over-licensing, LM-X does not allow the user to obtain a new grace license for 7 days after the expiration of the previously activated grace license. If the user needs a local license in the meantime, using borrow is suggested.

You should be cautious about over-licensing, because the server does not remove the grace license from the pool of available licenses in the same manner as for borrow. The grace licenses are counted *in addition* to the server licenses. For example, if a server has five available licenses of a feature, and two licenses are borrowed, the server would have only three licenses available until the two borrowed licenses are returned. However, if two grace licenses are being used, the server will still have those two licenses available for checkout until the grace period for those licenses has expired.

Grace licensing is enabled by default. As with license borrowing, you may wish to disable grace licensing to increase checkout speed (see [Optimizing license checkout speed](#)) or to increase security. Although fairly secure, grace licensing is by nature less secure than other mechanisms in LM-X, and there is a trade-off between convenience and the strength of your licensing solution.

To use grace licenses, you should set the permitted lease time length at license creation, as described in [FEATURE settings](#).

Licensed applications must then enable grace licenses by calling `LMX_SetOption()`. See [LMX\\_OPT\\_ALLOW\\_GRACE](#) in [LMX\\_SetOption](#).

# HOLD

*Some information on this page refers to LM-X v4.8.1 or newer, which extended the range of valid values for the HOLD directive. If you are running a previous version of LM-X, see [documentation for previous versions](#).*

The HOLD directive holds licenses for a specified period of time after they are checked in. The server holds licenses after the client disconnects, as if the client is still active, preventing licenses from being transferred from one machine to another for the specified time. Valid values range from 1 to 31536000 seconds (1 year).

The HOLD directive will hold licenses only when the client has checked in all licenses that it checked out. For example, say that 10 licenses are checked out for feature f1, and HOLD is set for 60 seconds. If only 2 of the 10 licenses are checked in, 2 licenses will be released at the same time, without HOLD. If later, the remaining 8 licenses are checked in, the server will hold 8 license for 60 seconds.

The HOLD directive can be particularly useful when combined with [host sharing](#).

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if HOLD is set.

# MIN\_CHECKOUT

*Some information on this page refers to LM-X v4.8.1 or newer, which extended the range of valid values for the MIN\_CHECKOUT directive. If you are running a previous version of LM-X, see [documentation for previous versions](#).*

The MIN\_CHECKOUT directive holds licenses after they are checked in if the license was used for less than the specified minimum checkout time. For example, if you specify MIN\_CHECKOUT=240 (4 minutes), and the license is used for 2 minutes, the license will be held for an additional 2 minutes after checkin.

In the same manner as the [HOLD](#) directive, the server holds licenses after the client disconnects until the minimum checkout time has been reached. Valid values range from 1 to 31536000 seconds (1 year).

The MIN\_CHECKOUT directive can be particularly useful when combined with host sharing, which is described in the [SHARE](#) section.

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if MIN\_CHECKOUT is set.

# PLATFORMS

*The information on this page refers to LM-X versions 5.2 and newer, which added support for Linux ARM. If you are using an older version of LM-X, see [documentation for previous versions](#).*

The PLATFORMS directive specifies one or more platforms to restrict usage to a single platform or a subset of platforms. This directive may be used for local or network licenses. When used for network licenses, it applies only to the client machine and not to the license server machine.

## Usage and recommendations

By limiting license use to one or more platforms, you can increase licensing flexibility and security. For example, you might produce software that runs on both Windows and Linux, but enable users to install and use the software on only one of the two platforms.

You use the PLATFORMS setting (see [FEATURE settings](#)) to create a list of platform keywords that specify the platforms the application will work with.

**Note:** The platforms are identified by the executables compiled with LM-X rather than by the actual operating system. For example, a 32-bit executable will still return Win32\_x86 on Win64\_x64.

LM-X lets you specify the following platforms:

Platform Keyword	Description
Win32_x86	Windows 32 bit on x86
Win64_x64	Windows 64 bit on x64
Macosx_Universal	Mac OS X
Linux_x86	Linux 32 bit on x86
Linux_x64	Linux 64 bit on x64
Linux_ARM	Linux 64-bit on ARM
Freebsd_x64	FreeBSD 64 bit on x64

## USERBASED and HOSTBASED

The USERBASED and HOSTBASED directives reserve a number of licenses (all or a specified number) that will be available only to the named users (for USERBASED) or hosts (for HOSTBASED). The users or hosts must be specified in the reservation lists of the license server, or you will get a warning in the license server log that the license is not available for usage.

Although they are not mutually exclusive, normally only one of these two keywords is used at one time.

This directive can be used only with network licenses; accordingly, the [COUNT](#) directive *must* be set if USERBASED or HOSTBASED is set.

### Usage and recommendations

Making a license user- or host-based effectively makes it a "semi-floating" license, as you must specify in advance who can use the license. This licensing model can be beneficial for such uses as applying a discount to floating licenses, since it places restrictions on the floating license but is not as rigid as a local license.

# TIME\_ZONES

The TIME\_ZONES directive defines the allowed time zones relative to GMT.

The time zones are specified by entering a comma-separated list of numeric values that represent the offset from GMT. Time zones that are earlier than GMT must be preceded by a minus sign (-).

For example, an entry of "-5" specifies the time zone GMT -05:00. An entry of "-5,1,2" specifies three different time zones in which the license may be used.

This directive can be used with network or local licenses.

## Usage and recommendations

Time zone restricted licenses, also known as WAN (Wide Area Network) licenses, let you control license usage based on geographical location. You can restrict a license to a single time zone or multiple time zones in a comma-separated list. The time zones are specified as numbers that are the hours offset to GMT.

Using time zone restricted licensing, you can ensure that licenses are used only in the country or region you intend. For example, you may sell licenses in the Continental U.S. and China at different prices. To make a license valid only in the Continental U.S., you would specify the time zones -8, -7, -6, -5 in the license file. For China, you would specify the time zone GMT 8 in the license file. This way, you can be sure that licenses aren't shared between a company's departments in the U.S. and China but are purchased separately.

# SHARE

The SHARE directive specifies the type of license sharing that is in use. Share options are described in the following table.

Option	Description	Used for Network Licenses	Used for Local Licenses
HOST	<p>This option specifies that processes running on the same host will share licenses and that processes running on different hosts will use different licenses.</p> <p>Using the HOST option allows a user to start multiple sessions of the same application on the same host, yet take licenses for only one session.</p>	Yes	No
USER	<p>This option specifies that processes started by the same user on a unique machine or on multiple machines will share the same license, regardless of how many machines the license runs on, and processes started by different users will use different licenses.</p> <p>Using the USER option allows a user to start multiple sessions of the same application on the same machine or on multiple machines, yet take licenses for only one session.</p>	Yes	No
CUSTOM	<p>This option specifies that processes started using the same custom sharing string will be shared. This option also specifies that processes started using different custom sharing strings will use different licenses.</p> <p>You can use the CUSTOM share option to custom-define when to enable sharing, thereby potentially making the license sharing more restrictive.</p>	Yes	No
VIRTUAL	<p>This option allows an application to run in a virtual machine environment; for example, using VMware or Virtual Server. See <a href="#">Licensing for virtual machines and cloud computing</a> for more information.</p>	Yes	Yes
TERMINALSERVER	<p>This option specifies that the licensed application will work for remote terminal server clients.</p> <p>When TERMINALSERVER is <i>not</i> specified, remote clients will be blocked, thereby preventing a local license from being shared for multiple users via a terminal server.</p> <p>The TERMINALSERVER tag applies only to Windows and is ignored by Unix.</p> <p>(For information about problems that may occur when you set the SHARE=TERMINALSERVER directive, see <a href="#">Licensing issues</a>.)</p>	No	Yes
SINGLE	<p>This option limits checkouts to one instance of a feature across multiple sessions. A single-usage license is essentially a one-count floating license for a single host, but eliminates the need to setup a license server.</p>	No	Yes

Note that the HOST, USER, and CUSTOM share options do not prevent a *single* process from using more than one copy of a feature license. Rather, these share options allow *different* processes to share the same licenses.

For share options used with network licenses, the [COUNT](#) directive *must* be set.

## Combining share options

For network licenses, HOST, USER, CUSTOM and VIRTUAL sharing can be used together. For local licenses, VIRTUAL, TERMINALSERVER and SINGLE sharing can be used together. When combining multiple share options, separate them with a vertical bar ( | ); for example, SHARE = HOST|USER|CUSTOM.

Combining the HOST, USER, and CUSTOM share options results in a logical AND, so the license will be shared between the options if the strings match for the specified criteria; otherwise, the license will not be shared. Combining these share options results in the license being shared in fewer cases than if only one share option is used.



# SYSTEMCLOCKCHECK

*The information on this page refers to LM-X v4.5.2 and newer, which added the ability to use a remote time server for SYSTEMCLOCKCHECK. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The SYSTEMCLOCKCHECK directive specifies the system clock check performed by the client application and license server.

You can set SYSTEMCLOCKCHECK to:

- Check the clock on the local file system time (the default)
- Check the clock on a remote Network Time Protocol (NTP) time server
- Disabled (do not check the clock)

Setting the SYSTEMCLOCKCHECK option is described in [FEATURE settings](#).

Disabling system clock check can improve checkout speed (see [Optimizing license checkout speed](#)); however, be aware that disabling system clock check weakens security.

For information about resolving issues users may encounter related to the system clock check, see [System clock check](#).

# HOSTID\_MATCH\_RATE

The HOSTID\_MATCH\_RATE directive specifies the percentage of HostIDs that must match for successful HostID verification. This lets you lock your software to multiple HostID values, but allow users to run the software if, for example, only two out of three HostID values are valid.

The HostID match rate setting is a percentage from 0 (HostID verification disabled) to 100 (exact matching) that is required for a successful HostID verification. If no match rate is set, the value defaults to 100.

Note that this setting works only for standard HostID checking; it cannot be used with custom HostID comparison.

## Usage and recommendations

HostID matching lets you offer a less strict hardware upgrade policy to your users, because you can decide whether to allow license checkouts using an exact 1:1 match or a more lenient partial match of HostID values.

For example, you might define a license file as follows:

```
<CLIENT_HOSTID>
<SETTING ETHERNET="123..." />
<SETTING HOSTNAME="ALPHA1" />
</CLIENT_HOSTID>
<SETTING HOSTID_MATCH_RATE="50" />
```

In the above example, both Ethernet and hostname HostIDs were specified. With the match rate set to "50," only one of the two HostIDs is required to match in order for the license to work.

The match rate is rounded down to the nearest full percentage. For example, if 2 out of 3 HostIDs match, the match rate is 66% (rounded down from 66.67%); thus, the HostID will be valid if the match rate is set to 66 or above.

## Optimizing license checkout speed

License checkouts normally take around 1 second, depending on the machine. For applications where license checkout speed is of high importance and/or when checking out a large number of features, you may wish to do one or more of the following.

- For network licenses, you can disable [Borrow](#) and [Grace](#) features.
- For node-locked licenses, you can disable [SystemClockCheck](#).
- For both node-locked and network licenses, you can disable unused HostIDs using the options [LMX\\_OPT\\_HOSTID\\_DISABLED](#) and [LMX\\_OPT\\_HOSTID\\_ENABLED](#), described in [LMX\\_SetOption](#). To determine the results disabling each HostID type has on checkout speed, you can run a benchmark test, as shown in the example "examples/checkout\_benchmark" in the SDK.

Making one or more of these changes will help to decrease checkout time. However, be aware that disabling system clock check reduces security and disabling borrow and grace features reduces flexibility for end users.

If checkouts are slow for unknown reasons, you can enable [extended logging](#) to help determine the cause for the poor performance.

# License server

If you ship network licenses to your end users, they will need to use a license server. The license server, `lmx-serv`, reads your network license and makes it available over the customer's network. The supplied configuration file, `lmx-serv.cfg`, which serves as a template for end users, should be used along with the license server.

Communication between the client (your application) and the server is done over a TCP/IP connection at port 6200 (default). When the client requests a license, it is taken from the pool of available licenses and is reserved for that client. A client can be connected to multiple license servers simultaneously. This provides the flexibility to use multiple servers in cases where a checkout request involves checking out more than one feature. For example, for an application that consists of multiple features, a client can check out Feature1 from ServerA and Feature2 from ServerB.

Since clients may at times disconnect unexpectedly, the license server maintains a timeout. If the server does not receive any communication from the client application within the timeout, any licenses assigned to that client will be returned to the available license pool.

The connection to the server is established when the client attempts to checkout a license. If the connection fails, the client will attempt to reestablish the connection it had before the failure.

Once the client has performed its last checkin (that is, has returned all licenses to the server), the connection to the server is closed. It is recommended that client applications explicitly checkin all licenses instead of using the LM-X API to checkin licenses (see [LMX\\_Checkin](#)).

# License server extensions

The license server supports extensions to accommodate custom behavior; for example, you can:

- Modify the length of connection timeouts.
- Set the custom HostID callback function to create licenses that are locked to a license server with a custom HostID.
- Enable notifications for cases when a client tries to checkout or reserve a feature that is not available.
- Get information prior to or as part of the checkout/checkin, reservation, and reservation early return requests on the server side, which allows for flexibility when determining which licenses should be checked in and out, and lets you change license requests dynamically.
- Implement custom handlers for the License Server start and stop events.

You include your custom code in [lmx\\_server\\_conf.c](#) file, which is used to build the file liblmxvendor.dll (for Windows) or liblmxvendor.so (for Unix). A default liblmxvendor file is included with the SDK.

You must distribute liblmxvendor.dll (or .so) to end users together with the license server executable (lmx-serv). See [End-user file distribution](#) for more information about distributing this and other files to end users.

## Supplying a license server for platforms you lack in-house

You may have customers who want to install the LM-X License Server on a platform that you do not have in-house. To accommodate such needs, you can use a license server pre-compiled for a particular platform.

To do this, you copy the vendor library liblmxvendor.dll (for Windows) or liblmxvendor.so (for Unix) for the desired platform to your Windows or Unix machine, and run the lmxdev command (see [Lmxdev developer tool](#)) to embed the security configuration information into the liblmxvendor.dll/so file. You then ship the LM-X End-user Tools installer (lmx-enduser-tools\_version.msi/sh), which also installs the license server and the vendor library to the customer.

This procedure needs to be done only once, for each platform on which you will distribute your software but do not have a physical machine.

For example, say you have purchased LM-X for Windows and AIX, but you do not have an AIX machine available in-house. After copying the liblmxvendor.dll/so file for AIX to your Windows or Unix machine, you embed the security configuration file into the liblmxvendor.dll/so file, as follows.

1. Copy the liblmxvendor.dll/so file from the LM-X distribution to the machine that contains the lmxdev executable.
2. Run lmxdev using the -embedsecurityconfig option, as follows:

```
C:\lm-x\win32_x86\no_security_config> ..\lmxdev.exe -embedsecurityconfig ../../config\xformation.lmx
liblmxvendor.dll
LM-X Developer Tool v4.5.1
Copyright (C) 2002-2015 X-Formation. All rights reserved.
Licensed to X-FORMATION
Security configuration successfully embedded!
```

3. Run lmxdev again using the -embedlicense option, as follows:

```
C:\lm-x\win32_x86\no_security_config> ..\lmxdev.exe -embedlicense ../../config\lmx.lic liblmxvendor.dll
LM-X Developer Tool v4.5.1
Copyright (C) 2002-2015 X-Formation. All rights reserved.
License file successfully embedded!
```

4. Send the LM-X End-user Tools installer and the liblmxvendor.dll/so file to your customers who wish to run the license server on that particular platform.

Note that failure to embed the security configuration file in the liblmxvendor.dll/so file will result in errors such as the following when the user attempts to start up the license server:

```
[2010-01-16 21:28:17] FAIL: Vendor security configuration is not embedded!
```

## Using alternative compilers

If you do not have or want to use the compiler that LM-X is compiled with, you can use a pre-compiled client library. For example, you may want to compile LM-X for AIX but do not want to use GCC; instead, you want to use the compiler that IBM provides.

To embed the security configuration file (located in the `no_security_config` directory and named `liblmxvendor.dll` for Windows and `liblmxvendor.so` or `liblmxvendor.dylib` for Unix) into the client library file, run the `lmxdev` command (see [Lmxdev developer tool](#)), as described below.

1. Run `lmxdev` using the `-embedsecurityconfig` option, as follows (this example is for Unix):

```
C:\lm-x\win32_x86\no_security_config> ..\lmxdev.exe -embedsecurityconfig ../../config/xformation.lmx
liblmx-MyVendorName.so
LM-X Developer Tool v3.3
Copyright (C) 2002-2016 X-Formation. All rights reserved.
Licensed to X-FORMATION
Security configuration successfully embedded!
```

2. Use the client library together with your application.

Note that failure to embed the security configuration file in the client library will result in license checkout failures.

# HostIDs

A HostID is a unique machine value, which can be used to lock a license file to a specific host.

LM-X provides a great deal of flexibility to give you fine-grained control over your licensing policy. For example, a license might be:

- Locked against a particular license server host (or license server hosts), so any client machine can use your application when using a license from the specified license server host(s).
- Locked against a client machine (or machines), so your application will run successfully if it is running on the particular machine(s) for which the license has been granted.
- Locked against both license server host(s) and client machine(s), so your application will run successfully if it is using a license from the specified license server host(s) and is running on the particular machine(s) for which the license has been granted.
- Using HostID match rate, a license may be locked to multiple HostID values but allow users to run the software if a specified percentage of the HostID values are valid. See [HOSTID\\_MATCH\\_RATE](#) for more information.

**Note:** For performance reasons, locking network licenses to the client's HostID(s) is disabled by default. To lock licenses to HostIDs, you must enable [LMX\\_OPT\\_CLIENT\\_HOSTIDS\\_TO\\_SERVER](#).



# HostID values

*The information on this page refers to LM-X v5.2 and newer, which added support for Linux ARM and increased support for HTTP-based HostIDs to include all platforms. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

For maximum flexibility in your licensing options, you can define the HostID values for your protected application within the license rather than inside the protected application. This enables you to license your application differently for different users, without requiring your application to be recompiled.

**Note:** The maximum length of a HostID value is 256 bytes.

You can use the `lmxendutil` utility or the LM-X End-user Configuration Tool to print out the valid HostIDs for a particular machine.

The following HostIDs are supported in the license:

HostID Type	Description	Platform Availability
LMX_HOSTID_ETHERNET	Network card HostID	All
LMX_HOSTID_USERNAME	Username HostID	All
LMX_HOSTID_HOSTNAME	Hostname HostID	All
LMX_HOSTID_IPADDRESS	IP address HostID	All
LMX_HOSTID_CUSTOM	Custom HostID	All
LMX_HOSTID_DONGLE_HASPHL	HaspHL Dongle HostID	Windows (x86 and x64) Linux (x86, x64, and ARM) Mac OS
LMX_HOSTID_HARDDISK	HostID of physical harddisk	Windows (x86 and x64; <i>not</i> available for use with MinGW compiler)
LMX_HOSTID_LONG	System-specific HostID	Mac OS
LMX_HOSTID_BIOS	Bios HostID	Windows (x86 and x64; <i>not</i> available for use with MinGW compiler)
LMX_HOSTID_WIN_PRODUCT_ID	Windows Product ID	Windows (x86 and x64)
LMX_HOSTID_AWS_INSTANCE_ID	Amazon EC2 Instance ID	
LMX_HOSTID_GCE_ID	Google Compute Engine ID	
LMX_HOSTID_AZURE_ID	Azure ID	

For help on deciding which HostID or combination of HostIDs fits your needs, see [Determining which HostID to use](#). For complete information about the operating system versions that are supported by LM-X, see [Supported platforms](#).

## LMX\_HOSTID\_ETHERNET

For license generation, LM-X can handle different Ethernet HostID formats in the license template based on the operating system representation of the address when using `ipconfig` (Windows) or `ifconfig` (Unix); for example, AABCCDDDEEFF, AA:BB:CC:DD:EE:FF or AA-BB-CC-DD-EE-FF.

## LMX\_HOSTID\_DONGLE\_HASPHL

LMX\_HOSTID\_DONGLE\_HASPHL is a HostID representing physical dongles. You may use 3rd-party dongles, which require some custom programming, or purchase dongles from X-Formation (provided by Aladdin) that work out of the box, without need for further customization. While dongles provide the best possible security of the aforementioned HostIDs, they come with additional distribution overhead.

To use dongles on Windows, you should instruct your end users to plug in the dongle and let Windows find the device driver automatically using Windows Update. For other platforms, you can [download the appropriate device driver from our website](#).

When LMX\_HOSTID\_DONGLE\_HASPHL is used with a license server, background checking is done to ensure that users don't remove the dongle from the server machine when serving licenses. Doing so will cause the license server to stop functioning after a short time.

When using LMX\_HOSTID\_DONGLE\_HASPHL with local standalone licenses, the client application should ensure that the dongle is not removed during client runtime.

You can check the dongle by calling [LMX\\_Heartbeat](#) on a separate thread continuously, or by ensuring that LMX\_HOSTID\_DONGLE\_HASPHL HostID is in use and comparing the HostID value against the last known good value, as demonstrated in the following example.

```

int i;
LMX_FEATURE_INFO FI;
LMX_HOSTID LmxHostid[LMX_MAX_HOSTIDS];
int nHostids;
LMX_GetFeatureInfo(LmxHandle, "my_app", &FI);
/* Go through each hostid used for this particular feature */
for (i = 0; i < FI.nClientLicenseHostids; i++)
{
    /* See if the hostid is a dongle hostid */
    if (FI.ClientLicenseHostid[i].nHostidType == LMX_HOSTID_DONGLE_HASPHL)
    {
        /* See if hostid function reports the same value as we used when checking out the license */
        if (LMX_Hostid(LmxHandle, LMX_HOSTID_DONGLE_HASPHL, LmxHostid, &nHostids) != LMX_SUCCESS)
            return BAD_DONGLE;
        if (nHostids != 1)
            return BAD_DONGLE;
        /* Compare if the hostid at checkout time matches the hostid at present time */
        if (strcmp(FI.ClientLicenseHostid[0].szValue, LmxHostid[0].szValue) != 0)
            return BAD_DONGLE;
    }
}

```

This check should preferably be done every few minutes to ensure that users do not move the dongle to other workstations and overuse standalone licenses.

See [LMX\\_Hostid](#) for further information on LMX\_Hostid.

## Using custom HostIDs

To make use of custom HostIDs, you must set a callback function using `LMX_SetOption` with the flag `LMX_OPT_CUSTOM_HOSTID_FUNCTION`. See [LMX\\_OPT\\_CUSTOM\\_HOSTID\\_FUNCTION](#) in [LMX\\_SetOption](#) for more information.

You should ensure that you complete the `LMX_HOSTID` structure correctly and return the appropriate error code, as shown in the following example.

```
LMX_STATUS LMX_CALLBACK MyHostidCallback(LMX_HOSTID *pHostid, int *npHostids)
{
    printf("My HostID called\n");
    /* Set the length of the data of this HostID. */
    /* Its length can be up to LMX_MAX_SHORT_STRING_LENGTH */
    strcpy(pHostid[0].szValue, "abcd");
    /* Set optional description. */
    /* Its length can be up to LMX_MAX_SHORT_STRING_LENGTH */
    strcpy(pHostid[0].szDescription, "my HostID");
    /* Now we do it once more for a 2nd HostID. */
    strcpy(pHostid[1].szValue, "12345");
    strcpy(pHostid[1].szDescription, "my HostID2");
    /* Set that we return 2 custom HostIDs. */
    *npHostids = 2;
    /* The function must return either LMX_SUCCESS or LMX_UNKNOWN_ERROR. */
    return LMX_SUCCESS;
    /* return LMX_UNKNOWN_ERROR; */
}
```

The same callback function is also required to use custom HostIDs on a license server. In `lmx_server_conf.c` you can set the custom HostID callback function to create licenses that are locked to a license server with a custom HostID.

You can easily use multiple HostID sources for such a custom callback function. Either fill the array with the independent values or use a cryptographic hash function such as SHA-1 to combine the HostIDs into a single value.

## Custom HostID compare

Custom HostID compare lets you lock your software to one or more HostID values using advanced comparison. With this method, you put multiple HostIDs in the license file, but use them under custom conditions. For example, you might create a condition to lock to the Ethernet card if it is available, and if it is not available, lock to the harddisk. Another example is to include all available information about the machine in the license file, and then allow the software to run if the harddisk and Ethernet HostIDs match, and ignore all other information.

For example, you might lock a feature to the license server machine hostname and at least one of two HostID adapters, instead of requiring that all three HostID values match for the license to be valid.

You cannot call API functions from the HostID callback functions. This will return the error LMX\_API\_NOT\_REENTRANT.

See [LMX\\_OPT\\_HOSTID\\_COMPARE\\_FUNCTION](#) in [LMX\\_SetOption](#) for details on setting this option.

# LM-X security configuration file

To protect your application, you need a security configuration file (`security_config.lmx`). This file holds various security-related parameters, and is generated by the [developer tool `lmxdev`](#).

The LM-X security configuration file is also used to create license files.

**Important:** You must reuse this security configuration file for the lifetime of your application.

Note the following:

- When updating to new builds of LM-X or compiling LM-X on multiple platforms, the installer will let you choose an existing security configuration file. If you do not choose an existing security configuration file while running the installer, you must manually place the file in the config directory after installing the SDK.
- Upon first compilation of the LM-X SDK, the LM-X security configuration file (`security_config.lmx`) is automatically created in the config directory of the SDK.

## LM-X tools

LM-X consists of several tools. Those intended for use by the end user are detailed separately in the [LM-X End Users Guide](#). The following section describes the [lmxdev developer tool](#).

# Lmxdev developer tool

The lmxdev developer tool is a multipurpose utility for generating security configuration files and includes files for compilation.

The lmxdev command usage is as follows:

- genconfig security\_config.lmx
- embedlicense lmx.lic file
- embedsecurityconfig security\_config.lmx file
- verifysecurityconfig file

Where:

-genconfig security_config.lmx	Generates an LM-X security configuration file ( <i>security_config.lmx</i> ) that holds developer-specific security parameters.
-embedlicense lmx.lic file	Embeds an LM-X License Manager license into the license server executable or the client library.
-embedsecurityconfig security_config.lmx file	Embeds security configuration into the license server executable or the client library.
-verifysecurityconfig file	Verifies if the security configuration is embedded.

**Note:** embedsecurityconfig and verifysecurityconfig additionally include extended versions (embedsecurityconfigext and verifysecurityconfigext, respectively) that allow you to embed/verify also private keys.

## Distributing applications to end users

This section gives an overview of both required and optional files to be distributed to end users.



# End-user file distribution

The following files should be distributed to end users:

Component		Description
Windows Filename(s)	Unix Filename(s)	
lmx-serv.exe lmx-serv.cfg	lmx-serv lmx-serv.cfg	License server for end users along with its configuration file. The license server can either be run from a console or as a service (on Windows) or daemon (on Unix).
liblmxvendor.dll	liblmxvendor.so/dylib	Vendor-specific file containing all vendor-implemented extensions to the license server. This file <i>must</i> be distributed to end users along with the license server file.  On Windows, the liblmxvendor.dll file must be placed in the same directory as the license server.  On Unix, the startup scripts provided in <i>\$platform/enduser</i> should be used for starting the LM-X License Server.
lmxendutil.exe	lmxendutil	End-user command line utility.
lmxconfigtool.exe	N/A	End-user GUI configuration utility (Windows only).

In addition, if a user has problems with the system clock, you may send the user the LmxResetSystemClock tool. See [System clock check](#) for more information about this tool.

With most installers under Windows, you can automate the license server registration as a service, because the license server is installed through a set of command line parameters. Also see [Performing a silent installation](#).

To automate license server registration, make sure your installer first copies the file to its destination, and then executes it with the appropriate parameters to [install it as a service](#). If the installation is successful, the return code is zero; otherwise, it is non-zero.

Further details on the usage of these tools can be found in the [LM-X End Users Guide](#).

## LM-X license files

The license files can easily be sent over e-mail. However, bear in mind that some email clients may change the content of received e-mails, which may affect the license. Additional new lines will not affect LM-X, but changes such as replacing special characters may affect the LM-X license and could change calculated checksums, thus invalidating the license.

The license files, when saved to disk, can be named anything that the operating system supports and do not need to have a .lic file extension. However, you should ensure that files are written as ASCII text files and are not encoded in any particular way, such as in Rich Text Format, Word or similar formats.

If the license file being shipped is to be a local license, we recommend that you set a predefined license path for where the application program should look for licenses, as shown in the following example.

On Windows systems:

```
LMX_SetOption(..., LMX_OPT_LICENSE_PATH, (LMX_OPTION) "licenses application.lic");
```

On Unix systems:

```
LMX_SetOption(..., LMX_OPT_LICENSE_PATH, (LMX_OPTION) "licenses/application.lic");
```

This example sets the preferred license path to the file *application.lic* in the *licenses* subdirectory of the current directory path.

If the license is for network use, you should instruct users on how to install the license server and get it up and running.

# System clock check

*The information on this page refers to LM-X v4.5.2 and newer, which added the ability to use a remote time server for SYSTEMCLOCKCHECK and changed functionality for the LmxResetSystemClock tool. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

LM-X can perform a clock check on the client side when calling `LMX_Checkout()` to ensure users have not backdated or tampered with their system clock. If tampering is detected, the user will be unable to check out the license and will be given an error regarding the invalid system clock time. The license server does a background check every 10 minutes, and if the clock has been tampered with, the license server shuts down.

This helps prevent unauthorized use of the software; for example, a user attempting to extend the end date of a trial license. Note that any clock discrepancy of less than 24 hours will not be detected as tampering. For example, a user can change the clock to reflect a different time zone without causing an error.

A system clock check is done using the local file system by default, depending on the license type as shown in the following table. You may change the system clock check to check time on a remote Network Time Protocol (NTP) time server instead of the local file system, or you can disable system clock check. For more information, see [FEATURE settings](#).

License Type	Default System Clock Check Behavior
Local	Check system clock only when end date is set. Otherwise, do not check system clock.
Network	Do not check system clock.
Trial	Check system clock.
Grace	Do not check system clock.
Borrow	Do not check system clock.

The system clock check evaluates several criteria to determine whether the system time has been tampered with. Under some circumstances, the check can return a false positive. For example, a user might accidentally change the clock to a future time. Normally, the user can reset the system clock to the correct time to resolve the issue; however, in some cases this may not work.

To resolve such individual cases, you can send the user the `LmxResetSystemClock` tool (included in the LM-X distribution). The user can run this tool to fix the problem in most cases. A connection to the internet is required when using the `LmxResetSystemClock` tool in order to verify that the clock is set to the correct date. If an incorrect date is detected, a warning message will instruct the user to set the correct date and re-run the tool. The tool requires the clock to be correct before applying fixes, so can be run any number of times. The following shows an example of running the `LmxResetSystemClock` tool:

```
C:\lm-x>lmxresetsystemclock.exe
LM-X Reset System Clock Utility v4.5.2
Copyright (C) 2002-2013 X-Formation. All rights reserved.
```

```
Number of fixes: 2
```

# Trial licenses

Trial licenses let end users use an application for a predetermined length of time or number of uses without requiring a license. This feature can be useful for eliminating the need to supply licenses with evaluation software distributed to a large number of anonymous users.

Trial licenses work similarly to all other license types and do not require that you change your procedures. However, trial licenses are limited and will not contain the same type of information as normal licenses. For example, [LMX\\_GetFeatureInfo\(\)](#) will exclude most fields.

You do not create a .lic file for trial licenses. Instead, you simply specify the number of trial days or uses as described below.

## Enabling trial licenses

To enable trial licenses, you must set the predefined trial length (see [LMX\\_OPT\\_TRIAL\\_DAYS](#) in [LMX\\_SetOption\(\)](#)) and/or the number of checkouts allowed (see [LMX\\_OPT\\_TRIAL\\_USES](#) in [LMX\\_SetOption\(\)](#)).

A trial license begins when the application is run for the first time and expires after the specified number of days or checkouts is reached (whichever comes first), after which the end user can no longer use the trial license. However, if the end user has a local or network license, it will supersede the trial license.

After you enable trial licenses, requests to [LMX\\_Checkout\(\)](#) will succeed for trial license checkouts. Trial licenses will be available only if no other licenses exist.

It is important to specify the version number in [LMX\\_Checkout\(\)](#). For example, say you have designed your trial version 2.1 to expire in 14 days. The trial can be run only once, so if a customer needs a second trial, you must increase the version number in [LMX\\_Checkout\(\)](#), say to 3.1. For example: (LMX.Checkout "Trial", 3, 1, 1). This way, the customer can install and run the trial again.

## Limitations and cautions

By default, trial licenses are not allowed on virtual machines or terminal servers. You may allow trial licenses to run on virtual machines or terminal servers by using [LMX\\_SetOption\(\)](#); see [LMX\\_OPT\\_TRIAL\\_VIRTUAL\\_MACHINE](#) and [LMX\\_OPT\\_TRIAL\\_TERMINAL\\_SERVER](#) in [LMX\\_SetOption\(\)](#).

Be aware that on most operating systems, the locations you can write files to are limited, and there is a risk that someone can delete the files intentionally or unintentionally.

Trial licenses are less secure than other license types; for example, licenses with an expiration date. This is not due to LM-X trial license limitations, but rather operating system limitations on where you can securely store data. There is always a risk that dishonest users will find a way to reset trials without your consent.

For these reasons, you should consider using additional security measures for trial licenses, such as establishing a policy of when trial licenses are used, doing additional verifications as needed, etc.

## Improving the security and usefulness of trial licenses using LAC

To enable trials with increased security, you may consider using [License Activation Center](#) (LAC). For example, by integrating trials with your website to allow users to sign up for a trial key online. This lets you offer your trials using a license with an expiration date.

LAC also gives you access to statistics about evaluations. With LAC, you can see who is and is not activating licenses, giving you business intelligence that helps you understand more about the users evaluating your software.

## Restarting a trial

If you need to restart a trial for any reason (for example, to allow the user more time to evaluate the software, or to fix errors that may rarely occur due to secure store corruption), you can send the user the `LmxResetTrial` tool.

The `LmxResetTrial` tool cleans up the trial information and lets the user re-run the trial. The `LmxResetTrial` tool can be run only once. Subsequent attempts to run the tool will report the message "XF Error: This utility can be used only once!"

## Automatic server discovery (floating licenses only)

Automatic server discovery (for floating licenses only) allows a client application to find license servers on the network automatically. This feature eliminates the need for end users to enter server information and makes it easy for system administrators to move from one license server to another without notifying users.

Automatic server discovery sends UDP broadcasts on a fixed port: 6200. Available license servers respond to the broadcast with information about their location on the network, which the clients then use to proceed with traditional TCP communication.

Typically, the time to perform a server discovery can range from 500 ms to 1 second during a call to `LMX_Checkout()`. When a license server is discovered, the information is locally cached on the client machine, so automatic server discovery does not have to be performed at each application startup.

Automatic server discovery works only on local networks and will not work on WAN or VPN connections. Automatic server discovery is not guaranteed to work on all networks, particularly enterprise networks on which local firewalls or routers cut off UDP broadcast traffic.

By default, automatic server discovery is disabled on the client side. Automatic server discovery can be enabled by the end user as described in [Controlling license behavior](#).

# Heartbeats

Maintaining the connection between the client and server helps to avoid potential license overuse. Therefore, implementing heartbeats and the processes that should result from receiving unexpected heartbeat responses is important for network licensing.

LM-X offers [manual](#) and [automatic](#) heartbeats, which are described in the next sections. Automatic heartbeats are invoked with a specified interval in a separate thread by the LM-X client. Manual heartbeats can be called with any interval and do not require a separate thread.

While the heartbeat is invoked it checks the connection to the license server(s). If the server does not respond to heartbeats, your application will behave in the manner you specify. For example, you may specify that licensed applications cease functioning when the server becomes unavailable.

Heartbeats are also used by the server in order to establish whether the client is still active. If no heartbeat is received from an application instance within the configurable timeout, any other license server for that feature will be automatically used if available.

The client should send heartbeats at an interval shorter than the license server's timeout setting to ensure that it keeps its connection. The default server timeout is 5 minutes, and it is recommended that client applications send heartbeats every 1-2 minutes.

If there is problem with the server connection, the client will try to reconnect to the server and re-checkout the feature, and the status will be passed to the callback functions, as described in [Callback functions](#).

## Automatic heartbeats

Automatic heartbeats are for network licenses only, and are disabled by default. You can enable automatic heartbeats using the [LMX\\_SetOption](#) option `LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS`.

The recommended setting for automatic heartbeats is between 30 seconds and 2 minutes. (The default interval is 2 minutes, which can be changed using the [LMX\\_SetOption](#) option `LMX_OPT_AUTOMATIC_HEARTBEAT_INTERVAL`.)

Automatic heartbeats start an extra thread that calls the license server on behalf of the client at the specified intervals. If the license server fully restarts between heartbeats from an application, automatic heartbeats will recognize that the server has restarted and will attempt to re-checkout its required license(s).

Until the failed heartbeats counter value equals `LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS`, the client will only attempt to reconnect to the server from which the feature was originally checked out. Only after the counter value is reached will the client attempt to connect to other servers.

If the checkout is not possible (for example, the remainder of available licenses have been checked out in the meantime), this information will be relayed to the application, which will behave in the manner specified. If, for example, five heartbeats in a row fail, automatic heartbeats will call an `exit()` function on behalf of the application and print error information to the screen. This behavior can be overridden with the `HEARTBEAT_EXIT_FUNCTION` callback, described in [Heartbeats](#).

### Using a counter to prevent license overuse

When the server is restarted it takes a few minutes for all clients to reconnect. Particularly for applications that run for short periods of time, license overuse may occur if the license server is stopped and restarted frequently.

To prevent intentional license overuse, you can implement a custom-coded counter. Using [LMX\\_OPT\\_AUTOMATIC\\_HEARTBEAT\\_ATTEMPTS](#) together with [LMX\\_OPT\\_AUTOMATIC\\_HEARTBEAT\\_INTERVAL](#), you can detect broken network connections. By implementing a custom counter in addition to these API calls, you can track the number of times the license server goes down and comes back up during a specified interval, and stop the application if a particular frequency is exceeded. For example, the counter could limit the number of times the license server goes down to 10 times per hour before making the application unavailable.

When considering implementing such a counter, be aware that an unstable network can cause the connection to the server to go down often, which will cause the application to stop. The need to implement a counter should be based on the reliability of the network and your trust in your customer.

## Manual heartbeats

Manual heartbeats are network licenses only and are implemented by default. You can send manual heartbeats by calling the API function [LMX\\_Heartbeat\(\)](#) from your application.

Manual heartbeats work the same as automatic heartbeats, with the retry attempt count set to 1. However, when manual heartbeats fail, the HEARTBEAT\_EXIT\_FUNCTION is not called, as it would be if automatic heartbeats fail with a count set to 1.



# Callback functions

If every connection is established, no callbacks are called and no actions are taken.

If there is problem with the server connection, the client will try to reconnect to the server and re-checkout the feature, and the status will be passed to the callback functions. The callback functions let you monitor various scenarios, such as which licenses are taken by other clients during network connection outages. In such cases, you could set flags or otherwise notify parts of your application that some licenses have been lost. Note that you do not necessarily need to set up all of the callback functions, but instead use only those needed to modify the default behavior.

The heartbeat callback functions can be enabled with `LMX_SetOption()`. **Note:** Attempting to call API functions from the heartbeat callback functions will return the error `LMX_API_NOT_REENTRANT`. This includes calling `LMX_GetErrorMessage`; however, you can call `LMX_GetErrorMessageSimple` instead.

Possible callback functions are:

Name	Status Information Returned
HEARTBEAT_CONNECTION_LOST_FUNCTION	Connection with license server was lost.
HEARTBEAT_RETRY_FEATURE_FUNCTION	Reconnection with license server succeeded. Re-attempt to checkout feature from license server.
HEARTBEAT_CHECKOUT_FAILURE_FUNCTION	License lost; could not be regained from license server.
HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION	License retained successfully.
HEARTBEAT_EXIT_FUNCTION	The automatic heartbeats thread calls this callback function when it has given up reconnecting to the license server. When using this callback, ensure you set a proper exit flag during the callback and then shut down or otherwise cease functionality of the application.

The callback functions work as follows:

In case of failure, the `HEARTBEAT_CONNECTION_LOST_FUNCTION` callback is always called. At the next heartbeat, if the connection is lost, the client tries to reconnect to the license server from which the feature was checked out. The number of reconnection attempts can be specified by setting the `LMX_OPT_AUTOMATIC_HEARTBEATS_ATTEMPTS` option.

After the last unsuccessful attempt to reconnect to the same server, the client will then try to connect to other servers. When the client successfully reconnects to the same server or a different server, the `HEARTBEAT_RETRY_FEATURE_FUNCTION` callback is called, and the client attempts to re-checkout the feature. The re-checkout will succeed only if the feature that is being renewed has a version equal to or less than the version of the feature that is currently being served by the server. To specify that the feature version being renewed must exactly match the feature version currently being served by the server, you must enable `LMX_OPT_EXACT_VERSION`.

The `HEARTBEAT_RETRY_FEATURE_FUNCTION` callback may be called many times, because LM-X will not stop retrying the checkout until all possible license servers have been tried. Every `HEARTBEAT_RETRY_FEATURE_FUNCTION` is followed by either `HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION` or `HEARTBEAT_CHECKOUT_FAILURE_FUNCTION`.

- If re-checkout succeeds, the `HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION` callback is called.
- If re-checkout fails (for example, when there are no licenses left for checkout), `HEARTBEAT_CHECKOUT_FAILURE_FUNCTION` is called.
- The `HEARTBEAT_EXIT_FUNCTION` callback is called only if the heartbeat count is exhausted.

## Examples

The following examples describe callback function behavior when heartbeats are lost.

Available Servers	Client Action	Type of Heartbeats Implemented	Callback Function Behavior When Connection Is Lost
S1	CHECKOUT feature from S1	Automatic	<i>(LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS set to 3)</i> <i>(S1 goes down)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION <i>(S1 goes up)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION

S1, S2, S3	CHECKOUT feature from S1	Automatic	<i>(LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS set to 5)</i> <i>(S1 goes down)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_CONNECTION_LOST_FUNCTION <i>(S2 is up but has no features left)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_FAILURE_FUNCTION <i>(S3 is up)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION
S1, S2, S3	CHECKOUT feature from S1	Manual	<i>(S1 goes down)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_FAILURE_FUNCTION <i>(S2 is up but has no features left)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION <i>(S3 is up)</i> HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION
S1 and S2	CHECKOUT features A and B from S1  CHECKOUT feature C from S2	Automatic	<i>(LMX_OPT_AUTOMATIC_HEARTBEATS_ATTEMPTS set to 3)</i>  <i>(S1 and S2 go down)</i> <i>(on S1:)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION <i>(on S2:)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION <i>(S1 goes up)</i> <i>(for feature A:)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION <i>(for feature B:)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION <i>(on S2:)</i> HEARTBEAT_CONNECTION_LOST_FUNCTION <i>(S2 goes up)</i> <i>(for feature C:)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION
S1 and S2	CHECKOUT feature from S1	Automatic	<i>(LMX_OPT_AUTOMATIC_HEARTBEATS_ATTEMPTS set to 3)</i>  <i>(S1 goes down)</i> CONNECTION LOST <i>(S1 goes up, but no features are available)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_FAILURE_FUNCTION HEARTBEAT_RETRY_FEATURE_FUNCTION <i>(this is the 3rd heartbeat)</i> HEARTBEAT_CHECKOUT_FAILURE_FUNCTION <i>(on S2:)</i> HEARTBEAT_RETRY_FEATURE_FUNCTION HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION

## License queuing

License queuing helps your users implement fair usage when licenses may not be immediately available. This feature is particularly useful for jobs scheduled for automatic execution. Without queuing enabled, checkout requests are immediately denied if the required license(s) are unavailable. If queuing is enabled, you determine its behavior; for example, the applications may be placed on hold until the necessary license(s) become available. (See [LMX\\_Checkout](#) for more information about how to implement license queuing behavior.)

License queuing has two variations: normal (the default) or fast queuing. By default, all license requests are appended to the end of the queue, regardless whether the request can be fulfilled immediately.

Alternatively, end users can enable fast queuing for their license server by changing a setting in their license server configuration file. Fast queuing allows requests that can be fulfilled immediately to be fulfilled.

For example, if a client is waiting for two licenses, and only one license is immediately available, another client that needs only one license can bypass the queue and take the single license without waiting. In this way, fast queuing allows smaller license requests to be processed more promptly and helps ensure higher license utilization. However, because it might enable users to bypass the queue, it does not necessarily implement fairness.

License queuing can be enabled by the end user by setting an environment variable. If you do not want to make license queuing available to end users, you can unset the environment variable as described in [Controlling license behavior](#). [Environment variables](#) are described in detail in the [LM-X End Users Guide](#).

## Secure store

Secure store is a technology used to enable a more fail-safe means of storing sensitive license content on a user's machine.

Traditionally, a single file location or registry key is used within applications to store information such as usage counters, trial information and other client-specific content. Conversely, secure store works similarly to a virtual file system, by encrypting the content and then duplicating it to provide redundancy.

This method of storage is more resistant to single-point-of-failure attacks, which are a common method for attempting to reset trial information within software applications. Furthermore, the path of storage is uniquely defined for each user's machine, making it more resistant against generic attacks.

Within LM-X, secure store is used with [borrow](#), [grace](#) and [trial](#) licenses and for [client store](#).

For maximum flexibility and safety, secure store technology is designed to be used per user, i.e. by the currently logged-in user of the machine.

Note that secure store might not migrate successfully from an older version of LM-X to a newer version. For example, if you use LMX\_ClientStoreSave with an older LM-X version and then attempt to use LMX\_ClientStoreLoad on the same variable with a newer LM-X version, you may get the error LMX\_FILE\_READ\_ERROR.

**Note:** Upon first compilation of the LM-X SDK, the LM-X security configuration file (security\_config.lmx) is automatically created in the config directory. This file holds various security-related parameters, and is used to create license files and embed the security configuration into the license server and client libraries.

## Client store

Client store builds on [secure store](#) by supplying a means of storing user content more securely. The user content can be stored in a virtual file system, which uses secure store to encrypt, uniquely store, and duplicate (for redundancy) the content on the user's machine. The default locations where the files are stored differ depending on an operating system. There is no way of removing all files, but you can delete individual pieces of content stored using LM-X API.

Client store is useful for storing trial information, such as usage counters and acting as a file system to support the persistence of the stored content. This makes client store a reliable means to help ensure your licensed software is used only as you intend.

See [LMX\\_ClientStoreSave](#) and [LMX\\_ClientStoreLoad](#) for details about using the client store API functions.

## Blacklisting issued licenses

Blacklisting lets you stop issued licenses from working in future versions of your application, normally to allow users to move licenses from one host to another. Blacklisting applies to a specific feature in the license file, based on the unique KEY value of the feature.

You can use blacklisting for local, network and borrow licenses. Blacklisting of a network license must be done on a license server machine.

When a blacklisted license is checked out and is then checked back in, the feature will thereafter be unavailable for checkout. For blacklisted local licenses, checkout attempts will result in the message "Feature is blacklisted." For blacklisted network licenses, checkout attempts will be treated as if the feature does not exist, resulting in the message "Feature not found."

When replacing a blacklisted license file with a newly generated license file (re-hosting), the feature will be available for checkout again.

To blacklist and re-host a license:

1. To revoke an existing license, set the [LMX\\_SetOption](#) flag [LMX\\_OPT\\_BLACKLIST](#).
2. To re-host a blacklisted license, replace the blacklisted license file with a newly generated license file.

Using the new license file, the feature will once again be available for checkout. Blacklisting can be verified using the end user utility `lmxendutil`, which can print out a list of blacklisted licenses.

**Note:** To replace one license with another license, you can also use the [REPLACES](#) directive. If network server re-hosting is frequent, you can manage re-hosting in a different way, such as locking the license to a dongle (see [HostID values](#)).

# Licensing for virtual machines and cloud computing

*The information on this page refers to LM-X License Manager 4.9.5 or newer, which introduced support for KVM and QEMU virtualization detection. If you are running a previous version of License Statistics, please see [documentation for previous versions](#).*

Licensing your software to run on virtual machines or in a cloud environment carries special considerations, as described below.

## Virtual machine licensing

You can install a license server on a virtual machine. To do this, enable support for virtual machines in the license file.

By default, LM-X denies all checkouts for local licenses in virtual environments, and refuses to load licenses on license servers to prevent potential license overuse. Virtualization detection is officially supported for VMware, Microsoft Hyper-V, VirtualBox, KVM and QEMU.

Virtual machine environments are typically undesirable for use with software licensing. Normally, operating systems running in virtual environments are designed to be hardware independent, so Ethernet or harddisk HostIDs no longer specify physical hardware identifiers. Duplicating the virtual machines and modifying the HostIDs is simple and takes minutes to perform.

However, you may choose to explicitly allow your application to run in a virtual environment (for example, limited to use by trusted and enterprise customers) by specifying the SHARE keyword VIRTUAL for the particular feature (see FEATURE settings for details on setting the virtual share option in your license.)

You may also allow trial licenses to run on a virtual machine using `LMX_OPT_TRIAL_VIRTUAL_MACHINE`, described in `LMX_SetOption`.

There are a number of ways that LM-X makes it safe for you to deliver your application for use on virtual machines:

1. *Use floating licenses.* Your software can be safely allowed to run on a virtual machine, because the license server, which is located on a physical machine, will ensure there is no license overuse.
2. *Use dongles.* If your customer wants a license server running on a virtual machine, you can lock the license server to a dongle, which is a physical device that locks the application to the machine on which it's installed. Dongles, which may be obtained from X-Formation, provide adequate security to prevent cloning of the license server.
3. *Lock the license to a BIOS HostID.* This solution works for both local licenses and network licenses. The BIOS on most virtual machines is unique, because it contains a unique virtual machine ID (UUID). An example is shown below:

```
LM-X End-user Utility v3.4
Copyright (C) 2002-2016 X-Formation. All rights reserved.
BIOS: Phoenix Technologies LTD - UNKNOWN
Hostid: VMware-42321a30c22ce364-aca97bac6ea0bdb8
```

Note that the UUID can easily be changed, so this is not an entirely secure solution. However, if the virtual machine is connected to a management control solution (such as VMware vCenter), duplication of UUID's is typically not permitted and causes problems for real-life production setups.

## Licensing for cloud applications

When using Amazon EC2, Microsoft Azure, or Google Compute Engine (GCE), you can use the Instance ID for the HostID (see [HostID values](#)). However, for other cloud service providers, virtual machines do not have HostIDs. To solve this problem:

- As recommended above for virtual machines, it is highly recommended to use floating licenses.
- Ask your customer to run a license server either inside or outside of the cloud environment. If your application does not require a high number of checkouts/checkins over short time intervals, it is recommended to connect the application to the existing license server on the end-user network.
- As an alternative to having the customer run a license server, you can host the license server for the customer.

# Token-based licensing

Token-based licensing lets you specify that a particular feature should use one or more other licenses to fulfill checkout requests. Token-based licensing applies only to network licenses.

A token-based license is a pointer, or reference, to a real license. This enables you to specify any number of individual token-based licenses that make use of the same real license.

When the license server gets a request for a token-based license, it uses one or more other licenses (as specified in the license file) to fill the request, rather than directly using a license for the originally requested feature. The feature that is used to allow the checkout is referred to as a *token dependency*.

When a normal network license is checked out, the license appears in the license file with a count. For example:

```
FEATURE MyFeatureA
{
  VENDOR = MyCompany
  COUNT = 5
  KEYTYPE = EXCLUSIVE
  MAJOR_VERSION = 1
  MINOR_VERSION = 0
  ...
}
```

However, when a token-based license is checked out, it appears in the license file without a count, and instead includes a dependency on one or more other features that are used to allow the checkout. For example:

```
FEATURE MyFeatureB
{
  VENDOR = MyCompany
  KEYTYPE = TOKEN
  MAJOR_VERSION = 1
  MINOR_VERSION = 5
  TOKEN_DEPENDENCY = "FEATURE=MyFeatureA VERSION=1.0 COUNT=5"
  ...
}
```



# Uses for token-based licensing

*The information on this page refers to v5.1 and newer, which banned sharing both tokens and their dependencies at the same time. For documentation applicable to earlier versions, see [documentation for previous versions](#).*

There are many different ways you can use token-based licensing, as described in the following examples.

## License pools

One primary use for token-based licensing is to let users purchase a number of pseudo-features that each require one or more real licenses. This gives users a "pool" of licenses they can draw upon for license checkout requests, providing the flexibility to use various combinations of features as their needs require.

For example, say that the product suite MySolutions consists of three individually sold features: MyDraw, MyWrite, and MySpreadsheet. ABC Corp purchases 20 token-based licenses of MySolutions, including all three applications. Whenever there is a checkout request for one of these three applications, a particular number of licenses is taken from the 20-license pool for the MySolutions suite: MyDraw requires 5 licenses for each checkout; MyWrite requires 2 licenses, and MySpreadsheet requires 1 license.

As you can see from the two different scenarios below, a pool of token-based licenses gives customers significantly more flexibility than purchasing a specific number of licenses for each feature. The users can check out any combination of the features, up to the maximum 20-license limit.

### Example Scenario 1

Feature	# of Licenses Consumed per Checkout	# of Checkout Requests	Total Licenses Consumed
MyDraw	5	2	10
MyWrite	2	3	6
MySpreadsheet	1	3	3
<b>Totals</b>		<b>8</b>	<b>19</b>
<b>Number of licenses remaining</b>			<b>1</b>

### Example Scenario 2

Feature	# of Licenses Consumed per Checkout	# of Checkout Requests	Licenses Consumed
MyDraw	5	1	5
MyWrite	2	4	8
MySpreadsheet	1	7	7
<b>Totals</b>		<b>11</b>	<b>20</b>
<b>Number of licenses remaining</b>			<b>0</b>

### Example

The following example shows a token-based license for the license pool scenario described above, where MySolutions is a license pool that is drawn upon to fulfill license requests for MyDraw, MyWrite and MySpreadsheet.

```

FEATURE MySolutions
{
  VENDOR=ABC_Software COUNT=20 KEYTYPE=EXCLUSIVE VERSION=1.0
  KEY=bhq3ed873qcrKHG6783rhJgvkhvTUtxcuBiouVtyCuyVy78Gftq...
}

FEATURE MyDraw
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=MySolutions VERSION=1.0 COUNT=5"
  KEY=b978bv5ybui7Noyg6c3Vd57fngN987NGSC54sDiugU6v5eio8g...
}

FEATURE MyWrite
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=MySolutions VERSION=1.0 COUNT=2"
  KEY=yig*bv7tu6r879yu09yut75evbGJvHGHdrCHJVJGct79g78gvv...
}

FEATURE MySpreadsheet
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=MySolutions VERSION=1.0 COUNT=1"
  KEY=hygvCTYGg6r67fg890hbyvGTCVKJBjhc5r7y9joiVGckjlnut8...
}

```

## Product suite licenses

Product suite licenses specify that one token-based license depends on multiple real licenses. Product suite licenses enforce a logical AND rule, requiring *a* // licenses to be valid in order to perform a checkout. This is essentially the opposite of license pools, which specify that multiple features depend on a single real license.

For example, MyDraw may be composed of two modules: Sketcher and Printer. A token-based license can specify that in order to use MyDraw, you must have 1 license of each of the two modules.

### Example

The following example shows a token-based license for the product suite license scenario described above.

```

FEATURE Sketcher
{
  VENDOR=ABC_Software COUNT=5 KEYTYPE=EXCLUSIVE VERSION=2.0
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d612hU3vSKT...
}

FEATURE Printer
{
  VENDOR=ABC_Software COUNT=5 KEYTYPE=EXCLUSIVE VERSION=1.5
  KEY=B9Uuz12b2B3v]vcsFBx7qEvcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}

FEATURE MyDraw
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Sketcher VERSION=2.0 COUNT=1"
  TOKEN_DEPENDENCY="FEATURE=Printer VERSION=1.5 COUNT=1"
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBxBx7qEQG1SwXCz8A9dj1g866USKT...
}

```

## Alternate licenses

You can use token-based licenses to allow license requests to be fulfilled by one or more alternate product licenses. This enforces a logical OR rule, since it requires one license *or* another to succeed with a checkout.

For example, instead of providing a real license of MyDraw, you could define a license where MyDraw has a token dependency fulfilled by either Lower\_Priced\_License or Higher\_Priced\_License. Although the alternate token licenses may be more expensive than the real license, the mapping allows users to more reliably access the software they need.

The order of the token-based licenses in the license file determines the order that alternate checkouts are attempted. For example, if the MyDraw features's Lower\_Priced\_License token-based license precedes the Higher\_Priced\_License token-based license, then Lower\_Priced\_License will be preferred for filling checkout requests. Higher\_Priced\_License will be used only if Lower\_Priced\_License is unavailable. End users can change the license order if they desire.

### Example

The following example shows a license for the alternate license scenario described above. Note that the first MyDraw token-based license refers to Lower\_Priced\_License, so this will be the preferred license for MyDraw checkout requests.

```
FEATURE Lower_Priced_License
{
  VENDOR=ABC_Software COUNT=5 KEYTYPE=EXCLUSIVE VERSION=1.0
  KEY=mBpIAWB9Uuz12b2B3v]8GJqW300arlnWmnT0lnZXS0IYdF...
}

FEATURE Higher_Priced_License
{
  VENDOR=ABC_Software COUNT=10 KEYTYPE=EXCLUSIVE VERSION=1.0
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}

FEATURE MyDraw
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Lower_Priced_License VERSION=1.0 COUNT=5"
  KEY=mYsfN30C6ShBYszCq2WVlcpTZXQwkfKJTohkzglwNkle163...
}

FEATURE MyDraw
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Higher_Priced_License VERSION=1.0 COUNT=10"
  KEY=2w66Ng3wVSP6ttmWCc8GyJqW300arlnWmnT0lnZXS0IYdF...
}
```

## Cascading (recursive) licenses

Cascading licenses let you specify a recursive list of token-based licenses, all of which must be available in order to fulfill a checkout request.

For example, say that the application MyWrite depends on one license of MyDraw, which in turn depends on two licenses of Sketcher. With this recursive dependency, MyWrite checkouts will consume two licenses.

The maximum number of recursive dependencies you may define for a token-based license is 16.

The maximum number of dependencies the token-based license may have is 512. (However, there is no limit on the number of token-based licenses you may define.)

### Example

The following example shows a token-based license for the cascading license scenario described above.

```
FEATURE Sketcher
{
  VENDOR=ABC_Software COUNT=10 KEYTYPE=EXCLUSIVE VERSION=1.0
  KEY=2w66Ng3wVSVp6ttmWcC8GyJqW300arlnWmnT01nZXSOIYdF...
}

FEATURE MyDraw
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=2.0
  TOKEN_DEPENDENCY="FEATURE=Sketcher VERSION=1.0 COUNT=2"
  KEY=4i]mYsfN30C6ShBYszCq2WVicpTZXQwkfKJTohkzglwNkle...
}

FEATURE MyWrite
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=2.0
  TOKEN_DEPENDENCY="FEATURE=MyDraw VERSION=2.0 COUNT=1"
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}
```

Token sharing and token dependency sharing

Token-based licenses are always unlimited; therefore, [sharing](#) tokens is unnecessary. However, sharing token dependencies may be useful in some cases.

For example, you may have an application with multiple features that can be sold separately or together. Multiple instances of each feature can be shared on a single host. With token sharing, all instances of the particular feature share licenses, but only amongst themselves, not between the different features.

Token dependency sharing allows the features and their instances to share their available licenses and thereby use the licenses more efficiently in cases where you do not want to count features' licenses independently. Dependency sharing can be implemented using multiple token-based licenses that share a common dependency.

Tokens and their dependencies cannot both be shared at the same time.

Example Scenario

For example, say you have two token-based features, "ModuleA" and "ModuleB," with a common token dependency, "Product." ModuleA requires three Product licenses and ModuleB requires two Product licenses, and there are a total of 10 Product licenses.

*Without sharing implemented*, 2 instances of ModuleA (3 licenses required per instance \* 2 instances = 6) and 2 instances of ModuleB (2 licenses required per instance \* 2 instances = 4) will consume all 10 Product licenses.

*Token sharing* lets you share licenses between multiple instances of ModuleA and ModuleB running on the same host. Using the same example with token sharing implemented, two instances of ModuleA and two instances of ModuleB on a single host will use only five licenses, because the licenses are shared *between each feature's instances*. Two instances of ModuleA share two Product licenses amongst themselves and two instances of ModuleB share three Product licenses among themselves.

*Token dependency sharing* allows for sharing the dependency's licenses freely between ModuleA and ModuleB. Continuing to use the same example with token dependency sharing implemented, two instances of ModuleA and two instances of ModuleB are able to share just three Product licenses.

The example scenario described above is illustrated in the table below.

Sharing Type	Product licenses used for two instances of ModuleA	Product licenses used for two instances of ModuleB
None	6	4
Token sharing	3	2
Token dependency sharing	3	0 (shares 2 licenses with ModuleA)

Token sharing license example

The following example shows a token-based license for the token sharing scenario described above.

```

FEATURE Product
{
  VENDOR=ABC_Software COUNT=10 VERSION=1.0
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}

FEATURE ModuleA
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0 SHARE=HOST
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=3"
  KEY=4i]mYsfN30C6ShBYszCq2WVipTZXQwkfKJTohkzglwNkle...
}

FEATURE ModuleB
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0 SHARE=HOST
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=2"
  KEY=2w66Ng3wVSPp6ttmWCc8GyJqW300arlnWmnT0lnZXSOIYdF...
}

```

### Token dependency sharing license example

The following example shows a token-based license for the token dependency sharing scenario described above.

```

FEATURE Product
{
  VENDOR=ABC_Software COUNT=10 VERSION=1.0 SHARE=HOST
  KEY=mBpIAWB9Uuz12b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}

FEATURE ModuleA
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=3"
  KEY=4i]mYsfN30C6ShBYszCq2WVipTZXQwkfKJTohkzglwNkle...
}

FEATURE ModuleB
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=2"
  KEY=2w66Ng3wVSPp6ttmWCc8GyJqW300arlnWmnT0lnZXSOIYdF...
}

```

## Generating a token-based license

To generate a token-based license, you must set the keytype to TOKEN, (see the [KEYTYPE](#) description in FEATURE settings). You must also use the TOKEN\_DEPENDENCY directive, which replaces the COUNT directive. See [FEATURE settings](#) for more details on using the TOKEN\_DEPENDENCY directive.

Examples of token-based licensing are shown in the preceding section, Uses for [token-based licensing](#).

## Obtaining information on token-based licenses

When using [LMX\\_GetFeatureInfo\(\)](#) to retrieve information on a token-based license, you will get information for the license you requested, not for the entire chain of licenses defined in the token dependency. You can use [LMX\\_GetFeatureInfo\(\)](#) to determine if the license is a token-based license, and then query all features using [LMX\\_GetLicenseInfo\(\)](#) to see the entire dependency list.

## Exceptions

- Token-based licenses cannot use [borrowing](#) or incremental checkin (see [LMX\\_Checkin](#)).
- Aside from specifying a dependency list instead of a count, a token-based license is identical to a normal network license. Therefore, token-based licenses can be locked with a HostID. However, to provide ease of use when users want to move licenses, we recommend that you do not lock token-based licenses using a HostID.
- [Token-based](#) licenses cannot be checked out using [dynamic reservations](#), because token-based licenses are always additive.



# Retrieving information on a token-based license using the example embedded in the LM-X SDK

The embedded example is located in the LM-X SDK under LMX\_SDK\_INSTALLATION\_PATH/examples/token. You can run an example of how to retrieve information on a token-based license to list all available information about any token dependencies the feature has.

To run the embedded example and start the license server, lmx-serv, with a token.lic license, run the following command.

```
lmx-serv -l LMX_SDK_INSTALLATION_PATH/examples/token/token.lic
```

The above command will start a license server and read a token.lic file that will be served by this server. (Note that the license server, lmx-serv, must be located in the path so that the copied/pasted command can work.)

## Usage

The default location of the token is as follows:

- *Under Windows:* LMX\_SDK\_INSTALLATION\_PATH/examples/token/token.exe
- *Under Unix:* LMX\_SDK\_INSTALLATION\_PATH/examples/token/token

The following command will run the token application and query a license server to obtain all available details of the "myproduct" feature:

```
token myproduct
```

Note that the token program, token, must be located in the path so that the copied/pasted command can work.

- 5 licenses of "prod\_a" in version 1.0
- 5 licenses of "prod\_b" in version 1.0

# Queuing token-based licenses

*The information on this page refers to v5.0 and later, which introduced the ability to queue token-based licenses.*

Queuing token-based licenses is similar to queuing regular (exclusive) licenses, with the following additional rules:

- Each token dependency must be able to be queued in accordance with any set [limitation](#), [reservation](#) or [license total count](#).
- When you queue a token-based license, only the requested token-based license is queued. All of its dependencies, including token-based dependencies, remain unaffected; however, you can queue the dependencies separately.
- When multiple [alternate licenses](#) are queued, each is queued separately. When one is checked out, all others are removed from the queue.
- Fast queuing works the same for token-based licenses as it does for exclusive licenses.

Successful checkout of a token-based license does not remove its dependencies from the queue when dependencies are queued by a separate request (as in example 1 below), unless the checkout results in the pending queued license requests exceeding the [total license count](#) (see example 2, below.) In this case, the dependencies are removed from the license queue to prevent blocking the dependency license queue, because the current client cannot check out more licenses of the dependency feature without returning some of the token-based licenses currently in use.

## Examples

For the following examples, we will use a multilevel token configuration. There will be two token-based licenses (*Token\_lvl1*, *Token\_lvl2*) and one dependency (*Product*). We will also have 3 LM-X clients: Alice, Bob and Charlie.

The license file for this configuration is shown below.

```
FEATURE Token_lvl1
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Token_lvl2 VERSION=1.0 COUNT=2"
  KEY=4ijmYsfn30C6ShBYszCq2WVvcpTZQXwkfKJTohkzg1wNkle...
}

FEATURE Token_lvl2
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=3"
  KEY=l2b2B3vjvcsFBx7qEQG1SI2b2B3vjvcsFBx7qEQG1S12e...
}

FEATURE Product
{
  VENDOR=ABC_Software COUNT=30 VERSION=1.0
  KEY=mBpIAWB9Uuzl2b2B3vjvcsFBx7qEQG1SwXCz8A9d6U3vSKT...
}
```

According to the way that token-based licenses work, when you check out 1 license of feature *Token\_lvl1*, you will also get 2 licenses of feature *Token\_lvl2* and 6 licenses of feature *Product*. Each license of feature *Token\_lvl1* takes 2 licenses of feature *Token\_lvl2*, and each license of feature *Token\_lvl2* takes 3 licenses of the feature *Product*.

### Example 1

The following example demonstrates that the order in the queue is important, and all token dependencies must have enough licenses for checkout requests.

1. Alice checks out 5 licenses of *Token\_lvl1*. (5 *Token\_lvl1*; 10 *Token\_lvl2*; 30 *Product*)
2. Bob wants 3 licenses of *Token\_lvl1*, but there are no licenses available. He goes to the *Token\_lvl1* queue.
3. Charlie wants 2 licenses of *Token\_lvl1*. He also goes to the *Token\_lvl1* queue.
4. Alice returns 2 licenses of *Token\_lvl1*. (2 *Token\_lvl1*; 4 *Token\_lvl2*; 12 *Product*)
5. Bob tries to check out 3 licenses of *Token\_lvl1* (3 *Token\_lvl1*; 6 *Token\_lvl2*; 18 *Product*), but there are only 12 *Product* licenses available. He remains in the *Token\_lvl1* queue.
6. Charlie tries to check out 2 licenses of *Token\_lvl1*, but he is not the first in the *Token\_lvl1* queue.\*
7. Alice returns 1 more license of *Token\_lvl1*. (1 *Token\_lvl1*; 2 *Token\_lvl2*; 6 *Product*)
8. Bob wants 3 licenses, and he finally gets them, because there are enough *Product* licenses available. (3 *Token\_lvl1*; 6 *Token\_lvl2*; 18 *Product*)
9. Charlie wants 2 licenses of *Token\_lvl2*, but all licenses are already taken by Alice and Bob. He remains in the *Token\_lvl1* queue.
10. Bob returns his 3 licenses of feature *Token\_lvl1*.
11. Charlie wants 2 licenses of feature *Token\_lvl1*, and he is now able to obtain them. (2 *Token\_lvl1*; 4 *Token\_lvl2*; 12 *Product*)

\* Note that with fast queuing (see example 3) enabled for the *Token\_lvl1* feature, Charlie would be able to check out the requested licenses.

### Example 2

The following example demonstrates that there is a separate queue for each feature, and the ability to check out any token or token dependency feature.

1. Alice wants 5 licenses of *Token\_lvl2* and she gets them. (5 *Token\_lvl2*; 15 *Product*)
2. Bob wants 2 licenses of *Token\_lvl1* and he gets them. (2 *Token\_lvl1*; 4 *Token\_lvl2*; 12 *Product*)
3. Charlie wants 1 license of *Product* and he gets it. (1 *Product*)
4. There are 2 licenses of *Token\_lvl1*, 9 licenses of *Token\_lvl2* and 28 licenses of *Product* in use.

5. Bob wants another 2 licenses of *Token\_Ivl1*, but not enough licenses of Product are available. He goes to the *Token\_Ivl1* queue.
6. Charlie wants 10 more licenses of Product, but only 2 licenses are available. He goes to the Product queue.
7. Alice returns her 5 licenses of *Token\_Ivl2*. There are 17 licenses of Product available.
8. Charlie requests 10 more licenses of Product before Bob does, and he is able to obtain them, because he is in a different queue than Bob.
9. Bob wants 2 licenses of *Token\_Ivl1*, but there are not enough licenses of Product available. He remains in the *Token\_Ivl1* queue.

### Example 3

The following is an example of fast queuing.

1. Fast queuing is enabled for *Token\_Ivl1*.
2. Alice takes 5 licenses of *Token\_Ivl1*.
3. Bob wants 5 licenses of *Token\_Ivl1*. There are not enough licenses available. He goes to the *Token\_Ivl1* queue.
4. Charlie wants 3 licenses. He also goes to the queue.
5. Alice returns 4 of her 5 *Token\_Ivl1* licenses.
6. Bob still wants 5 licenses of *Token\_Ivl1*. But There are not enough licenses available.
7. Charlie wants only 3 licenses of *Token\_Ivl1*, but he is not first in the queue. Fortunately, fast queueing is enabled for *Token\_Ivl1*. Charlie gets 3 licenses.

### Exception

Queuing token based-licenses generally does not affect its dependency queues. The following example illustrates such a standard case, where successful checkout does *not* remove the dependency from the queue.

1. Alice takes 3 licenses of *Token\_Ivl1*.
2. Bob takes 2 licenses of *Token\_Ivl1*.
3. Charlie wants 2 licenses of *Token\_Ivl1*, but there are no licenses available. Charlie goes to the *Token\_Ivl1* queue.
4. Charlie wants 12 licenses of Product, but there are no licenses available. Charlie goes to the Product queue.
5. Bob returns his 2 licenses of *Token\_Ivl1*.
6. Charlie takes 2 licenses of *Token\_Ivl1* and exits the *Token\_Ivl1* queue.
7. Charlie wants 12 licenses of Product, but there are no licenses available. He remains in the Product queue.
8. Charlie continues to remain in the Product queue, because when Alice returns her *Token\_Ivl1* licenses, Charlie will be able to get the requested 12 Product licenses.

The client is removed from the token dependency queue only when the total number of token dependency licenses minus the number of licenses in use by the client is less than the client's requested number of licenses in the token dependency queue. The following example illustrates such a case, where successful token checkout removes the dependency from the queue.

1. Alice takes 4 licenses of *Token\_Ivl1*.
2. Bob takes 6 licenses of Product.
3. Charlie wants 4 licenses of *Token\_Ivl1*, but there are no licenses available. Charlie goes to the *Token\_Ivl1* queue.
4. Charlie also wants 24 licenses of Product. There are no licenses available, so he goes to the Product queue.
5. Charlie is now in two queues: *Token\_Ivl1* queue and Product queue.
6. Alice wants 6 licenses of Product, but no licenses are available. Alice goes to the Product queue.
7. Alice returns 4 licenses of *Token\_Ivl1*.
8. Charlie takes 4 licenses of *Token\_Ivl1*. This removes him from the *Token\_Ivl1* queue.
9. Charlie is still in the Product queue, but now it's impossible for him to take additional Product licenses, because he already has 24 licenses through *Token\_Ivl1*, and he requested 24 more. This is more than Product has (30); therefore, Charlie is also removed from the Product queue.

## Pay Per Use

The Pay Per Use feature lets you bill customers for using your software on a per-usage basis. For example, you can supply a customer with an unrestricted usage license that is paid for based on monthly usage. To determine the usage, you ask the customer to write the usage information to a pay-per-use SQLite database each month, and send the database file to you for billing purposes. (For more information about SQLite, see <http://www.sqlite.org/>.)

You can enable pay per use by editing the `USAGE_DATABASE` entry in the license server configuration file (`lmx-serv.cfg`). The `USAGE_DATABASE` setting is disabled in the configuration file by default. In addition, the user can edit the `USAGE_LEVEL` setting to specify whether to generate a standard report that excludes user information (the default), or a detailed report that includes the username, hostname and ipaddress for each checkout and checkin request. In addition, the `USAGE_WRITE_INTERVAL` option in the configuration file can be used to specify the number of pay-per-use actions (checkouts, checkins, etc.) that should occur before pay-per-use records will be written to the usage database file. (Note that records will immediately be written to the database file when the server is shut down.) The configuration file contains descriptions and examples for using these settings.

The database is authenticated. You can check for any tampering with the data by running the following command using the `lmxendutil` utility:

```
lmxendutil -readusagedb usage.db
```

Although you can check that the database contains valid data, accurate billing for pay per use is entirely dependent on the customer writing their usage data to the database. LM-X does not enforce the use of the pay-per-use database (the customer could alter the license server configuration file), so you should use the pay per use feature only with customers who you trust to reliably report their usage information.

For more information about the Pay Per Use feature, see the Pay Per Use chapter in the [LM-X End Users Guide](#).

# Dynamic license reservations

Dynamic license reservations let you optimize license usage by reserving required licenses in advance. In this way, you can ensure there are no license usage denials, no gaps in license usage, and no under-utilization of licenses.



Dynamic license reservations work only with virtual and terminal server sharing; they *do not* work with user, host or custom sharing.

Unlike a normal checkout/checkin, a reservation performed by one client can be consumed by any other client, including a different physical machine. You can use reservations in the following ways:

- You can do both a checkout and a reservation for the same feature only when using different LM-X handles for each action.
- You can do multiple reservations for one feature using a different token for each reservation.
- You can use one token to reserve multiple individual features.
- You can make multiple reservations for an individual feature using one token. Each additional call of `LMX_Admin_Reserve` will increase the amount of reserved licenses by the given count. Reservation expire time will be overwritten by the last call of `LMX_Admin_Reserve`.
- Reservations do *not* work with [token-based licenses](#). Attempting to reserve a token-based license will return `LMX_NOT_IMPLEMENTED`. If licenses with the same name are a mix of token-based and normal licenses, LM-X can differentiate between them to use a valid license for a reservation.

If you consume (checkout) a reservation and then later do a return (checkin), the licenses will be returned to the reservation pool, not to the general license pool. You can checkout and checkin a reservation multiple times using multiple clients, and the reservation pools will be maintained. The reservation remains valid until it expires or is explicitly returned using `LMX_Admin_ReserveEarlyReturn`.

For example, a job scheduler handling batch jobs could reserve a license using reservation token 9876 for a specific length of time, say 10 minutes. Within that reservation time, the batch job application can then consume (checkout) the reservation by providing reservation token 9876 to the license server. This ensures that the application requiring the checkout will be able to get a license when needed.

A standard workflow for implementing dynamic license reservations is as follows:

1. A job scheduler uses `LMX_Admin_Reserve` to do the reservation on behalf of another application.
2. With a successful reservation, the scheduler passes on the reservation token to the client application when the application is ready.
3. Using the `LMX_SetOption` flag `LMX_OPT_RESERVATION_TOKEN` together with `LMX_Checkout`, the client application consumes the license.

The following table shows an example progression of how licenses are reserved and consumed. This example is for a single feature with a pool of 10 licenses.

Action	Licenses in Pool	Reserved Licenses	Licenses Consumed by Client 1	Licenses Consumed by Client 2
Default state, before reservations/consumptions	10	0	0	0
Reserve 8 licenses of feature using token ABC	2	8 using token ABC	0	0
Reserve 2 licenses of feature using token XYZ	0	8 using token ABC; 2 using token XYZ	0	0
Any client attempts normal checkout of 1 license (returns <code>LMX_NOT_ENOUGH_LICENSES</code> )	0	8 using token ABC; 2 using token XYZ	0	0
Client 1 consumes 1 reservation using token ABC	0	7 using token ABC; 2 using token XYZ	1 using token ABC	0
Client 2 consumes 1 reservation using token XYZ	0	7 using token ABC; 1 using token XYZ	1 using token ABC	1 using token XYZ
Client 1 consumes 5 reservations using token ABC	0	2 using token ABC; 1 using token XYZ	6 using token ABC	1 using token XYZ
Client 2 attempts normal checkout of 1 license (returns <code>LMX_FEATURE_ALREADY_RESERVED</code> )	0	2 using token ABC; 1 using token XYZ	6 using token ABC	1 using token XYZ
Client 1 checks in all licenses	0	8 using token ABC; 1 using token XYZ	0	1 using token XYZ
Client 2 performs early return using token XYZ	2	8 using token ABC	0	0
Client 2 performs early return using token ABC	10	0	0	0

## Specifying username and hostname for checkouts

Specifying the username and/or hostname for checkouts is particularly useful when all checkouts occur from the same machine, but you want to do user-specific checkouts. To set the username and hostname, you use [LMX\\_OPT\\_CUSTOM\\_USERNAME](#) and [LMX\\_OPT\\_CUSTOM\\_HOSTNAME](#), respectively, as described in [LMX\\_SetOption](#).

A primary use for specifying username and hostname is when you are using a web server or other types of application servers. For example, by default, if the application is located on a web server and the client is a web browser, every license will be consumed by the same user (the web server), instead of a license being consumed for each individual client. This causes a problem when you want to make reservations or perform other actions that rely on the username or hostname.

You can override this behavior by specifying a custom username and hostname that is unique for all sessions, just like a normal user account. This will result in a license being consumed by a unique user every time a client logs in from a browser.

The [LMX\\_OPT\\_CUSTOM\\_USERNAME](#) and [LMX\\_OPT\\_CUSTOM\\_HOSTNAME](#) options can be set only once. If you try to set them more than once, you will get the error `LMX_INVALID_PARAMETER`. If you need to use multiple usernames/hostnames inside one application, you must use separate LM-X clients.

# Upgrade licenses

*The information on this page refers to v5.0 and later, which introduced the upgrade license type.*

Upgrade licenses let you add licenses for software features that a customer buys after the initial purchase of a license, without changing the original license. Upgrade licenses increase the license count of [exclusive licenses](#) by a particular number of licenses, while simultaneously being completely transparent to the client. This approach allows for more flexibility in handling license add-ons than [additive licenses](#) do, because additive licenses are seen as a separate license pool of the same feature.

The license server automatically activates or deactivates upgrade licenses at midnight, and may be limited using the [START](#) and [END](#) directives in the same way as for other license types. You can also create a perpetual (non-expiring) upgrade license, as long as the exclusive license it is based on is perpetual. Also note that existing sessions remain unaffected when an upgrade license expires.

You can upgrade a particular exclusive license as many times as necessary, as long as [COUNT](#) is within its range. It will sum up all of the license upgrades, allowing for a different expiration date for each upgrade.

An upgrade license can be locked to the license server [HostIDs](#) to prevent using the same upgrade license on more than one machine. Duplicated instances of an upgrade license within one license server will be discarded.

Upgrade licenses have the following limitations:

- The upgrade license's expiration dates must be within the exclusive license's expiration dates.
- Neither the upgrade nor exclusive license can be [unlimited](#).
- Client [HostIDs](#) are not supported for upgrade licenses, and [LMX\\_GetLicenseInfo](#) will not report upgrade licenses in network or local paths. These limitations exist because the upgrade licenses are designed to be transparent to the client. However, exclusive licenses report their upgraded license count in both [LMX\\_GetFeatureInfo](#) and [LMX\\_GetLicenseInfo](#).

## License template

To create an upgrade license, create a feature with the same [feature name](#) as the original license, set the number of licenses using the [COUNT](#) directive, and set [KEYTYPE](#)="UPGRADE" in the XML license template file.

In addition to the [feature name](#), [COUNT](#) and [KEYTYPE](#) settings that are required by the [license generator](#) to create an upgrade license, the optional settings that may be used with upgrade licenses are [START](#), [END](#), [SHARE](#), [HOSTID\\_MATCH\\_RATE](#), [HAL\\_SERVERS](#) and [server HostIDs](#).

To maintain maximum compatibility with the corresponding exclusive license, the settings allowed in the upgrade license template are limited. All of the settings behave the same as for exclusive licenses, except that [SHARE](#) is limited to [VIRTUAL](#) only.

## Example

The following license template would result in an upgrade license that increases the feature "f1" license count by one, and is valid from the license server startup to 2030-01-01.

```
<LICENSEFILE>
  <FEATURE NAME="f1">
    <SETTING COUNT="1" />
    <SETTING KEYTYPE="UPGRADE" />
    <SETTING END="2030-01-01" />
  </FEATURE>
</LICENSEFILE>
```

# Building

The following sections describe build requirements, including instructions for compiling your protected application and the LM-X distribution.



# The LM-X protected application and license generator

*This page refers to LM-X v4.5 and newer, which made changes to the SDK distribution. If you are using a version older than v4.5, see [documentation for previous versions](#).*

You generate licenses using xmllicgen, as described in [Generating licenses](#). For your protected application to compile, you need to include the lmx.h header file.

When linking your application, make sure to use the correct set of libraries for your compilation environment.

For Windows only: A single 32-bit or 64-bit LM-X Windows SDK provides a compatible set of LM-X libraries for each supported version of Visual Studio. During the compilation process, the correct libraries will be selected automatically based on the compiler you use.

If you need to use multiple versions of Visual Studio, or you need to use a different Windows compiler (such as an open source compiler), you can build a dll version of LM-X that lets you use LM-X with both supported and unsupported compilers. This alternative offers the best possible flexibility, because the dll will be independent of the compiler used for building your product.

For full details, see the dll example provided with the distribution.

On Windows, multiple versions of the libraries are provided, with suffixes \_mt, \_mtd, \_md or \_mdd according to the type:

Suffix	Description	Restrictions
_mt	Multithreaded static C/C++ runtime	
_mtd	Multithreaded static C/C++ debug runtime	<b>You are not permitted to ship your software using this library.</b>
_md	Multithreaded dynamic C/C++ runtime	
_mdd	Multithreaded dynamic C/C++ debug runtime	<b>You are not permitted to ship your software using this library.</b>

Please consult your compiler documentation for further information.

The protected application should be linked with the liblmxclient library (liblmxclient[suffix].lib on Windows or liblmxclient.a on Unix) libraries. See the example makefiles for additional system libraries that might be required for successful linking.

# Generating licenses

*As of March 1, 2014, xmllicgen uses internet clock check; therefore, a connection to the internet is required to be able to successfully generate a license.*

You generate licenses using xmllicgen, the license generator provided with LM-X.

To generate a license:

1. Modify the sample file detailed.xml, floating.xml or nodelocked.xml (as detailed in the following sections) to fit your specific business model.
2. Execute the license generator by typing the following command:

```
xmllicgen [options] xml_file
```

You can specify the following option to use the specified file to set and override the output file setting in the XML file:

```
-output file
```

As an alternative to specifying the output file path at the command line, you can specify this information in the XML file (using the optional attribute OUTPUTFILE, as described in [LICENSEFILE tag](#)).

If xmllicgen succeeds in creating a license, it returns the error code 0. If xmllicgen fails, it returns the error code 1.

## XML file structure

A license file consists of a LICENSEFILE tag that encloses one or more FEATURE tags. Each FEATURE tag consists of a number of SETTING tags that specify the individual attributes for the feature. A FEATURE tag can also contain one or more CLIENT\_HOSTID or SERVER\_HOSTID tags, which lock the license to either a client machine or license server host, respectively (see [Locking](#) for detailed information about locking).

The valid structure for a license file is shown below.

```
<LICENSEFILE ...>
  <FEATURE NAME="value">
    <SETTING FIELD="value" />
    <CLIENT_HOSTID>
      <SETTING FIELD="value" />
      <SETTING FIELD2="value2" />
    </CLIENT_HOSTID>
    <SERVER_HOSTID>
      <SETTING FIELD="value" />
    </SERVER_HOSTID>
    <SERVER_HOSTID>
      ...
    </SERVER_HOSTID>
  </FEATURE>
  <FEATURE>
    ...
  </FEATURE>
</LICENSEFILE>
```

# Locking

You can lock a single feature to multiple machines (license servers and/or clients) by using multiple <CLIENT\_HOSTID> and <SERVER\_HOSTID> tags. For each machine, you can lock the feature to multiple items (Ethernet HostID, custom HostID, etc.). For example, you can make a local license that works on multiple specified client machines or a floating license that works on multiple servers.

For example, to create a node-locked local license that will work on two known systems, you can lock against multiple, different items on each host in the following format (this example locks against Ethernet and hostname HostIDs for system 1, and username and custom HostIDs on system 2):

```
<CLIENT_HOSTID>
  <SETTING ETHERNET="123..." />
  <SETTING HOSTNAME="ALPHA1" />
</CLIENT_HOSTID>
<CLIENT_HOSTID>
  <SETTING CUSTOM="ABCDEF..." />
  <SETTING USERNAME="joe user" />
</CLIENT_HOSTID>
```

If you make the license counted and specify both <CLIENT\_HOSTID> and <SERVER\_HOSTID> tags, the license can be hosted only on a specific license server machine offering licenses to a known client. Effectively, this is double system locking. You may also allow any license server to host the license by excluding the <SERVER\_HOSTID> tag, and allow only certain clients to use the license from the server by specifying multiple <CLIENT\_HOSTID> tags.

You can use the setting SETTING HOSTIDS to specify multiple HostIDs within the CLIENT\_HOSTID or SERVER\_HOSTID tags. For example:

```
<CLIENT_HOSTID>
  <SETTING HOSTIDS="IPADDRESS=192.168.64.121,HOSTNAME=Alpha1,ETHERNET=C8A516AD01AFC9FA" />
</CLIENT_HOSTID>
```

This setting lets you more easily specify a greater number of HostIDs. You can use the output of [LMX\\_HostidSimple](#) as the value for this setting.

# HostID Types

*The information on this page refers to LM-X v4.9 and newer, which added HostIDs for Google Compute Engine and Azure. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The following HostID types can be used with <CLIENT\_HOSTID> and <SERVER\_HOSTID> tags:

HostID Type	Description
ETHERNET	Ethernet card locking
CUSTOM	Custom locking
IPADDRESS	IP address locking
HOSTNAME	Hostname locking
USERNAME	Username locking
DONGLE_HASPHL	HaspHL dongle locking
HARDDISK	Harddisk locking
LONG	Unix system specific locking
BIOS	Bios locking
WIN_PRODUCT_ID	Windows Product ID locking
AWS_INSTANCE_ID	Amazon EC2 Instance ID locking
GCE_INSTANCE_ID	Google Compute Engine ID locking
AZURE_INSTANCE_ID	Microsoft Azure ID locking

(See [Determining which HostID to use](#) for more information about choosing a HostID that will work best for your needs.)

# LICENSEFILE tag

The <LICENSEFILE> tags can contain the following values.

Value	Description
CONFIGFILE= " <i>config_file.lmx</i> "	Specifies the configuration file to be used for license generation. Note that CONFIGFILE can be specified either within the license file or at the command line (see <a href="#">Generating licenses</a> ).
OUTPUTFILE= " <i>license_file.lic</i> "	Specifies the output file to be used for license generation. Note that OUTPUTFILE can be specified either within the license file or at the command line (see <a href="#">Generating licenses</a> ).
COMMENT= " <i>comment_text</i> "	Comments specific to the license file, such as the type of license being provided. You may use this field as many times as needed to put multiple comment lines into the license.

# FEATURE settings

*The information on this page refers to LM-X v5.0 and newer, which added the upgrade license type. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

As a minimum requirement, each feature must have a name. All other feature settings are optional. The syntax for settings is `<SETTING tagname="value">`. All strings in settings are empty by default and version numbers default to 1.0.

The following is a list of valid values that can be used in `<FEATURE>` tags. This section contains only information for setting each feature. For full descriptions of feature settings, see [Feature descriptions](#).

Value	Usage
<b>Required Settings</b>	
<code>FEATURE="feature_name"</code>	<p>Specify the feature name.</p> <p>The specified value is case-insensitive and may be up to 64 characters in length. Feature names must consist of only letters, numbers, underscores and dashes, and cannot contain spaces.</p> <p>Example:</p> <pre>&lt;FEATURE="F1"&gt;   &lt;SETTING MAJOR_VERSION="2" /&gt;   &lt;SETTING MINOR_VERSION="1" /&gt;... &lt;/FEATURE&gt;</pre>
<b>General License Information Settings</b>	
<code>MAJOR_VERSION="version number"</code>	<p>Specify the major version of the feature. If no version number is set, the version number defaults to 1.</p> <p>Example:</p> <pre>&lt;SETTING MAJOR_VERSION="2" /&gt;</pre>
<code>MINOR_VERSION="version number"</code>	<p>Specify the minor version of the feature. If no version number is set, the version number defaults to 0.</p> <p>Example:</p> <pre>&lt;SETTING MINOR_VERSION="5" /&gt;</pre>
<code>LICENSEE="customer name"</code>	<p>Specify the LICENSEE field (customer name) in the license.</p> <p>Example:</p> <pre>&lt;SETTING LICENSEE="ABC Inc." /&gt;</pre>
<code>START="YYYY-MM-DD"</code>	<p>Specify the license activation date in YYYY-MM-DD format.</p> <p>Example:</p> <pre>&lt;SETTING START="2014-01-25" /&gt;</pre>
<code>END="YYYY-MM-DD"</code>	<p>Specify the license expiration date in YYYY-MM-DD format.</p> <p>The specified date may be up to 9999-12-31.</p> <p>Example:</p> <pre>&lt;SETTING END="2015-01-25" /&gt;.</pre>
<code>MAINTENANCE_START="YYYY-MM-DD"</code>	<p>Specify the license maintenance start date in YYYY-MM-DD format.</p> <p>Example:</p> <pre>&lt;SETTING MAINTENANCE_START="2014-01-25" /&gt;</pre>
<code>MAINTENANCE_END="YYYY-MM-DD"</code>	<p>Specify the license maintenance end date in YYYY-MM-DD format.</p> <p>The specified date may be up to 9999-12-31.</p> <p>Example:</p> <pre>&lt;SETTING MAINTENANCE_END="2015-01-25" /&gt;</pre>

ISSUED="YYYY-MM-DD"	<p>Specify the license issue date in YYYY-MM-DD format.</p> <p>The special keyword "TODAY" can be used to set the current date.</p> <p>Example:</p> <pre>&lt;SETTING ISSUED="2014-01-25" /&gt;</pre>
DATA="custom data"	<p>Set the DATA field (custom data) in the license.</p> <p>Optional field for specifying a comment, such as additional licensing options and additional licensing restrictions. This field is not evaluated during the generation of the KEY checksum; therefore, you can edit the DATA field values without needing to generate a new key.</p> <p>Example:</p> <pre>&lt;SETTING DATA="My custom data" /&gt;</pre>
COMMENT="product description"	<p>Specify the COMMENT field (product description) in the license. (<b>Note:</b> This field is the same as the OPTIONS field.)</p> <p>The specified value is validated in the key, so it is protected from tampering.</p> <p>Example:</p> <pre>&lt;SETTING COMMENT="My software product description" /&gt;</pre>
OPTIONS="licensing options"	<p>Specify the OPTIONS field (licensing options) in the license. (<b>Note:</b> This field is the same as the COMMENTS field.)</p> <p>The specified value is validated in the key, so it is protected from tampering.</p> <p>Example:</p> <pre>&lt;SETTING OPTIONS="My option" /&gt;</pre>
REPLACES="feature name or license key"	<p>Specify the REPLACES field in the license.</p> <p>A license can be replaced by specifying the feature name or the license key.</p> <p>Multiple entries are separated by commas.</p> <p>Example:</p> <pre>&lt;SETTING REPLACES="FEATURE=F1" /&gt; or &lt;SETTING REPLACES="KEY=0fNzdLhI5MJGgDyJV... " /&gt;</pre>
KEYCOMMENT="my text"	<p>Set the KEYCOMMENT field (raw data) in the license.</p> <p>The specified value is hidden inside the KEY field in the generated license.</p> <p>Example:</p> <pre>&lt;SETTING KEYCOMMENT="My comment" /&gt;</pre>
PLATFORMS="platform strings"	<p>Specify a single platform or a subset of platforms to which license usage will be restricted.</p> <p>Multiple entries are separated by spaces.</p> <p>See <a href="#">PLATFORMS</a> for more details, including a list of platform keywords.</p> <p>Example:</p> <pre>&lt;SETTING PLATFORMS="Win32_x86 Linux_x64" /&gt;</pre>
TIME_ZONES="GMT offset values"	<p>Specify the time zones in which to limit license usage relative to GMT.</p> <p>Time zones earlier than GMT use a minus sign preceding the number (for example, an entry of "-1" specifies the time zone GMT -01:00). Multiple entries are separated by commas.</p> <p>Example:</p> <pre>&lt;SETTING TIME_ZONES="1,-4,-5,-6,-7,-8" /&gt;</pre>
<b>Settings for Network Licenses</b>	



COUNT= <i>"number of network licenses"</i>	<p>Set the license count to the specified number, making this a network license that requires a license server.</p> <p>The license count can be up to 2147483647 (the 32-bit integer limit). The count can alternatively be set to "UNLIMITED," which creates a network license with no upper limit.</p> <p>Example:</p> <pre>&lt;SETTING COUNT="10" /&gt;</pre>
TOKEN_DEPENDENCY= <i>"FEATURE=product_name VERSION=version_number COUNT=number of network licenses to fulfill token-based license checkout requests"</i>	<p>Define the license as token-based, and specify the real license upon which the token-based license is dependent, including the real product's name, version number, and number of licenses required to fulfill checkout requests for the token-based license.</p> <p>This setting must be used together with KEYTYPE="TOKEN" and is used in place of the COUNT setting. The value for COUNT used within this setting can be up to 2147483647 (the 32-bit integer limit).</p> <p>Example:</p> <pre>&lt;SETTING TOKEN_DEPENDENCY="FEATURE=realproduct VERSION=1.5 COUNT=5" /&gt;</pre>
KEYTYPE="EXCLUSIVE/ADDITIVE/UPGRADE/TOKEN"	<p>Used for network licenses only. Enable the license type for a network license to be:</p> <ul style="list-style-type: none"> <li>• <b>Exclusive</b> - the default network license type; used with COUNT for a regular network license</li> <li>• <b>Additive</b> - used with COUNT to add a license pool alongside an exclusive network license</li> <li>• <b>Upgrade</b> - used with COUNT to add licenses to an exclusive network license</li> <li>• <b>Token-based</b> - used with TOKEN_DEPENDENCY for a token-based license</li> </ul> <p>Example:</p> <pre>&lt;SETTING KEYTYPE="UPGRADE" /&gt;</pre>
SN= <i>"serial number"</i>	<p>Set the SN field (custom serial number) in the license.</p> <p>The special keyword "RANDOM" can be used to specify a random serial number in the format AAAA-BBBB-CCCC-DDDD, output as hexadecimal characters.</p> <p>Example:</p> <pre>&lt;SETTING SN="RANDOM" /&gt;</pre>
SOFTLIMIT= <i>"number of network licenses"</i>	<p>Enable segmentation of the license count. The number of licenses must be less than that specified for the COUNT field.</p> <p>Example:</p> <pre>&lt;SETTING SOFTLIMIT="8" /&gt;</pre>
HAL_SERVERS= "3"	<p>Enable redundant servers, known as high availability licensing (HAL). Using HAL requires the end user to have a set of 3 license servers, so the HAL_SERVERS setting is set to 3.</p> <p>See <a href="#">HAL_SERVERS (redundant servers)</a> for more information.</p>
BORROW= <i>"hours"</i>	<p>Enable the license borrow time for a network license up to the specified number of hours, from 1 to 8760 hours (1 year).</p> <p>Example:</p> <pre>&lt;SETTING BORROW="168" /&gt;</pre>
GRACE= <i>"hours"</i>	<p>Enable the license grace time for a network license, from 1 to 168 hours (7 days).</p> <p>Example:</p> <pre>&lt;SETTING GRACE="48" /&gt;</pre>

HOLD=" <i>seconds</i> "	<p>Enable the license hold time (time before licenses are checked in) for a network license, from 1 to 31536000 seconds (1 year).</p> <p>Example:</p> <pre>&lt;SETTING HOLD= "60 " /&gt;</pre>
USERBASED=" <i>number of user-based licenses</i> "	<p>Specify a number of licenses that will be available only to named users. The number of licenses can be up to 2147483647 (the 32-bit integer limit). The number can alternatively be set to "ALL," which reserves all licenses. This setting may be used only with network licenses.</p> <p>Example:</p> <pre>&lt;SETTING USERBASED= "10 " /&gt;</pre>
HOSTBASED=" <i>number of host-based licenses</i> "	<p>Specify a number of licenses that will be available only to named hosts. The number of licenses can be up to 2147483647 (the 32-bit integer limit). The number can alternatively be set to "ALL," which reserves all licenses.</p> <p>This setting may be used only with network licenses.</p> <p>Example:</p> <pre>&lt;SETTING HOSTBASED= "5 " /&gt;</pre>
<b>License Share Settings</b>	
<p><b>Note:</b> For network licenses, HOST, USER, CUSTOM and VIRTUAL sharing can be used together. For local licenses, TERMINALSERVER and VIRTUAL can be used together. When combining multiple share options, separate them with a vertical bar (   ); for example, SHARE = HOST USER CUSTOM.</p>	
SHARE="HOST"	For counted network licenses, enable license sharing amongst multiple processes running on the same host. If you combine this with another share option, sharing is enabled only if the combination matches (for example, HOST+USER).
SHARE="USER"	For counted network licenses, enable license sharing amongst processes running as the same user on multiple hosts. If you combine this with another share option, sharing is enabled only if the combination matches (for example, USER+HOST).
SHARE="CUSTOM"	For counted network licenses, enable license sharing amongst processes running with the same custom sharing criteria. If you combine this with another share option, sharing is enabled only if the combination matches (for example, CUSTOM+USER).
SHARE="TERMINALSERVER"	For local licenses, enable application license sharing for remote terminal server clients. If this flag is not used, terminal server clients cannot check out local licenses.
SHARE="SINGLE"	For local licenses, enable application license sharing to limit checkouts to one instance of a feature across multiple sessions.
SHARE="VIRTUAL"	Enable application usage with virtual machines. If this flag is not used, checkout requests for local licenses are denied and license servers will refuse to load the license.
<b>Additional Security Settings</b>	
SYSTEMCLOCKCHECK="LOCAL/INTERNET/FALSE"	<p>Specify the system clock check performed by the client application and license server. You can set this value to LOCAL to check the clock against the local file system (the default) or to INTERNET to check the clock against a remote Network Time Protocol (NTP) time server. Setting the value to FALSE disables the system clock check.</p> <p>Example:</p> <pre>&lt;SETTING SYSTEMCLOCKCHECK= "LOCAL" /&gt;</pre>

HOSTID_MATCH_RATE=" <i>match rate percentage</i> "	<p>Specify a HostID match rate for the license feature. Possible values are 0 to 100. The default value is 100.</p> <p>See <a href="#">HOSTID_MATCH_RATE</a> for more information about using HostID match rate.</p> <p>Example:</p> <pre>&lt;SETTING HOSTID_MATCH_RATE="50" /&gt;</pre>
--	--

# XML file examples

The following are examples of XML files for floating and node-locked licenses, and a more detailed example including multiple features and additional, optional settings. These examples are included as sample files in the LM-X distribution, so you can modify the file that most closely fits your needs.

## Node-locked license example

The following shows an example of the XML file content for a node-locked license with a single feature (sample file nodelocked.xml). Note that the CONFIGFILE and OUTPUTFILE settings can be specified either within the file or at the command line (see [Generating licenses](#)).

```
<LICENSEFILE CONFIGFILE="myprogram.lmx" OUTPUTFILE="nodelocked.lic">
  <FEATURE NAME="f1">
    <SETTING MAJOR_VERSION="1" />
    <SETTING MINOR_VERSION="5" />
    <!-- Note that settings below optional and can be enabled or disabled e.g. for unlimited or demo licenses --
  >
    <SETTING END="2015-01-01" />
    <CLIENT_HOSTID>
    <SETTING ETHERNET="C8A516AD01AFC9FA" />
    </CLIENT_HOSTID>
  </FEATURE>
</LICENSEFILE>
```

## Floating network license example

The following shows an example of the XML file content for a floating license with a single feature (sample file floating.xml).

```
<LICENSEFILE CONFIGFILE="myprogram.lmx" OUTPUTFILE="floating.lic">
  <FEATURE NAME="f1">
    <SETTING END="2015-01-01" />
    <SETTING COUNT="5" />
    <SERVER_HOSTID>
    <SETTING ETHERNET="C8A516AD01AFC9FA" />
    </SERVER_HOSTID>
  </FEATURE>
</LICENSEFILE>
```

Note that:

- The COUNT setting is included, which identifies this as a floating license.
- The CONFIGFILE and OUTPUTFILE settings can be specified either within the file or at the command line (see [Generating licenses](#)).
- The feature's version number is not specified, so it will default to 1.0.

## Detailed license example

The following shows an example of the XML file content for a floating license with multiple features, plus many of the optional settings such as license sharing, borrowing, etc. (sample file detailed.xml).

```

<LICENSEFILE CONFIGFILE="myprogram.lmx" OUTPUTFILE="detailed.lic">
<!-- You can put comments into the header of the license if needed -->
<LICENSEFILE COMMENT="This license was created by the xml based license generator."/>
<LICENSEFILE COMMENT="You could use this space to explain the type of license you are providing."/>
  <FEATURE NAME= "f2">
    <SETTING START="20013-12-24"/>
    <SETTING END="2015-12-25"/>
    <SETTING MAJOR_VERSION="1"/>
    <SETTING MINOR_VERSION="3"/>
    <SETTING LICENSEE="our_user"/>
    <SETTING COMMENT="Our_product_setting_values"/>
    <SETTING KEYCOMMENT="hidden text retrieved from API"/>
    <SETTING SHARE="TERMINALSERVER"/>
    <CLIENT_HOSTID>
    <SETTING ETHERNET="C8A516AD01AFC9FA"/>
    </CLIENT_HOSTID>
    <CLIENT_HOSTID>
    <SETTING CUSTOM="C8A516AD01AFC9FB"/>
    </CLIENT_HOSTID>
  </FEATURE>
  <FEATURE NAME= "f1">
    <SETTING END="2016-01-01"/>
    <SETTING MAJOR_VERSION="1"/>
    <SETTING MINOR_VERSION="5"/>
    <SETTING LICENSEE="Our_valued_customer"/>
    <SETTING COUNT="10"/>
    <SETTING SHARE="HOST"/>
    <SETTING SHARE="USER"/>
    <SETTING BORROW="100"/>
    <SERVER_HOSTID>
    <SETTING ETHERNET="C8A516AD01AFC9FA"/>
    <SETTING CUSTOM="C8A516AD01AFC9FB"/>
    </SERVER_HOSTID>
  </FEATURE>
</LICENSEFILE>

```

Note that the CONFIGFILE and OUTPUTFILE settings can be specified either within the file or at the command line (see [Generating licenses](#)).

## LM-X maintenance

The following sections describe how to update the LM-X distribution when new releases are made available and describe procedures specific to maintaining LM-X for multiple platforms.

# Maintaining LM-X for multiple platforms

If you are building LM-X for multiple platforms:

1. Extract one of the distributions, and make sure the security configuration file is in place Follow the instructions given in [Getting Started with LM-X License Manager](#) for compiling the LM-X distribution.
2. Copy or move the platform-specific directory (for example, win32\_x86 or solaris\_sparc) to the root directory of LM-X. (Only platform-specific files are stored in the platform-specific directory. All shared files exist in the include directory, which is also located in the LM-X root directory.)
3. From the platform-specific directory, use nmake (for Windows) or make (for Unix) to compile the platform-specific files.
4. After you've compiled the platform-specific files, your LM-X directory structure should look similar to the following:

Directory	Description
/include.mk	File defining which platform to compile examples for.
/config	Security configuration file.
/include	Shared header files across all platforms.
<i>Platform-specific directories, such as:</i> /win32_x86/linux_x86/macosex_universal/solaris_sparc  /freebsd_x86	Platform-specific files.

- If you want to compile the examples for a particular platform, you must change the include.mk, where the variable PLATFORM specifies which platform the examples are compiled for.
- Sharing works only between identical versions of LM-X; for example, v3.1 files are not compatible with v3.2 files. You can verify the same version is being used by checking the version.txt file.

# Upgrading LM-X

This page lists instructions for upgrading LM-X License Manager.

**Important:** As of LM-X v4.4, the structure of the SDK has changed. If you are upgrading from a previous version of LM-X License Manager, note the SDK considerations described in [SDK changes for LM-X v4.4 and newer](#).

To upgrade to a newer version of LM-X:

1. [Download the LM-X SDK distribution](#).
2. Install the LM-X SDK on [Windows](#) or [Unix](#).
3. Reuse your existing [security configuration file](#) and avoid generating a new file by copying the file to the config directory under the newly downloaded distribution directory. Also copy your existing `lmx_server_conf.c` file to the config directory if you made changes to this file.
4. Recompile and test your application with the new version of LM-X.

After upgrading, your existing deployed client applications and existing license files should continue to work with the new version, as described in [Version compatibility](#).



## SDK changes for LM-X v4.4 and newer

As of LM-X v4.4, the structure of the LM-X SDK has changed. Previously, implementing license server extensions such as callbacks required changing *\$platform/lmxserverconf.c*, which was later compiled and linked together with the license server library to create the license server executable.

With the release of LM-X v4.4, the license server executable is static and is the same for all vendors. The vendor code and the vendor security information are now stored inside the vendor library *liblmxvendor.dll* (for Windows) or *liblmxvendor.so* (for Unix).

A precompiled default *liblmxvendor.dll/liblmxvendor.so* file is shipped with the SDK. The *liblmxvendor.dll/liblmxvendor.so* file must be distributed to end users together with the license server executable *lmx-serv.exe/lmx-serv*.

Note the following:

- If you have an existing *vendor.lmx* file, it should be renamed to *security\_config.lmx*.
- The *lmxserverconf.c* file has been moved to *\$platform/vendor* and its internal structure modified. Please review the new file and modify your existing code according to the changes in this file.

## SDK considerations for versions older than 4.2

In LM-X SDK versions older than 4.2, [the LM-X security configuration file](#) was named after your vendorname.lmx.

# Version compatibility

At X-Formation we make every effort to ensure that existing deployed software will continue to work with future license servers, and older license files can be read by newer servers. However, although we will strive for backward compatibility in all future releases of LM-X, we cannot guarantee maintaining compatibility if doing so might compromise the security of protected applications.

Be sure to read the [release notes](#) for all new LM-X releases to learn of any issues that may affect version compatibility.

## Backward compatibility

LM-X is designed to be backwards compatible, so that new license servers can read existing license files and existing license clients.

LM-X does not support future compatibility, where the client is newer than the server. Supporting future compatibility would be analogous to an application like Word 2010 supporting the new Word 2013 document format, which would be highly unexpected. Attempting to provide such future compatibility would result in a product that is very complex to maintain and prone to problems.

The following table shows compatibility between LM-X versions, where version 1 is older than version 2:

License Files/Clients	License Servers	Compatible?
Version 1	Version 1	Yes
Version 1	Version 2	Yes
Version 2	Version 1	No

Note that the license file cannot be newer than the LM-X version; therefore, be sure to generate license files as old as the oldest version of LM-X used to deploy your application.

## Version compatibility considerations when upgrading LM-X

When upgrading LM-X to a newer version, note the following version compatibility considerations:

- Existing deployed client applications and existing license files will continue to work with the new version. However, new license files cannot be used by older applications.
- If you distribute floating licenses, your end users must also upgrade their LM-X License Server, because the old license server cannot support new licenses (that is, we cannot provide "future" compatibility). In most cases, borrowed licenses will also continue to work. However, borrowed licenses rely on using Secure Store, so if Secure Store is changed (which is rare), borrowed licenses will not be backward compatible.
- If you are using any external shared libraries—for example, for .NET, Java, etc.—you must also deploy these new versions with your software.
- You can install multiple versions of LM-X, but **do not mix files between different distributions**, because this can cause problems such as compilation or runtime errors. The only file that can safely be shared between platforms is the security configuration file, security\_config.lmx.

See [Upgrading the license server](#) for more information about upgrading the server at the end user site.

## Client API specification

The client API is available for use from your application. The following sections describe the [basic](#) and [advanced](#) client API functions and the error codes that are returned upon any failure of API functions.

**Note:** For all string lengths, such as LMX\_MAX\_NAME\_LENGTH, LMX\_MAX\_SHORT\_STRING\_LENGTH, etc., you should use +1 (e.g., LMX\_MAX\_NAME\_LENGTH+1) to account for the NULL termination in the string.

## Basic API functions

The following sections describe the most commonly used API functions:

- [LMX\\_Checkin](#)
- [LMX\\_Checkout](#)
- [LMX\\_Free](#)
- [LMX\\_GetErrorMessage](#)
- [LMX\\_GetExpireTime](#)
- [LMX\\_GetFeatureInfo](#)
- [LMX\\_Init](#)

For additional API functions that let you implement more advanced features in your licenses, see [Advanced API functions](#).

## Handling exit status

Most LMX API functions return status information LMX\_STATUS that indicates the success or failure of the requested operation. You should review this information to be able to more easily and accurately identify the problem.

The following code is an example of an error handler that writes errors to stderr. Additionally, it is used in all code snippets provided in subsequent sections to make them more concise. Please make sure this function is defined if you want to compile the code as it stands.

```
LMX_HANDLE h;

void exit_on_error(LMX_STATUS s)
{
    if (s != LMX_SUCCESS)
    {
        if (h != NULL)
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessage(h));
            LMX_Free(h); // Release memory allocated by LM-X
        }
        else
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessageSimple(s));
        }
        fflush(stderr);
        exit(1);
    }
}
```

On failure, when a function returns the status code other than LMX\_SUCCESS, the above example will display the corresponding error and its context, as described in [LMX\\_GetErrorMessage](#). If the handler is null, it will only display the error name (see [LMX\\_GetErrorMessageSimple](#)).

# LMX\_Checkin

The LMX\_Checkin function will return the licenses for a single checked in feature or all checked in features.

## Prototype

```
LMX_STATUS LMX_Checkin
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    int nCount
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szFeatureName

[in] Feature name. Use parameter LMX\_ALL\_FEATURES to check in all features.

### nCount

[in] Number of licenses to check in.

Use parameter LMX\_ALL\_LICENSES to check in all licenses.

## Return values

None.

## Remarks

If all licenses are returned to a license server, the licensing system will also ensure that the network connection is closed.

Using LMX\_ALL\_FEATURES with borrow early return is not supported. Instead, check in the features individually.

## Example

The following example returns a license for feature "f2" to a license server.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkin(h, "f2", 1));
    return 0;
}
```

The example below shows initializing LM-X and an attempt to check out a license, after which it displays information about the license including the time remaining to expiration and it tries to check in the license.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    exit_on_error(LMX_Init(&h));
    printf("LM-X client handle successfully initialized: %p\n",h);

    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));
    printf("Get a license.\n");

    LMX_FEATURE_INFO FI;
    exit_on_error(LMX_GetFeatureInfo(h, "f2", &FI));

    printf("FeatureName : %s\n", FI.szFeatureName);
    printf("VendorName : %s\n", FI.szVendorName);
    int nTimeLeft;
    nTimeLeft = LMX_GetExpireTime(h, "f2");
    if (nTimeLeft == -2)
        printf("This feature does not expire\n");
    else if (nTimeLeft == -1)
        printf("This feature is expired\n");
    else
        printf("Hours left for this feature: %d\n", nTimeLeft);

    exit_on_error(LMX_Checkin(h, "f2", 1));
    printf("License returned.\n");

    LMX_Free(h);
    return 0;
}
```



# LMX\_Checkout

The LMX\_Checkout function will checkout one or more licenses for a specific feature.

## Prototype

```
LMX_STATUS LMX_Checkout
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    int nVerMajor,
    int nVerMinor,
    int nCount
);
```

## Parameters

- LmxHandle**  
[in/out] LM-X handle.
- szFeatureName**  
[in] Feature name.
- nVerMajor**  
[in] Major version number, in the range 0-9999.
- nVerMinor**  
[in] Minor version number, in the range 0-9999.
- nCount**  
[in] Number of licenses to checkout.

This value can be one of the following:

An integer in the range 1-2147483647	Lets you set the count to the specified number.
LMX_LOGICAL_CPU_COUNT	Lets you implement processor-based licensing by setting the count to the number of logical CPUs.
LMX_PHYSICAL_CPU_COUNT	Lets you implement processor-based licensing by setting the count to the number of physical CPUs.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

If [softlimit licensing](#) is enabled and the license count has exceeded the specified softlimit licenses, this function can return the status code LMX\_SOFTLIMIT, which also indicates success.

On failure, this function returns an error code in the format described in [Return codes](#). Note that the error code returned reflects only the last license file tested or license server contacted. (See the execution order in Remarks, below.)

To get a complete error description, use the API function [LMX\\_GetErrorMessage\(\)](#).

## Remarks

See [Search paths](#) in the *LM-X End Users Guide* for more information about how the LMX\_Checkout method finds the license.

For local license files, the count value is ignored and checking will depend only on HostID, version, etc.

By default, the version number requested in the function call can be lower than the one available in the license file.

This function will attempt to retrieve the feature from either a license server or a local license file. If the feature is available on a license server, all other feature requests will go to the same license server. LM-X will establish the license server connection transparently to the client application. The execution order is as follows:

1. Try to checkout feature from [Borrow licenses](#) store.
2. Try to checkout feature from string.
3. Try to checkout feature from local path.
4. If network licenses are enabled:
  - a. Try to checkout feature from license server.
  - b. In case of an error other than LMX\_NOT\_ENOUGH\_LICENSES, try to checkout feature from [Grace licenses](#) store.
5. Try to checkout feature from [Trial licenses](#) store.

It is recommended that all licenses come from the same source (that is, the same local license file or the same license server) or as few sources as possible for simplicity and to make licensing most straightforward for users. However, it is technically possible to take licenses from multiple license files and multiple license servers at the same time. See [License server](#) for more information.

It is also recommended that you do as few checkouts as possible to fulfill requests. Preferably, checkouts should map directly to the number of licenses you sell to your customer. This helps ensure that customers understand the products they have bought and the licenses associated with them.

When [License queuing](#) is enabled, LMX\_QUEUE will be returned (see [Return codes](#)) if a license checkout is not successful, after which you should call LMX\_Checkout again to see if a license is available. The queue request is made simultaneously to all the license servers to which the client connects. When one of the servers satisfies the client request, the queue request is removed from all of the servers.

For information on how to increase license checkout performance, see [Optimizing license checkout speed](#).

### Example

The following example illustrates the process of checking out a license for feature "f2", version 1.0.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));
    return 0;
}
```

# LMX\_Free

The LMX\_Free function frees any allocated memory used by the licensing system and closes any open connection to a license server.

## Prototype

```
void LMX_Free
(
    LMX_HANDLE LmxHandle
);
```

## Parameters

### LmxHandle

[in] LM-X handle.

## Return values

None.

## Remarks

A handle of value NULL is valid, but does not cause cleanup.

Calling any other licensing functions on the handle after this one will have undefined behavior.

## Example

The following example shows the use of the LMX\_Free function, which deallocates all memory that was previously allocated by the licensing system.

It is important to note that you should call this function before you leave the application that initialized a handler.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_FEATURE_INFO FI;
    LMX_STATUS s;

    exit_on_error(LMX_Init(&h));
    if ((s = LMX_GetFeatureInfo(h, "nonExistingFeature", &FI)) != LMX_SUCCESS)
    {
        if (h != NULL)
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessage(h));
        }
        else
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessageSimple(s));
            fflush(stderr);
            return 1;
        }
    }
    LMX_Free(h);
    return 0;
}
```

# LMX\_GetErrorMessage

The LMX\_GetErrorMessage function retrieves a NULL-terminated string for the last LM-X function call that occurred.

## Prototype

```
const char * LMX_GetErrorMessage
(
    LMX_HANDLE LmxHandle
);
```

## Parameters

### LmxHandle

[in] LM-X handle.

## Return values

The return value is a pointer to a NULL-terminated string that contains descriptive text for the error code.

## Remarks

This function returns a detailed message for the last LM-X function call that occurred.

The message includes the feature name, internal error code, and context-specific error whenever relevant. Typically, the context-specific error and internal error code are used only by X-Formation for support purposes.

## Example

```
LM-X Error: (Internal: 33 Feature: f1)
Feature not found
For further information go to http://www.x-formation.com
```

The pointer returned is guaranteed to be valid only until the next function call that uses the LmxHandle parameter. Accordingly, it is recommended not to store the pointer.

**Note:** You cannot call LMX\_GetErrorMessage from the heartbeat callback functions (see [Heartbeats](#) and [LMX\\_SetOption](#)). Instead, you may call [LMX\\_GetErrorMessageSimple](#).

The following is a basic example of using LMX\_GetErrorMessage, which returns a detailed error message for the last LM-X function call that occurred.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_FEATURE_INFO FI;

    exit_on_error(LMX_Init(&h));
    if (LMX_GetFeatureInfo(h, "nonExistingFeature", &FI) != LMX_SUCCESS)
    {
        if (h != NULL)
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessage(h));
            LMX_Free(h);
        }
        else
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessageSimple(s));
            fflush(stderr);
            return 1;
        }
    }
    return 0;
}
```

# LMX\_GetExpireTime

The LMX\_GetExpireTime function returns the number of hours remaining before the feature expires.

## Prototype

```
int LMX_GetExpireTime
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName
);
```

## Parameters

- LmxHandle**  
[in] LM-X handle.
- szFeatureName**  
[in] Feature name.

## Return values

This function returns the number of hours left before the feature expires.

Return value	Description
> 0	Number of hours left before the feature expires.
0	The feature expires within the hour.
-1	The feature is expired or an error occurred (for example, LmxHandle is NULL or the feature does not exist).
-2	The feature has no expiration date.

## Remarks

If a local or network license has an expiration, that is the expiration time returned. If a borrow, grace or trial license is in use, that is the expiration time returned. After the borrow or grace license expires, the expiration time of the local or network license will once again be the information that is returned.

You can use [LMX\\_GetFeatureInfo](#) to get the license expiration date.

## Example

You can use the following code to return the number of hours left before the feature expires.

```
/* Get the expire time for a feature */
int nTimeLeft;
LMX_FEATURE_INFO FI;
/* Try to see which type of feature this is */
LMX_GetFeatureInfo(LmxHandle, "f2", &FI);
nTimeLeft = LMX_GetExpireTime(LmxHandle, "f2");
if (nTimeLeft == -2)
    printf("This feature does not expire\n");
else if (nTimeLeft == -1)
    printf("This feature is expired\n");
else /* feature has not yet expired */
    printf("Hours left for this feature: %d\n", nTimeLeft);
```

# LMX\_GetFeatureInfo

*Note that as of LM-X v4.9.20 and newer, the LMX\_FEATURE\_INFO field "szStartDate" applies to trial licenses in addition to regular licenses.*

The LMX\_GetFeatureInfo function retrieves information for a checked out feature.

## Prototype

```
LMX_STATUS LMX_GetFeatureInfo
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    LMX_FEATURE_INFO *pFI
);
```

## Parameters

**LmxHandle**

[in] LM-X handle.

**szFeatureName**

[in] Feature name.

**pFI**

[out] Pointer to the feature information structure. For more details, see the table of LMX\_FEATURE\_INFO fields in Remarks, below.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

Use this API call to retrieve feature settings, as described in the following table. Note that trial licenses have substantially less information available than other license types, because they are a pre-configured license type and do not have the same flexibility.

You can call LMX\_GetFeatureInfo from the heartbeat callback functions (see [Heartbeats](#) for further information).

LMX\_FEATURE\_INFO contains the following fields.

LMX_FEATURE_INFO field	Description
char szFeatureName [LMX_MAX_NAME_LENGTH]  char szVendorName [LMX_MAX_NAME_LENGTH]	These strings contain the feature name and vendor name. The feature name is case-insensitive.
char szStartDate [LMX_MAX_SHORT_STRING_LENGTH]  char szEndDate [LMX_MAX_SHORT_STRING_LENGTH]	These strings contain the start and expire date in format "YYYY-MM-DD".  If no limitations are set, the strings will be empty.
char szActualExpireTime [LMX_MAX_SHORT_STRING_LENGTH]	This string contains the actual license expire time in format "YYYY-MM-DD HH:MM".  If a borrow, grace or trial license is in use, that is the expiration time returned. Otherwise, the expiration time of the local or network license is returned.
char szMaintenanceStartDate [LMX_MAX_SHORT_STRING_LENGTH]  char szMaintenanceEndDate [LMX_MAX_SHORT_STRING_LENGTH]	These strings contain the start and expire date for optional maintenance in the format "YYYY-MM-DD".  If no limitations are set, the strings will be empty.

char szIssuedDate [LMX_MAX_SHORT_STRING_LENGTH]	This string contains the license issue date in the format "YYYY-MM-DD" if specified in the license.
char szPlatforms [LMX_MAX_SHORT_STRING_LENGTH]	This field contains the platforms the license can be used with.
char szComment [LMX_MAX_FIELD_LENGTH]	This field contains the COMMENT string content of the license.
char szData [LMX_MAX_FIELD_LENGTH]	This field contains the DATA string content of the license.
char szLicensee [LMX_MAX_FIELD_LENGTH]	This field contains the LICENSEE string content of the license.
char szKeyComment [LMX_MAX_FIELD_LENGTH]	This field contains the string content embedded within the KEY field.
char szOptions [LMX_MAX_FIELD_LENGTH]	This field contains the string content embedded within the OPTIONS field.
char szSN [LMX_MAX_FIELD_LENGTH]	This field contains the string content embedded within the SN field.
char szKey [LMX_MAX_FIELD_LENGTH]	This field contains the KEY signature of the license.
int nAvailableLicCount	<p>This integer contains the number of licenses available on the license server:</p> <p>-1 (Unlimited network licenses [same as LMX_UNLIMITED_COUNT])</p> <p>0 (Local licenses)</p> <p>1 - 2147483647 (Normal network licenses [same as LMX_MIN_COUNT to LMX_MAX_COUNT])</p>
int nUsedLicCount	This integer contains the number of licenses taken from the license server for the particular feature. The number is within the range LMX_MIN_COUNT to LMX_MAX_COUNT.
int nMajorVer	This integer contains major and minor version numbers.
int nMinorVer	Note that for trial licenses these numbers will both be zero regardless of the version specified in your checkout.
int nSoftLimit	<p>This integer contains the soft limit for the specific feature.</p> <p>This is only available for network licenses and must be less than the count of available licenses.</p>
int nShareCode	<p>This integer contains the type of sharing in use for the specific feature.</p> <p>This flag can be set to LMX_SHARE_HOST, LMX_SHARE_USER, LMX_SHARE_CUSTOM and/or LMX_SHARE_VIRTUAL for network licenses, and LMX_SHARE_TS or LMX_SHARE_SINGLE for local licenses.</p> <p>If there is no sharing in use, this flag is set to LMX_SHARE_NONE.</p> <p>To test whether a specific type of sharing is enabled, use the following (this example checks whether host sharing is enabled):</p> <p>if (FI.nShareCode &amp; LMX_SHARE_HOST)</p>
LMX_LICENSE_TYPE eLicenseType	<p>This enum contains the license type, which can be one of the following:</p> <ul style="list-style-type: none"> <li>• LMX_TYPE_LOCAL</li> <li>• LMX_TYPE_NETWORK</li> <li>• LMX_TYPE_BORROW</li> <li>• LMX_TYPE_GRACE</li> <li>• LMX_TYPE_TRIAL</li> </ul> <p>Note that trial licenses have substantially less information available than other license types, and these fields will be zero or empty.</p>
LMX_KEYTYPE eKeyType	<p>This integer contains the keytype, which can be one of the following:</p> <ul style="list-style-type: none"> <li>• LMX_KEYTYPE_EXCLUSIVE</li> <li>• LMX_KEYTYPE_ADDITIVE</li> <li>• LMX_KEYTYPE_TOKEN</li> </ul>

LMX_TOKEN_DEPENDENCY *pTokenDependency	<p>This pointer holds information about any token dependencies the feature has. (Applies only to network licenses.)</p> <p>This is always set to NULL when using LMX_GetFeatureInfo. You can use LMX_GetLicenseInfo to obtain token dependencies for token-based features. See <a href="#">LMX_GetLicenseInfo</a> for usage of linked lists.</p>
char szPath [LMX_MAX_SHORT_STRING_LEN GTH]	<p>This field contains the path for the license, which will be one of the following:</p> <ul style="list-style-type: none"> <li>• The hostname for a network license</li> <li>• The path to the license file for a local license</li> <li>• "Embedded" for license checkout from a string</li> <li>• "Securestore" for a borrow, trial or grace license</li> </ul>
int nServerPort	<p>This integer contains the port number used by the license server. It is set only when eLicenseType is LMX_TYPE_NETWORK.</p>
int nClientLicenseHostids  int nServerLicenseHostids  LMX_HOSTID ClientLicenseHostid [LMX_MAX_HOSTIDS]  LMX_HOSTID ServerLicenseHostid [LMX_MAX_HOSTIDS]	<p>These fields contain information about how many and which HostIDs the feature is bound to.</p> <ul style="list-style-type: none"> <li>• If nClientLicenseHostid is 0, the application is not locked to client host.</li> <li>• If nServerLicenseHostid is 0, the application is not locked to license server.</li> </ul>
int nHoldMinutes  int nBorrowHours  int nGraceHours	<p>These integers contain information about the number of hold minutes, borrow hours, and grace hours set for the feature in the license.</p>
int nActualBorrowHours	<p>This integer contains information about the actual number of borrow hours, which may be restricted by an administrator on the license server.</p> <p>If no restrictions are set, then this will be the value specified by the license.</p>
char szUniqueID [LMX_MAX_SHORT_STRING_LEN GTH]	<p>This string can be used to identify a unique issued license. You can use this information to track specific licenses or for blacklisting.</p> <p>A unique license is identified by the feature name and this string, for example: {F2, 12345678901234567890}</p>
int nSystemClockCheck	<p>This integer contains information about whether a system clock check has been performed.</p> <p>For standalone licenses, this applies to the license client.</p> <p>For network licenses, this applies to the license server, but not the license client.</p> <p>Possible values are:</p> <p>1 (Enabled)</p> <p>0 (Disabled)</p>
int nHostidLicenseMatchRate	<p>This integer contains the HostID matching rate as specified by the license.</p> <p>Possible values are 0 - 100.</p>
int nHostidActualMatchRate	<p>This integer contains the actual HostID matching performed at checkout. Using this information, you can see the results of the actual HostID verification.</p> <p>Possible values are 0 - 100.</p>
int nUserBasedCount	<p>This integer contains the user-based reservation count required for the license.</p> <p>Possible values are:</p> <p>-1 The user-based reservation count is applied to all licenses. Use integer nAvailableLicCount.</p> <p>0 Disabled</p> <p>&gt;0 Specific number of licenses that are required to be reserved.</p>



int nHostBasedCount	<p>This integer contains the host-based reservation count required for the license.</p> <p>Possible values are:</p> <p>-1 (The host-based reservation count is applied to all licenses. Use integer nAvailableLicCount.)</p> <p>0 (Disabled.)</p> <p>&gt;0 (Specific number of licenses that are required to be reserved.)</p>
int sTimeZones [LMX_MAX_SHORT_STRING_LENGTH] int nTimeZonesCount	<p>This array of integers contains the allowed time zones relative to GMT.</p> <p>For example, an entry of "-5" specifies the timezone GMT -5. If the array is empty, all time zones are allowed.</p>
int nTrialUses	<p>This integer contains the number of possible uses for trial licenses.</p> <p>Default value is -1, which means that trial uses is not set.</p>
int nBlacklisted	<p>This integer contains information about whether a license has been blacklisted.</p> <p>Possible values are:</p> <p>1 - Enabled</p> <p>0 - Disabled</p>

### Example

You can use the following code to retrieve information for a checked out feature. On success, the function prints out the feature name and vendor name.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main() {
    LMX_FEATURE_INFO FI;

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));
    exit_on_error(LMX_GetFeatureInfo(h, "f2", &FI));
    printf("FeatureName : %s\n", FI.szFeatureName);
    printf("VendorName : %s\n", FI.szVendorName);
    return 0;
}
```

You can use the following code to check the time remaining to the expiration of a feature. Depending on which type of license is in use, the function either prints out information that the feature does not expire, or it returns the number of hours before the feature expires.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_FEATURE_INFO FI;

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));
    exit_on_error(LMX_GetFeatureInfo(h, "f2", &FI));
    int nTimeLeft;
    nTimeLeft = LMX_GetExpireTime(h, "f2");
    if (nTimeLeft == -2)
        printf("This feature does not expire\n");
    else if (nTimeLeft == -1)
        printf("This feature is expired\n");
    else /* feature has not yet expired */
        printf("Hours left for this feature: %d\n", nTimeLeft);
    return 0;
}
```

# LMX\_Init

The LMX\_Init function initializes the protection system.

## Prototype

```
LMX_STATUS LMX_Init  
(  
    LMX_HANDLE *pLmxHandle  
);
```

## Parameters

### pLmxHandle

[out] Pointer to LM-X handle.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function should be the first one called.

It is important to note that you should call LMX\_Init only once, because you should need to create only one LMX\_HANDLE. You should continue to use the single handle throughout the lifetime of your application.

## Example

You can use the following code to declare and initialize LMX-handle. Note that any allocated memory must be freed before the program exits.

```
#include <lmx.h>  
#include <stdio.h>  
  
LMX_HANDLE h;  
  
int main()  
{  
    exit_on_error(LMX_Init(&h));  
    // ...  
    LMX_Free(h)  
    return 0;  
}
```

## Advanced API functions

The following sections describe more advanced API functions that you can use to include optional calls for grace and borrow, client store, and many other capabilities to your licenses. The advanced API functions include the following.

- [LMX\\_GetErrorMessageSimple](#)
- [LMX\\_GetError](#)
- [LMX\\_GetLicenseInfo](#)
- [LMX\\_SetOption](#)
- [LMX\\_Hostid](#)
- [LMX\\_HostidSimple](#)
- [LMX\\_ServerLog](#)
- [LMX\\_ServerFunction](#)
- [LMX\\_ClientStoreSave](#)
- [LMX\\_ClientStoreLoad](#)
- [LMX\\_GetVersion](#)
- [LMX\\_Putenv](#)

# LMX\_ClientStoreLoad

The LMX\_ClientStoreLoad function loads content saved in the client store.

## Prototype

```
LMX_STATUS LMX_ClientStoreLoad
(
    LMX_HANDLE LmxHandle,
    const char *szVirtualFilename,
    char *szString
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szVirtualFilename

[in] Filename under which data is stored in the virtual file system.

### szString

[out] The content to load from the client store.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

The LMX\_ClientStoreLoad function and the [LMX\\_ClientStoreSave](#) function work together to store sensitive license information in an encrypted manner into the client store.

The szString parameter length can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

See [Secure store](#) and [Client store](#) for more information about secure store and client store.

## Example

The following is a basic example of using LMX\_ClientStoreLoad, which loads the content saved in "myFile" and displays it as "m" string.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    char m[LMX_MAX_LONG_STRING_LENGTH];

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_ClientStoreSave(h, "myFile", "DataToBeStored"));

    exit_on_error(LMX_ClientStoreLoad(h, "myFile", m));
    printf("Loaded: %s\n", m);

    return 0;
}
```

# LMX\_ClientStoreSave

The LMX\_ClientStoreSave function saves data to the local client store.

## Prototype

```
LMX_STATUS LMX_ClientStoreSave
(
    LMX_HANDLE LmxHandle,
    const char *szVirtualFilename,
    const char *szString
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szVirtualFilename

[in] Filename under which data should be stored in the virtual filesystem.

### szString

[in] The content to store in the client store.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

The LMX\_ClientStoreSave function and the [LMX\\_ClientStoreLoad](#) function work together to store sensitive license information in an encrypted manner into the client store.

The length of the content should be limited to LMX\_MAX\_LONG\_STRING\_LENGTH to avoid buffer overflows.

When the length is zero, the content is deleted from the client store.

See [Secure store](#) and [Client store](#) for more information about secure store and client store.

## Example

The following is a basic example of using LMX\_ClientStoreSave, which saves the "DataToBeStored" string to "myFile".

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    char m[LMX_MAX_LONG_STRING_LENGTH];

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_ClientStoreSave(h, "myFile", "DataToBeStored"));

    exit_on_error(LMX_ClientStoreLoad(h, "myFile", m));
    printf("Loaded: %s\n", m);

    return 0;
}
```

# LMX\_GetError

The LMX\_GetError function returns an error structure containing the complete error message.

## Prototype

```
const LMX_ERROR_INFO *LMX_GetError
(
    LMX_HANDLE LmxHandle
);
```

## Parameters

### LmxHandle

[in] LM-X handle.

## Return values

The return value is a pointer to a structure containing the error information. See the lmx.h header file for details on this structure.

## Remarks

This function is useful for custom error processing, when you want to display the error message from LM-X in a custom format.

For each error message, you can retrieve a feature name to which the error applies, a context-specific error and an internal error code. Typically, the context-specific error and internal error code are used only by X-Formation for support purposes.

**Note:** [LMX\\_GetErrorMessage](#) prints the same error information, but the output is formatted.

## Example

The following example shows an error caused by checking out a non-existent feature. It also shows information contained in the LMX\_ERROR\_INFO structure and calls an exit() function.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_STATUS s;

    exit_on_error(LMX_Init(&h));
    if ((s = LMX_Checkout(h, "nonExistingFeature", 1, 1, 1)) != LMX_SUCCESS)
    {
        const LMX_ERROR_INFO* ePtr = LMX_GetError(h);
        fprintf(stderr, "Status: %s\n", LMX_GetErrorMessageSimple(s));
        fprintf(stderr, "Line: %d\n", ePtr->nInternal);
        fprintf(stderr, "Error code: %d\n", ePtr->nContext);
        fprintf(stderr, "Description: %s\n", ePtr->szDescription);
        fprintf(stderr, "Feature: %s\n", ePtr->szFeatureName);
        fflush(stderr);
        exit(1);
    }
    return 0;
}
```

# LMX\_GetErrorMessageSimple

The LMX\_GetErrorMessageSimple function retrieves a string for a corresponding status value.

## Prototype

```
const char * LMX_GetErrorMessageSimple
(
    LMX_STATUS LmxStat
);
```

## Parameters

### LmxStat

[in] Error status variable.

## Return values

The return value is a pointer to a NULL-terminated string that contains descriptive text for the error code.

## Example

You can use the following code to print out the name of the corresponding LM-X return code in the format described in [Return codes](#).

```
#include <lmx.h>
#include <stdio.h>

int main()
{
    LMX_STATUS s = LMX_INVALID_PARAMETER;
    fprintf(stderr, "%s\n", LMX_GetErrorMessageSimple(s));
    return 0;
}
```



# LMX\_GetLicenseInfo

*The information on this page refers to LM-X v4.9.1 and newer, which added the szHalPath field to the LMX\_LICENSE\_INFO structure. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The LMX\_GetLicenseInfo function retrieves license information from one or more license servers or from a local path.

## Prototype

```
LMX_STATUS LMX_GetLicenseInfo
(
    LMX_HANDLE LmxHandle,
    LMX_LICENSE_INFO **ppLicenseInfo
);
```

## Parameters

**LmxHandle**  
[in/out] LM-X handle.

**ppLicenseInfo**  
[out] Address of LMX\_LICENSE\_INFO structure pointer.

## Return values

On success, this function returns the status code LMX\_SUCCESS.  
On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function is used to retrieve information about both network and local licenses. This function will query one or more license servers to determine the server's status and other license information, and retrieve license information from local paths (a local .lic file or all .lic files in a specified local directory).

This function uses the existing license path set by users through environment variables LMX\_LICENSE\_PATH, VENDOR\_LICENSE\_PATH and custom license paths to query each license server and/or local machine (see [Controlling license behavior](#) for more information on environment variables). Furthermore, this function takes into account whether automatic server discovery is enabled, which affects queries to license servers.

**Note:** You should call this function only once, because it can take several seconds to complete depending on the number of license servers being queried.

Queries can be used to understand license usage, check whether licenses are available prior to license checkout, and determine whether borrow, grace or trial licenses are currently checked out. The same method is used by both the [LM-X End-User Utility](#) and [LM-X End-user Configuration Tool](#) to show who is currently using the licenses on a particular server.

All information in the output data is organized as linked lists. The structures are carefully described in lmx.h, as follows:

Name	Description
LMX_LICENSE_INFO	Holds information about a particular license server or local license file, including information for borrow, grace and trial licenses.
LMX_FEATURE_INFO	Holds information about a particular feature present on a license server or local license, including information for borrow, grace and trial licenses.
LMX_CLIENT_USER	Holds information about a particular user logged onto a license server. (Applies only to network licenses.)
LMX_CLIENT_LEASE	Holds information about a particular license lease done by a specific user. (Applies only to network licenses.)
LMX_CLIENT_QUEUE	Holds information about a particular license queue request done by a specific user. (Applies only to network licenses.)

To enable cycling through objects in a list, each structure contains a pointer named pNext, which identifies the next record. The last record is identified by pNext being NULL.

LMX\_LICENSE\_INFO contains the following fields.

LMX_LICENSE_INFO field	Description
char szPath[LMX_MAX_LONG_STRING_LENGTH+1]	The path and name of the license file (for local licenses) or license server host name (for network licenses).

LMX_STATUS LmxStat	The status of the license path. If this is not LMX_SUCCESS, then all remaining information in LMX_LICENSE_INFO or related structures may be invalid.
char szHalPath[3][LMX_MAX_LONG_STRING_LENGTH+1]	Paths for the 3 HAL servers, in the form of "port@host", where the first element is the master, the second is the first slave, and the third is the second slave.
int nPort	The port number of the license server.
char szVendorName[LMX_MAX_SHORT_STRING_LENGTH+1]	The vendor name.
char szVersion[LMX_MAX_SHORT_STRING_LENGTH+1]	The version of the license server.
int nServerUptimeSeconds	The uptime of the license server.
LMX_FEATURE_INFO *pFeature	Pointers to the features that are present on the license server.
LMX_CLIENT_USER *pUser	Pointers to the users that are present on the license server.

### Example

You can use the following code to extract information about available licenses, their paths, ports, types and list all their features.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_STATUS s;
    LMX_LICENSE_INFO *pLicenseInfo;
    LMX_FEATURE_INFO *pFI;

    exit_on_error(LMX_Init(&h));
    if ((s = LMX_GetLicenseInfo(h, &pLicenseInfo)) != LMX_SUCCESS)
        exit_on_error(s);
    for (; pLicenseInfo != NULL; pLicenseInfo = pLicenseInfo->pNext)
    {
        printf("LicencePath : %s\n", pLicenseInfo->szPath);
        printf("Port : %d\n", pLicenseInfo->nPort);
        printf("Type : ");
        if (pLicenseInfo->eLicenseType == LMX_TYPE_LOCAL)
            printf("Local\n");
        else if (pLicenseInfo->eLicenseType == LMX_TYPE_NETWORK)
            printf("Network\n");
        printf("Features:\n\n");
        for (pFI = pLicenseInfo->pFeature; pFI != NULL; pFI = pFI->pNext)
        {
            printf("FeatureName : %s\n", pFI->szFeatureName);
            printf("VendorName : %s\n", pFI->szVendorName);
        }
    }
    return 0;
}
```

For a more complete example of using these structures, see the sample files included in the LM-X distribution.

All memory used by these structures is handled and maintained by LM-X. It is assumed to be valid until a new request to LMX\_GetLicenseInfo is made.

This function is currently available for use only with C, C++, and .NET.

# LMX\_GetVersion

The LMX\_GetVersion function retrieves the major and minor version number of the LM-X SDK.

## Prototype

```
int LMX_GetVersion();
```

## Return values

This function returns an integer in the format (major)(minor)(minor). For example, a return value of 121 represents version number 1.21.

## Example

The following code prints the LM-X version that is currently in use, and does not require initializing LM-X handle.

```
#include <lmx.h>
#include <stdio.h>

int main()
{
    printf("Version: %d", LMX_GetVersion());
    return 0;
}
```

# LMX\_GetSystemClockTime

*Note that the LMX\_GetSystemClockTime API call was introduced in LM-X v4.9.20. This API call is not available in previous releases of LM-X.*

The LMX\_GetSystemClockTime function returns the system time and performs a system clock check based on a checked out feature.

## Prototype

```
LMX_STATUS LMX_GetSystemClockTime
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    time_t *pTime
);
```

## Parameters

### LmxHandle

[in] LM-X handle.

### szFeatureName

[in] Feature name.

### pTime

[out] Address of time\_t variable.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function validates the system clock time based on a checked out feature, providing a safe way to determine whether a user has changed the time on their system. The system clock check type depends on the feature, as detailed in [SYSTEMCLOCKCHECK](#).

Note that the internet clock check will allow a difference of  $\pm 24$  hours when checking the system clock time.

## Example

The following code shows an example of using this API call.

```
time_t Time;
LMX_STATUS LmxStat = LMX_GetSystemClockTime(LmxHandle, "feature", &Time);
if (LmxStatus == LMX_BAD_SYSTEMCLOCK)
    printf("System clock has been changed!\n");
```

# LMX\_Heartbeat

For local features, the LMX\_Heartbeat function checks whether HostIDs are valid. Note that devices such as dongles can be removed during runtime.

For network features, in addition to checking HostIDs, the LMX\_Heartbeat function checks whether the network connection to the license server is working.

## Prototype

```
LMX_STATUS LMX_Heartbeat
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

## Return values

On success (when local HostIDs are valid and, for network licenses, when the license server connection is up), this function returns the status code LMX\_SUCCESS. Features that do not require HostID checks (borrow, trial, and grace) always return LMX\_SUCCESS.

On failure (one or more local HostIDs are invalid or the license server is down or otherwise unreachable), this function returns an error code in the format described in [Return codes](#).

If you call this function on a feature that does not exist, it will return LMX\_FEATURE\_NOT\_FOUND.

## Remarks

There is no need to call this function for network licenses if you are using [automatic heartbeats](#). (Note that automatic heartbeats are for network licenses only.)

LMX\_Heartbeat should be called in increments of between 30 seconds and 2 minutes. If you call it more often than this, it will return the last cached result.

If heartbeats fail, the client can attempt to make a transparent connection to another server by going through a list of all known servers.

If the license is lost, LMX\_Heartbeat will return LMX\_HEARTBEAT\_LOST\_LICENSE. In the case of a lost license, the client must perform a checkin and then a checkout to continue using the license.

## Example

After a checkin of the feature "f2", the LMX\_Heartbeat function checks whether "f2" connection to the license server is working; for local licenses, LMX\_Heartbeat checks whether HostIDs are still valid, as shown below.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main() {
    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));
    exit_on_error(LMX_Heartbeat(h, "f2"));
    return 0;
}
```

The following example shows how to set an automatic heartbeat with the heartbeat frequency set for 60 seconds.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main() {
    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_SetOption(h, LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS, (LMX_OPTION) 1));
    exit_on_error(LMX_SetOption(h, LMX_OPT_AUTOMATIC_HEARTBEAT_INTERVAL, (LMX_OPTION) 60));
    return 0;
}
```

# LMX\_Hostid

*The information on this page refers to LM-X v4.9 and newer, which added HostIDs for Google Compute Engine and Azure. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The LMX\_Hostid function retrieves the HostID values from the computer system.

## Prototype

```
LMX_STATUS LMX_Hostid
(
    LMX_HANDLE LmxHandle,
    LMX_HOSTID_TYPE eHostidType,
    LMX_HOSTID *lpHostid,
    int *npHostids
);
```

## Parameters

**LmxHandle**  
[in/out] LM-X handle.

**eHostidType**  
[in] Value that specifies the HostID type to be retrieved.

Possible values are:

HostID Type	Description
LMX_HOSTID_ETHERNET	Network card HostID
LMX_HOSTID_USERNAME	Username HostID
LMX_HOSTID_HOSTNAME	Hostname HostID
LMX_HOSTID_IPADDRESS	IP address HostID
LMX_HOSTID_CUSTOM	Custom HostID
LMX_HOSTID_DONGLE_HASPHL	HaspHL Dongle HostID
LMX_HOSTID_HARDDISK	HostID of physical harddisk
LMX_HOSTID_LONG	System-specific HostID
LMX_HOSTID_BIOS	Bios HostID
LMX_HOSTID_WIN_PRODUCT_ID	Windows product ID
LMX_HOSTID_AWS_INSTANCE_ID	Amazon EC2 Instance ID
LMX_HOSTID_GCE_ID	Google Compute Engine ID
LMX_HOSTID_AZURE_ID	Microsoft Azure ID
LMX_HOSTID_ALL	All HostIDs

**lpHostid**  
[out] Pointer to array of LMX\_HOSTID structures. See lmx.h for a description of LMX\_HOSTID. The array must be of size LMX\_MAX\_HOSTIDS.

**npHostids**  
[out] Pointer to a variable that will hold the number of HostIDs of a specific type retrieved. If no HostIDs are available of the type requested, this variable will be set to zero.

## Return values

On success, this function returns the status code LMX\_SUCCESS.  
On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

To make use of [custom HostIDs](#), you must set a callback function using [LMX\\_SetOption](#) with the flag [LMX\\_OPT\\_CUSTOM\\_HOSTID\\_FUNCTION](#).

## Example

You can use the following code to retrieve the HostIDs that are currently in use and list information contained in LMX\_HOSTID.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    LMX_HOSTID hostID[LMX_MAX_HOSTIDS];
    int nbHosts, i;

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Hostid(h, LMX_HOSTID_ALL, hostID, &nbHosts));
    printf("HostIDs found: %d\n", nbHosts);

    for(i = 0; i < nbHosts; ++i)
    {
        printf("Host Type: %d\n", hostID[i].eHostidType);
        printf("Description: %s\n", hostID[i].szDescription);
        printf("Value: %s\n\n", hostID[i].szValue);
    }

    return 0;
}
```



# LMX\_HostidSimple

*The information on this page refers to LM-X v4.9 and newer, which added HostIDs for Google Compute Engine and Azure. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The LMX\_HostidSimple function retrieves the HostID from the computer system and returns it as a NULL-terminated string.

## Prototype

```
LMX_STATUS LMX_HostidSimple
(
    LMX_HANDLE LmxHandle,
    LMX_HOSTID_TYPE eHostidType,
    char *szHostid
);
```

## Parameters

**LmxHandle**  
[in/out] LM-X handle.

**eHostidType**  
[in] Value that specifies the HostID type to be retrieved.

Possible values are:

Hostid Type	Description
LMX_HOSTID_ETHERNET	Network card HostID
LMX_HOSTID_USERNAME	Username HostID
LMX_HOSTID_HOSTNAME	Hostname HostID
LMX_HOSTID_IPADDRESS	IP address HostID
LMX_HOSTID_CUSTOM	Custom HostID
LMX_HOSTID_DONGLE_HASPHL	HaspHL Dongle HostID
LMX_HOSTID_HARDDISK	Hostid of physical harddisk
LMX_HOSTID_LONG	System-specific HostID
LMX_BIOS_HOSTID	Bios HostID
LMX_HOSTID_WIN_PRODUCT_ID	Windows product ID
LMX_HOSTID_AWS_INSTANCE_ID	Amazon EC2 Instance ID
LMX_HOSTID_GCE_ID	Amazon Google Compute Engine ID
LMX_HOSTID_AZURE_ID	Amazon Azure ID
LMX_HOSTID_ALL	All HostIDs

**szHostid**  
[out] Pointer to a string that will hold the HostID(s). The length of the string can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

## Return values

On success, this function returns the status code LMX\_SUCCESS.  
On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

If there are no HostIDs of the requested type, the string will be empty. If there is more than one of the requested type, the HostIDs are separated by commas.

When using LMX\_HOSTID\_ALL, all HostIDs are returned, with the HostID type prefixing the HostID; for example:

```
"ETHERNET=0123456789012345,ETHERNET=54321009876543210,USERNAME=MyUserName1,USERNAME=MyUserName2,
HOSTNAME=MyHostName,..."
```

For single HostIDs, the HostID is returned with multiple HostIDs of that type separated by commas; for example:

```
ETHERNET=0123456789012345,ETHERNET=54321009876543210,..."
```

To make use of [custom HostIDs](#), you must set a callback function using [LMX\\_SetOption](#) with the flag [LMX\\_OPT\\_CUSTOM\\_HOSTID\\_FUNCTION](#).

### Example

You can use the following code to retrieve all the HostIDs with the LMX\_HOSTID\_IPADDRESS type that are currently in use and display them on success.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    char s[LMX_MAX_LONG_STRING_LENGTH];

    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_HostidSimple(h, LMX_HOSTID_IPADDRESS, s));
    printf("HostIDs found: %s\n", s);

    return 0;
}
```

# LMX\_Putenv

The LMX\_Putenv function lets you change environment variables.

## Prototype

```
LMX_STATUS LMX_Putenv
(
    const char *szEnvironmentVariable
);
```

## Parameters

### szEnvironmentVariable

[in] Environment variable name and value, in the format "*name=value*" (identical to the C API putenv() function).

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function should be used instead of using functions such as putenv or unsetenv.

## Example

```
LMX_Putenv( "LMX_LICENSE_PATH=local_license.lic" );
```

# LMX\_ServerFunction

LMX\_ServerFunction sends a message to the license server for processing for a checked out feature. The license server will invoke the specified custom callback function and return a response.

## Prototype

```
LMX_STATUS LMX_ServerFunction
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    char *szMessage
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szFeatureName

[in] Feature name.

### szMessage

[in/out] The message to send and the response.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function will send a message to the server from which a feature was checked out.

The length of the message sent to the server and the response can be up to LMX\_MAX\_LONG\_STRING\_LENGTH bytes.

In order for this function to succeed, the license server must have a function callback implemented. See `lmx_server_conf.c` for details.

## Example

The following example shows a function that could be registered in `lmx_server_conf.c`:

```
/* The function parameters should be overwritten by the response back to the client. */
void LMX_CALLBACK ServerFunc(char *szMessage)
{
    char *szTest = "this is a simple response";
    LmxLogprintf("server function called: %s", szMessage);

    /* Use the client message in some way ...*/
    ...
    ...
    ...
    /* Prepare response */
    strcpy(szMessage, szTest);
}
```

The client can then invoke the function given above as follows:

```
char szMyMessage[LMX_MAX_LONG_STRING_LENGTH+1];
char *szString = "This is my simple query";
int nLength;
LMX_STATUS LmxStat;
/* Specify a message to send (here we use a zero-terminated string, but any data could be used */
strcpy(szMyMessage, szString);
LmxStat = LMX_ServerFunction(handle, "f1", szMyMessage);
/* Now the query has been replaced with a response */
/* Print out the response since we know the server sent us a string */
printf("%s\n", szMyMessage);
```

# LMX\_ServerLog

The LMX\_ServerLog function will log a message in the license server's log for a checked out feature.

## Prototype

```
LMX_STATUS LMX_ServerLog
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    const char *szMessage
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szFeatureName

[in] Feature name.

### szMessage

[in] The message to send.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function will send a message to the server from which a feature was checked out.

## Example

You can use the following code to send a message to the server from which a feature was checked out. Please note that a feature must be checked out before a message can be logged in the license server's log.

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    exit_on_error(LMX_Init(&h));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1)); // Before sending a message we need to checkout a license
    exit_on_error(LMX_ServerLog(h, "f2", "A message to be logged by the server"));
    return 0;
}
```

# LMX\_SetOption

The LMX\_SetOption function sets flags that change licensing behavior prior to license checkout.

## Prototype

```
LMX_STATUS LMX_SetOption
(
    LMX_HANDLE LmxHandle,
    LMX_SETTINGS eOption,
    const void *pSetting
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### eOption

See the eOption information for the specific option, listed below.

### pSetting

See the Setting information for the specific option, listed below.

**Note:** To avoid warnings when using the pSetting parameter, you can typecast it to the type (LMX\_OPTION).

## LMX\_SetOption functions

The functions for LMX\_SetOption are listed below.

**Note:** Unless otherwise specified, the LMX\_SetOption options affect checkout requests.

- [LMX\\_OPT\\_HEARTBEAT\\_CHECKOUT\\_SUCCESS\\_FUNCTION](#)
- [LMX\\_OPT\\_HEARTBEAT\\_CHECKOUT\\_FAILURE\\_FUNCTION](#)
- [LMX\\_OPT\\_HEARTBEAT\\_RETRY\\_FEATURE\\_FUNCTION](#)
- [LMX\\_OPT\\_HEARTBEAT\\_CONNECTION\\_LOST\\_FUNCTION](#)
- [LMX\\_OPT\\_HEARTBEAT\\_EXIT\\_FUNCTION](#)
- [LMX\\_OPT\\_HEARTBEAT\\_CALLBACK\\_VENDORDATA](#)
- [LMX\\_OPT\\_AUTOMATIC\\_HEARTBEAT\\_ATTEMPTS](#)
- [LMX\\_OPT\\_AUTOMATIC\\_HEARTBEAT\\_INTERVAL](#)
- [LMX\\_OPT\\_ALLOW\\_BORROW](#)
- [LMX\\_OPT\\_ALLOW\\_CHECKOUT\\_LESS\\_LICENSES](#)
- [LMX\\_OPT\\_ALLOW\\_GRACE](#)
- [LMX\\_OPT\\_ALLOW\\_MULTIPLE\\_SERVERS](#)
- [LMX\\_OPT\\_TRIAL\\_DAYS](#)
- [LMX\\_OPT\\_TRIAL\\_USES](#)
- [LMX\\_OPT\\_TRIAL\\_TERMINAL\\_SERVER](#)
- [LMX\\_OPT\\_TRIAL\\_VIRTUAL\\_MACHINE](#)
- [LMX\\_OPT\\_HOSTID\\_ENABLED](#)
- [LMX\\_OPT\\_HOSTID\\_DISABLED](#)
- [LMX\\_OPT\\_HOSTID\\_COMPARE\\_FUNCTION](#)
- [LMX\\_OPT\\_HOSTID\\_CACHE\\_CLEANUP\\_INTERVAL](#)
- [LMX\\_OPT\\_CLIENT\\_HOSTIDS\\_TO\\_SERVER](#)
- [LMX\\_OPT\\_CUSTOM\\_HOSTID\\_FUNCTION](#)
- [LMX\\_OPT\\_CUSTOM\\_HOSTNAME](#)
- [LMX\\_OPT\\_CUSTOM\\_SHARE\\_STRING](#)
- [LMX\\_OPT\\_CUSTOM\\_USERNAME](#)
- [LMX\\_OPT\\_LICENSE\\_PATH](#)
- [LMX\\_OPT\\_LICENSE\\_STRING](#)
- [LMX\\_OPT\\_LICENSE\\_IDLE](#)
- [LMX\\_OPT\\_EXACT\\_VERSION](#)
- [LMX\\_OPT\\_BIND\\_ADDRESS](#)
- [LMX\\_OPT\\_BLACKLIST](#)
- [LMX\\_OPT\\_RESERVATION\\_TOKEN](#)
- [LMX\\_OPT\\_SERVERSIDE\\_REQUEST\\_STRING](#)

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

Note that some of the flags are used in combination with checkout requests. This enables you to set flags for single checkout requests if needed.

### Example

The following example shows the use of LMX\_SetOption by activating and deactivating [LMX\\_OPT\\_EXACT\\_VERSION](#).

```
#include <lmx.h>
#include <stdio.h>

LMX_HANDLE h;

int main()
{
    exit_on_error(LMX_Init(&h));
    /* First checkout request requires an exact version match to pass */
    exit_on_error(LMX_SetOption(h, LMX_OPT_EXACT_VERSION, (LMX_OPTION) 1));
    exit_on_error(LMX_Checkout(h, "f2", 1, 0, 1));

    /* Second checkout request does not require an exact version match to pass
       and can request a version up to the one specified in the .lic file */
    exit_on_error(LMX_SetOption(h, LMX_OPT_EXACT_VERSION, (LMX_OPTION) 0));
    exit_on_error(LMX_Checkout(h, "f2", 0, 5, 1));

    return 0;
}
```



# LMX\_OPT\_HEARTBEAT\_CHECKOUT\_SUCCESS\_FUNCTION

eOption	<div>LMX_OPT_HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION</div> <div>This flag causes a callback function to be called when the heartbeat (per feature) has the status LMX_SUCCESS; that is, a heartbeat succeeds in reclaiming a license after a license server disconnection.</div>
pSetting	<div>Function pointer value.</div> <div>Default value:</div> <div>NULL (disabled)</div> <div>Prototype:</div> <div><pre>void (LMX_CALLBACK *HeartbeatCheckoutSuccess_pfn) (     void *pVendorData, /* Vendor-specified data */     const char *szFeatureName, /* Name of feature successfully reclaimed */     int nUsedLicCount /* Number of licenses reclaimed, which is equal to the number originally checked out */ );</pre></div>

# LMX\_OPT\_HEARTBEAT\_CHECKOUT\_FAILURE\_FUNCTION

eOption	<div>LMX_OPT_HEARTBEAT_CHECKOUT_FAILURE_FUNCTION</div> <div>This flag causes a callback function to be called when the heartbeat (per feature) has a status different than LMX_SUCCESS; that is, if a heartbeat fails to reclaim a license after a license server reconnection.</div> <div>If this function is called, the licenses for the given feature are considered lost.</div>
pSetting	<div>Function pointer value.</div> <div>Default value:</div> <div>NULL (disabled)</div> <div>Prototype:</div> <div><pre>void (LMX_CALLBACK *HeartbeatCheckoutFailure_pfn) (     void *pVendorData, /* Vendor-specified data */     const char *szFeatureName, /* Name of lost feature */     int nUsedLicCount, /* Number of licenses lost */     LMX_STATUS LmxStat /* Error code that indicates why they were lost */ );</pre></div>

# LMX\_OPT\_HEARTBEAT\_RETRY\_FEATURE\_FUNCTION

eOption	<div>LMX_OPT_HEARTBEAT_RETRY_FEATURE_FUNCTION</div> <div>This flag causes a callback function to be called whenever there is an attempt to re-checkout a feature from a working server.</div> <div>After this callback function is called, it is expected that the status for the feature will be given by calling either:</div> <div>LMX_OPT_HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION</div> <div>or</div> <div>LMX_OPT_HEARTBEAT_CHECKOUT_FAILURE_FUNCTION</div>
pSetting	<div>Function pointer value.</div> <div>Default value:</div> <div>NULL (disabled)</div> <div>Prototype:</div> <div><pre>void (LMX_CALLBACK *HeartbeatRetryFeature_pfn) (     void *pVendorData, /* Vendor-specified data */     const char *szFeatureName, /* Name of feature about to be checked out */     int nUsedLicCount /* Number of licenses to checkout */ );</pre></div>

# LMX\_OPT\_HEARTBEAT\_CONNECTION\_LOST\_FUNCTION

e O p ti on	<p>LMX_OPT_HEARTBEAT_CONNECTION_LOST_FUNCTION</p> <p>This flag causes a callback function to be called each time the connection to the license server is lost and heartbeats are failing. When this is called, all features are considered lost temporarily or permanently, depending on the results of the LMX_OPT_HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION and LMX_OPT_HEARTBEAT_CHECKOUT_FAILURE_FUNCTION callbacks.</p> <p>The parameter nFailedHeartbeats starts at 1 and goes up to the number of heartbeat attempts specified by LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS. This parameter is per server.</p>
p S e tti ng	<p>Function pointer value.</p> <p>Default value:</p> <p>NULL (disabled)</p> <p>Prototype:</p> <div><pre>void (LMX_CALLBACK *HeartbeatConnectionLost_pfn) (     void *pVendorData, /* Vendor-specified data */     const char *szHost, /* NULL-terminated string containing license server host */     int nPort, /* TCP port number of license server */     int nFailedHeartbeats /* The number of times a heartbeat has failed per server*/ );</pre></div>

# LMX\_OPT\_HEARTBEAT\_EXIT\_FUNCTION

e O pti on	<p>LMX_OPT_HEARTBEAT_EXIT_FUNCTION</p> <p>This flag causes a callback function to be called when automatic heartbeats cease attempts to reconnect to the license server. When this happens, the heartbeat thread is stopped, and no more heartbeats are called, but the application will continue to run.</p> <p>If this flag is not set and automatic heartbeats cease attempts to reconnect to the license server, the heartbeat thread is stopped, and the client prints the message, "Lost license(s)! Unable to reconnect to license server!" to stderr, and calls exit(1). This causes the application to quit with return code 1 (see <a href="#">Return codes</a>).</p>
p S ett ing	<p>Function pointer value.</p> <p>Default value:</p> <p>NULL (disabled)</p> <p>Prototype:</p> <div><pre>void (LMX_CALLBACK *HeartbeatExit_pfn) (     void *pVendorData, /* Vendor-specified data */ );</pre></div>

## LMX\_OPT\_HEARTBEAT\_CALLBACK\_VENDORDATA

eOption	<p>LMX_OPT_HEARTBEAT_CALLBACK_VENDORDATA</p> <p>This flag lets you set a void pointer that will be supplied to the heartbeat callbacks. This allows you to pass custom information to the heartbeat callbacks.</p>
pSetting	<p>Void pointer value.</p> <p>Default value:</p> <p>NULL</p>

## LMX\_OPT\_AUTOMATIC\_HEARTBEAT\_ATTEMPTS

e O pti on	<p>LMX_OPT_AUTOMATIC_HEARTBEAT_ATTEMPTS</p> <p>This flag specifies whether automatic heartbeats are enabled and the maximum number of failed heartbeat attempts before LM-X shuts down the application.</p> <p>When this option is enabled and network licensing is used, a separate thread is started that performs automatic heartbeats once every 2 minutes. This interval can be changed using <a href="#">LMX_OPT_AUTOMATIC_HEARTBEAT_INTERVAL</a>.</p> <p>Setting the value to NULL will result in setting it to the default value.</p>
pS ett ing	<p>Possible values:</p> <p>An integer, as follows:</p> <p>-1 (Ignore failed heartbeat attempts. With this setting, your application will not shut down even if the license server goes down infinitely after a checkout. This setting is useful for a lenient license model. New instances of your application will not be able to start, but existing sessions will continue to run.)</p> <p>0 (Disabled. The client should perform manual heartbeats instead using <a href="#">LMX_Heartbeat()</a> to keep the license.</p> <p>&gt;0 (Shut down the application if no heartbeat response is received after the specified number of consecutive attempts. Once a valid heartbeat response is received, the internal counter will reset.)</p> <p>Default value:</p> <p>0 (disabled)</p>

# LMX\_OPT\_AUTOMATIC\_HEARTBEAT\_INTERVAL

eOption	<div>LMX_OPT_AUTOMATIC_HEARTBEAT_INTERVAL</div> <div>This flag specifies the time in seconds between each automatic heartbeat performed.</div> <div>This option is used only when automatic heartbeats are enabled.</div> <div>Setting the value to NULL will result in setting it to the default value.</div> <div>Note that changes to this option might require that you adjust the client timeout on the license server accordingly to prevent your application timing out from the server.</div>
pSetting	<div>Possible values:</div> <div>An integer, as follows:</div> <div>30 - 900 (seconds)</div> <div>Default value:</div> <div>120 (seconds)</div>



## LMX\_OPT\_ALLOW\_BORROW

eOption	<p>LMX_OPT_ALLOW_BORROW</p> <p>When this flag is enabled, license borrowing is allowed on the client side.</p> <p>When this flag is disabled, the client will reject any borrowing or borrowed licenses.</p> <p>See <a href="#">BORROW</a> for more information about using borrow licenses.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>1 (enabled)</p>

## LMX\_OPT\_ALLOW\_CHECKOUT\_LESS\_LICENSES

eOption	<p>LMX_OPT_ALLOW_CHECKOUT_LESS_LICENSES</p> <p>When this flag is enabled, the license server that does not have enough licenses to fulfill the request is allowed to return fewer licenses than requested during <a href="#">LMX_Checkout</a>.</p> <p>When this flag is disabled, <a href="#">LMX_Checkout</a> fails with LMX_NOT_ENOUGH_LICENSES when the license server does not have enough licenses to fulfill the request.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled)</p> <p>1 (enabled)</p> <p>Default value:</p> <p>0 (disabled)</p>

## LMX\_OPT\_ALLOW\_GRACE

eOption	<p>LMX_OPT_ALLOW_GRACE</p> <p>When this flag is enabled, grace licenses are allowed on the client side.</p> <p>When this flag is disabled, the client will reject loading any grace licenses, even if they are enabled at the license server side.</p> <p>See <a href="#">GRACE</a> for more information about using grace licenses.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>1 (enabled)</p>

## LMX\_OPT\_ALLOW\_MULTIPLE\_SERVERS

eOption	<p>LMX_OPT_ALLOW_MULTIPLE_SERVERS</p> <p>This flag lets you switch between using one server or using multiple servers simultaneously during checkout. If this flag is enabled, each feature can be checked out from a separate server.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>1 (enabled)</p>

# LMX\_OPT\_TRIAL\_DAYS

eO pti on	<p>LMX_OPT_TRIAL_DAYS</p> <p>This flag specifies the number of days a trial license will be available in the event that no other license is found. You may use this in combination with LMX_OPT_TRIAL_USES (see below) if desired. If both options are used, the trial will expire based on whichever limit occurs first.</p> <p>Setting the value to NULL will result in setting it to the default value.</p> <p>See <a href="#">Trial licenses</a> for more information about using trial licenses.</p>
pS etti ng	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled—trial licenses are not available)</p> <p>1 - 60 (number of days a trial license will be available)</p> <p>Default value:</p> <p>0 (disabled)</p>

## LMX\_OPT\_TRIAL\_USES

eOption	<p>LMX_OPT_TRIAL_USES</p> <p>This flag specifies the number of uses allowed for a trial license. You may use this in combination with <a href="#">LMX_OPT_TRIAL_DAYS</a> if desired. If both options are used, the trial will expire based on whichever limit occurs first.</p> <p>Setting the value to NULL will result in setting it to the default value.</p> <p>See <a href="#">Trial licenses</a> for more information about using trial licenses.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <ul style="list-style-type: none"><li>-1 (disabled)</li><li>1 - unlimited (number of times a trial license may be checked out)</li></ul> <p>Default value:</p> <ul style="list-style-type: none"><li>-1 (disabled)</li></ul>

## LMX\_OPT\_TRIAL\_TERMINAL\_SERVER

eOption	<p>LMX_OPT_TRIAL_TERMINAL_SERVER</p> <p>This flag lets you control whether trial licenses are allowed to work on terminal servers.</p>
pSetting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disallow) 1 (allow)</p> <p>Default value:</p> <p>0 (disallowed)</p>

## LMX\_OPT\_TRIAL\_VIRTUAL\_MACHINE

eOption	LMX_OPT_TRIAL_VIRTUAL_MACHINE  This flag lets you control whether trial licenses are allowed to work on virtual machines.
pSetting	Possible values:  An integer, as follows:  0 (disallow) 1 (allow)  Default value:  0 (disallowed)



# LMX\_OPT\_HOSTID\_ENABLED

e O pti on	<p>LMX_OPT_HOSTID_ENABLED</p> <p>This flag specifies a particular HostID type to use during checkout.</p> <p>When this flag is set, only the specified HostID will be used during checkout. You should ensure that the HostID specified is a HostID type that is used by the client.</p> <p>By default, all HostID types are enabled; therefore, when using this option you must first set the option LMX_OPT_HOSTID_DISABLED using LMX_HOSTID_ALL HostID as the HostID type. This will disable all HostIDs so that only the HostID specified in LMX_OPT_HOSTID_ENABLED will be used.</p>
pS ett ing	<p>Enum value of type LMX_HOSTID_TYPE.</p> <p>Default value:</p> <p>None. By default, all HostID types are enabled.</p>

## LMX\_OPT\_HOSTID\_DISABLED

eOption	<p>LMX_OPT_HOSTID_DISABLED</p> <p>This flag specifies a particular HostID type to remove during checkout.</p> <p>When this flag is set, the specified HostID type is not used during checkout, which can improve performance.</p> <p>Setting this flag to LMX_HOSTID_ALL will disable all HostID types, so that you can set a specific HostID type using LMX_OPT_HOSTID_ENABLED.</p>
pSetting	<p>Enum value of type LMX_HOSTID_TYPE.</p> <p>Default value:</p> <p>None. By default, all HostID types are disabled.</p>

# LMX\_OPT\_HOSTID\_COMPARE\_FUNCTION

eOption	<div>LMX_OPT_HOSTID_COMPARE_FUNCTION</div> <div>This flag causes a callback function to be called to override the existing compare to verify whether a server HostID or client HostID is valid.</div>
pSetting	<div>Function pointer value.</div> <div>Default value:</div> <div>NULL (disabled)</div> <div>Prototype:</div> <div><pre>LMX_STATUS (LMX_CALLBACK *HostidCompareCallBack_pfn) (     LMX_KEY_HOSTID_TYPE eKeyHostidType, /* LMX_CLIENT_HOSTID or LMX_SERVER_HOSTID expected */     const LMX_HOSTID pLicenseHostid[],     int nLicenseHostids,     const LMX_HOSTID pSystemHostid[],     int nSystemHostids );</pre></div> <div>See <a href="#">Custom HostID compare</a> for information on using this callback function.</div>

# LMX\_OPT\_HOSTID\_CACHE\_CLEANUP\_INTERVAL

eO ption	<p>LMX_OPT_HOSTID_CACHE_CLEANUP_INTERVAL</p> <p>This flag lets you change the HostID cache interval (in seconds).</p> <p>By default, HostIDs are cached the first time during checkout requests.</p> <p>By setting this flag, you can avoid devices being used as HostIDs because they were cached. For example, a user could make a checkout using a dongle, then move the dongle to another computer and use it again while the dongle HostID is cached.</p> <p>Use caution if changing this setting. An interval that is too low can decrease performance by creating too many checks. Conversely, an interval that is too high (or disabled) can compromise security. When this flag is disabled, no cache cleanup occurs.</p>
pS etti ng	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) &gt;0 (seconds)</p> <p>Default value:</p> <p>120 (seconds)</p>

# LMX\_OPT\_CLIENT\_HOSTIDS\_TO\_SERVER

e O pt ion	<p>LMX_OPT_CLIENT_HOSTIDS_TO_SERVER</p> <p>When this flag is set, client HostIDs will be sent to the license server.</p> <p>You should set this flag only when locking a network license to the client's HostIDs. Enabling this option can affect performance, so it is recommended to minimize the number of HostIDs that will be sent by specifying LMX_OPT_HOSTID_ENABLED and LMX_OPT_HOSTID_DISABLED as described in <a href="#">Optimizing license checkout speed</a>.</p>
p S et ti ng	<p>Possible Values:</p> <p>An integer, as follows:</p> <p>0 (disabled)</p> <p>1 (enabled)</p> <p>Default value:</p> <p>0 (disabled)</p>

# LMX\_OPT\_CUSTOM\_HOSTID\_FUNCTION

e O pt ion	<p>LMX_OPT_CUSTOM_HOSTID_FUNCTION</p> <p>This flag causes a callback function to be called if using the HostID functions LMX_Hostid() and LMX_HostidSimple() with parameter LMX_HOSTID_CUSTOM. This callback function is also called during checkouts when the license includes one or more custom HostIDs, regardless of whether a local license or a license server is being used.</p>
p S et ti ng	<p>Function pointer value.</p> <p>Default value:</p> <p>NULL (disabled)</p> <p>Prototype:</p> <div><pre>LMX_STATUS (LMX_CALLBACK *CustomHostidCallBack_pfn) (     LMX_HOSTID *pHostid,     int *npHostids );</pre></div> <p>See <a href="#">LMX_Hostid</a> for information on using this callback function.</p>

## LMX\_OPT\_CUSTOM\_HOSTNAME

eOption	<p>LMX_OPT_CUSTOM_HOSTNAME</p> <p>This flag sets a hostname that overrides the hostname sent to the license server during checkout.</p> <p>For more information, see <a href="#">Specifying username and hostname for checkouts</a>.</p>
pSetting	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (disabled)</p>

# LMX\_OPT\_CUSTOM\_SHARE\_STRING

e O pt ion	<p>LMX_OPT_CUSTOM_SHARE_STRING</p> <p>This flag enables a custom sharing string that is used when custom sharing is enabled in the network license. Processes that use the same share string will share the same license unless the HOST and/or USER options are set and the processes have different hosts and/or users. Processes using different share strings will not share licenses.</p> <p>Setting the value to NULL will result in setting it to the default value.</p>
p S et ti ng	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (Custom sharing not enabled)</p>



## LMX\_OPT\_CUSTOM\_USERNAME

eOption	LMX_OPT_CUSTOM_USERNAME  This flag sets a username that overrides the username sent to the license server during checkout.  For more information, see <a href="#">Specifying username and hostname for checkouts</a> .
pSetting	NULL-terminated string.  Default value:  NULL (disabled)

### Remarks

The [LMX\\_OPT\\_CUSTOM\\_USERNAME](#) option can be set only once. If you try to set it more than once, you will get the error LMX\_INVALID\_PARAMETER.

# LMX\_OPT\_LICENSE\_PATH

e O pt ion	<p>LMX_OPT_LICENSE_PATH</p> <p>This flag sets the predefined license path, which is the default path the client application searches through to find a license. See <a href="#">How a protected application finds its license</a> in the <i>LM-X End Users Guide</i> for complete information about the license format, as well as the format of the environment variables users can set to specify a license file path.</p>
p S et ti ng	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (disabled)</p>

# LMX\_OPT\_LICENSE\_STRING

e O p ti on	<p>LMX_OPT_LICENSE_STRING</p> <p>This flag sets a predefined license in a string. Passing a license through a string enables you to eliminate the requirement of external license files in your shipped application. A checkout attempt will be done on this string prior to any other license paths being used. If you call this flag multiple times using different license strings, the previous license string will be overwritten, not appended to.</p> <p>Setting the value to NULL will result in setting it to the default value.</p>
p S e tti ng	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (disabled)</p>

## LMX\_OPT\_LICENSE\_IDLE

eOp tion	<p>LMX_OPT_LICENSE_IDLE</p> <p>This flag specifies whether license idling is enabled. When enabled, no heartbeats are sent to the license server, resulting in the license server releasing the licenses after 5 minutes (configurable as described in License server).</p>
pSe tting	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>0 (disabled)</p>

## LMX\_OPT\_EXACT\_VERSION

eO ption	<p>LMX_OPT_EXACT_VERSION</p> <p>When this flag is enabled, the feature version specified in the license must exactly match the requested version in the checkout call.</p> <p>When this flag is disabled, features can be checked out if their version is equal to or less than the version specified in the license.</p> <p>With this flag disabled, you can provide licenses that will be "future proof" up to a certain feature version.</p> <p>For example, say you are currently shipping version 1.3 of your software, but will soon ship version 1.5, and want to offer your customers an update promotion that lets them buy the current version now and receive version 1.5 for free when it ships.</p> <p>To do this, you would set the feature version to 1.5 in the provided license, which will then work with both the current version and with version 1.5 once it is released, but will not work with any versions later than 1.5.</p> <p>If this flag is enabled, however, the license will work only with the specified feature version.</p>
pS etti ng	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>0 (disabled)</p>

# LMX\_OPT\_BIND\_ADDRESS

eOption	<div>LMX_OPT_BIND_ADDRESS</div> <div>This flag lets you limit which networks the client can connect through. When this flag is set, the client can use only the specified IP address to connect to a network.</div> <div>You can set either an IPV4 address (for example, 192.168.0.2) or IPV6 address (for example, 1:2:3:4::1) that represents the local IP address you want to communicate through.</div> <div>When this flag is not set, the client can connect to any available network.</div> <div>Setting the value to NULL will result in setting it to the default value.</div>
pSetting	<div>NULL-terminated string.</div> <div>Default value:</div> <div>NULL (disabled)</div>

# LMX\_OPT\_BLACKLIST

e O p ti on	<p>LMX_OPT_BLACKLIST</p> <p>This flag lets you enable blacklisting of local, network or borrow licenses.</p> <p>Blacklisting applies to a specific feature in the license file, based on the unique KEY value of the feature. If this flag is set for a feature that is checked out and checked back in, the feature will no longer be available for checkout. For network licenses, calls to <a href="#">LMX_Checkout</a> for the blacklisted license will return LMX_FEATURE_NOT_FOUND; for local licenses, calls to <a href="#">LMX_Checkout</a> will return LMX_BLACKLIST.</p> <p>You can re-host the license (replace the blacklisted license file with a newly generated license file) to make the feature available for checkout again.</p> <p>For more information about using blacklisting, see <a href="#">Blacklisting issued licenses</a>.</p>
p S e tti ng	<p>Possible values:</p> <p>An integer, as follows:</p> <p>0 (disabled) 1 (enabled)</p> <p>Default value:</p> <p>0 (disabled)</p> <p>For example:</p> <pre>// Check out the feature to be blacklisted if (LMX_Checkout(hLMX, "feature_to_blacklist", 1, 0, 1) == LMX_SUCCESS) {     // Blacklist the currently checked out license     LMX_SetOption(hLMX, LMX_OPT_BLACKLIST, (LMX_OPTION)1);     // When the license is checked back in, it will no longer be available for future checkouts     LMX_Checkin("feature_to_blacklist", 1);     // Return to normal checkin/checkout functionality     LMX_SetOption(hLMX, LMX_OPT_BLACKLIST, (LMX_OPTION)0); }</pre>

## LMX\_OPT\_RESERVATION\_TOKEN

eOption	<p>LMX_OPT_RESERVATION_TOKEN</p> <p>This flag sets a reservation token that can be used together with <a href="#">LMX_Checkout</a> to consume a reservation that was made for a particular feature and token.</p> <p>Setting the value to NULL will result in setting it to the default value.</p>
pSetting	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (disabled)</p>



## LMX\_OPT\_SERVERSIDE\_REQUEST\_STRING

eO pti on	<p>LMX_OPT_SERVERSIDE_REQUEST_STRING</p> <p>This flag sets a custom string that is sent to the server as a part of the checkout (see <a href="#">LMX_Checkout</a>) or checkin (see <a href="#">LMX_Checkin</a>) request. The set string will be available from the LMX_SERVERSIDE_REQUEST structure as a callback on the license server.</p> <p>Setting the value to NULL will result in setting it to the default value.</p>
pS etti ng	<p>NULL-terminated string.</p> <p>Default value:</p> <p>NULL (disabled)</p>

# Administrative API functions

The following sections describe API functions that you can use to perform administrative functions on the license server. The administrative API functions include the following:

- [LMX\\_Admin\\_Reserve](#)
- [LMX\\_Admin\\_ReserveEarlyReturn](#)
- [LMX\\_Admin\\_UploadLicense](#)
- [LMX\\_Admin\\_RestartServer](#)
- [LMX\\_Admin\\_ShutdownServer](#)

# LMX\_Admin\_Reserve

The LMX\_Admin\_Reserve function will use a token to reserve one or more licenses for a specified time.

## Prototype

```
LMX_STATUS LMX_Admin_Reserve
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    int nVerMajor,
    int nVerMinor,
    int nCount,
    int nReservationTime,
    const char *szReservationToken
);
```

## Parameters

- LmxHandle**  
[in/out] LM-X handle.
- szFeatureName**  
[in] Feature name.
- nVerMajor**  
[in] Major version number, in the range 0-9999.
- nVerMinor**  
[in] Minor version number, in the range 0-9999.
- nCount**  
[in] Number of licenses to reserve for future checkouts.

This value can be one of the following:

An integer in the range 1-2147483647	Lets you set the count to the specified number.
LMX_LOGICAL_CPU_COUNT	Lets you implement processor-based licensing by setting the count to the number of logical CPUs.
LMX_PHYSICAL_CPU_COUNT	Lets you implement processor-based licensing by setting the count to the number of physical CPUs.

- nReservationTime**  
[in] Length of time (in seconds) to reserve the license(s). After this time expires, the reserved license(s) will be returned to the pool.
- szReservationToken**  
[in] Reservation token (a unique string that you specify) used to acquire and consume reservations.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#). Note that the error code returned reflects only the last license file tested or license server contacted.

To get a complete error description, use the API function [LMX\\_GetErrorMessage\(\)](#).

See [LMX\\_Checkout](#) for additional information about return values, which also apply to this function.

## Remarks

The reservation token is a unique string you specify, and can have a length up to LMX\_MAX\_SHORT\_STRING\_LENGTH. The token is used to acquire the reservation, which later can be consumed (checked out) using the same token.

See [LMX\\_Checkout](#) for additional remarks, which also apply to this function.

# LMX\_Admin\_ReserveEarlyReturn

The LMX\_Admin\_ReserveEarlyReturn function will return all licenses reserved for the features using the given token.

## Prototype

```
LMX_STATUS LMX_Admin_ReserveEarlyReturn
(
    LMX_HANDLE LmxHandle,
    const char *szFeatureName,
    const char *szReservationToken
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szFeatureName

[in] Feature name. Use parameter LMX\_ALL\_FEATURES to return all reserved features.

### szReservationToken

[in] Reservation token used to acquire and consume reservations.

## Return values

On failure, this function returns an error code in the format described in [Return codes](#). Note that the error code returned reflects only the last license file tested or license server contacted.

## Remarks

None.

# LMX\_Admin\_UploadLicense

The LMX\_Admin\_UploadLicense function uploads a license file to the server.

## Prototype

```
LMX_STATUS LMX_Admin_UploadLicense
(
    LMX_HANDLE LmxHandle,
    const char *szLicenseFilename,
    const char *szLicenseContent
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szLicenseFilename

[in] The name under which to save the license file on the server side, with an lic extension; for example, filename.lic. If this parameter is empty, then the name is automatically generated on the server side. The maximum length of the filename can be up to LMX\_MAX\_NAME\_LENGTH.

### szLicenseContent

[in] License file content. The maximum length of the license file content can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function sends license file content to the license server that is specified in [LMX\\_OPT\\_LICENSE\\_PATH](#) and saves the license file to the default location on that license server. This license file will then be used automatically upon the next license server restart. Only paths set through [LMX\\_SetOption](#) and LMX\_OPT\_LICENSE\_PATH will be used; all other paths will be ignored. This ensures full control of which server the license will be uploaded to.

If the license file already exists, the new license file will overwrite the existing license file. This also makes it possible to remove a license file of the same name by replacing it with an empty file.

(Note: License file upload can also be done using the LM-X web-based UI as described in [Managing licenses](#).)

# LMX\_Admin\_RestartServer

*The information on this page refers to LM-X v4.9.16 or newer, which added the LMX\_Admin\_RestartServer function. This setting is not available in previous versions of LM-X.*

The LMX\_Admin\_RestartServer function restarts the specified server.

## Prototype

```
LMX_STATUS LMX_Admin_RestartServer
(
    LMX_HANDLE LmxHandle,
    const char *szHostname,
    int nPort,
    const char *szPassword
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szHostname

[in] Hostname of the license server to be restarted. The maximum length of the hostname can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

### nPort

[in] Port of the license server to be restarted. Port must be within the range of 1 - 65535.

### szPassword

[in] Password of the license server to be restarted. The maximum length of the password can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function sends a restart command to the license server specified in szHostname. If the password specified in szPassword matches the password in the license server's configuration file, the server will be restarted, and LMX\_SUCCESS will be returned. If the passwords do not match, the server will not be restarted, and LMX\_LMX\_NETWORK\_DENY will be returned.

(Note: License server restart can also be done using the LM-X web-based UI, as described in [Restarting or shutting down the server](#).)

# LMX\_Admin\_ShutdownServer

*The information on this page refers to LM-X v4.9.16 or newer, which added the LMX\_Admin\_ShutdownServer function. This setting is not available in previous versions of LM-X.*

The LMX\_Admin\_ShutdownServer function shuts down the specified server.

## Prototype

```
LMX_STATUS LMX_Admin_ShutdownServer
(
    LMX_HANDLE LmxHandle,
    const char *szHostname,
    int nPort,
    const char *szPassword
);
```

## Parameters

### LmxHandle

[in/out] LM-X handle.

### szHostname

[in] Hostname of the license server to be shut down. The maximum length of the hostname can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

### nPort

[in] Port of the license server to be shut down. Port must be within the range of 1 - 65535.

### szPassword

[in] Password of the license server to be shut down. The maximum length of the password can be up to LMX\_MAX\_LONG\_STRING\_LENGTH.

## Return values

On success, this function returns the status code LMX\_SUCCESS.

On failure, this function returns an error code in the format described in [Return codes](#).

## Remarks

This function sends a shut down command to the license server specified in szHostname. If the password specified in szPassword matches the password in the license server's configuration file, the server will be shut down, and LMX\_SUCCESS will be returned. If the passwords do not match, the server will not be shut down, and LMX\_LMX\_NETWORK\_DENY will be returned.

(Note: License server shut down can also be done using the LM-X web-based UI, as described in [Restarting or shutting down the server](#).)

# Java and .NET client APIs

The Java and .NET APIs for LM-X are available to use from your Java/.NET application. LM-X's Java and .NET APIs deliver functionality equivalent to LM-X's C/C++ API and are consistent with all existing data structures.

J a v a A P I	<p>The complete Java API code is enclosed in the com.xformation.lmx java package, which you can easily import into your project as a JAR archive library (liblmxjava.jar file).</p> <p>Detailed Java API documentation is included with LM-X in a javadoc format that can be self-contained or integrated with IDE. You can view this documentation by opening the file index.html.</p> <p>Example code that demonstrates how to use the Java API is included in the LM-X examples subdirectory.</p>
. N E T A P I	<p>The supported languages for the .NET API include C++/CLI, C# and VB. The complete .NET API code is enclosed in the library liblmxnet.dll, which you can easily import into your project. The custom LicenseProvider (LmxLicenseProvider) and License (LmxLicense) classes have been implemented to make it easy to license controls and components using .NET methods with LM-X.</p> <p>Detailed .NET API documentation is included with LM-X in a doxygen format, which you can see by opening the file index.html.</p> <p>Example code that demonstrates how to use the .NET API is included in the LM-X examples subdirectory.</p>



## Return codes

*The information on this page refers to LM-X v5.1.2 or newer. LM-X v5.1.2 added the new error code 71. This code is not included in previous versions of LM-X.*

The following table lists the possible error codes that are returned upon any failure of API functions:

Return Code #	Return Code	Description
0	LMX_SUCCESS	Operation successful.
1	LMX_UNKNOWN_ERROR	Unknown error occurred.
2	LMX_INVALID_PARAMETER	Invalid input parameter.
3	LMX_NO_NETWORK	Unable to initialize network subsystem.
4	LMX_BAD_LICFILE	License file is using unknown/invalid syntax.
5	LMX_NO_MEMORY	No more available memory.
6	LMX_FILE_READ_ERROR	Unable to read file.
7	LMX_BAD_DATE	Invalid date.
8	LMX_BAD_KEY	Invalid license key.
9	LMX_FEATURE_NOT_FOUND	Feature not found.
10	LMX_BAD_HOSTID	HostID does not match license.
11	LMX_TOO_EARLY_DATE	Software activation date is not yet reached.
12	LMX_TOO_LATE_DATE	Software expired.
13	LMX_BAD_VERSION	Software version does not match license.
14	LMX_NETWORK_ERROR	Unexpected network-related error occurred.
15	LMX_NO_NETWORK_HOST	Unable to connect to license server.
16	LMX_NETWORK_DENY	Rejected actively from license server.
17	LMX_NOT_ENOUGH_LICENSES	Request for more licenses than available on license server.
18	LMX_BAD_SYSTEMCLOCK	System clock has been set back.
19	LMX_TS_DENY	Feature not allowed to run on <a href="#">terminal server clients</a> .
20	LMX_VIRTUAL_DENY	Feature not allowed to run on a virtual machine.
21	LMX_BORROW_TOO_LONG	The specified borrow period is too long.
22	LMX_FILE_SAVE_ERROR	Unable to save file.
23	LMX_ALREADY_BORROWED	Feature already borrowed.
24	LMX_BORROW_RETURN_ERROR	Unable to return borrowed feature.
25	LMX_SERVER_BORROW_ERROR	Deprecated. License server returned borrow error.
26	LMX_BORROW_NOT_ENABLED	Borrow functionality not enabled on client side.
27	LMX_NOT_BORROWED	The feature that was attempted to be returned was not borrowed.
28	LMX_DONGLE_ERROR	Dongle is not attached or does not function correctly.
29	LMX_SOFTLIMIT	Request exceeds the number of softlimit licenses available.
30	LMX_BAD_PLATFORM	Platform not permitted by license.
31	LMX_RESET_SYSTEMCLOCK_EXCEEDED	Deprecated. Number of allowed reset system clock attempts exceeded.
32	LMX_TOKEN_LOOP	Infinite token loop detected.
33	LMX_BLACKLIST	Feature is blacklisted.

34	LMX_VENDOR_DENY	Feature checkout rejected by vendor-defined rules.
35	LMX_NOT_NETWORK_FEATURE	Unable to use local license as a network license.
36	LMX_BAD_TIMEZONE	Checkout is not permitted in the client time zone.
37	LMX_SERVER_NOT_IN_USE	License server is not currently in use.
38	LMX_LICSERVICE_ERROR	Deprecated. Problem with License Distribution Service.
40	LMX_NOT_IMPLEMENTED	Functionality not implemented.
41	LMX_BORROW_LIMIT_EXCEEDED	License server limitation on number of borrowed features exceeded.
42	LMX_SERVER_FUNC_ERROR	License server function error occurred.
43	LMX_HEARTBEAT_LOST_LICENSE	License is lost due to heartbeat failure.
44	LMX_SINGLE_LOCK	Unable to obtain single-usage lock.
45	LMX_AUTH_ERROR	Cannot authenticate user on license server.
46	LMX_NETWORK_SEND_ERROR	Error sending message over network.
47	LMX_NETWORK_RECEIVE_ERROR	Error receiving message over network.
48	LMX_QUEUE	Feature has been queued.
49	LMX_BAD_SECURITY_CONFIG	LM-X security configuration file mismatch.
50	LMX_FEATURE_HAL_MISMATCH	Feature has different HAL settings than other features on the same license server.
51	LMX_NOT_LOCAL_FEATURE	Unable to use network license as a local license.
52	LMX_FEATURE_NOT_REPLACEABLE	Unable to replace missing feature.
53	LMX_HOSTID_NOT_AVAILABLE	HostID is not available on the current machine.
54	LMX_FEATURE_ALREADY_RESERVED	Feature is already reserved.
55	LMX_FEATURE_ALREADY_CHECKED_OUT	Feature is already checked out.
56	LMX_RESERVATION_NOT_FOUND	Reservation not found.
57	LMX_API_NOT_REENTRANT	Calling an API function from a callback function is not allowed.
58	LMX_LICENSE_UPLOAD_ERROR	Problem with license file upload.
59	LMX_INTERNAL_LICENSE_NOT_EMBEDDED	Internal LM-X license file is not embedded.
60	LMX_SYSTEM_INTERPROCESS	Interprocess resource locking error.
61	LMX_CANNOT_LOAD_SHARED_LIBRARY	Cannot load LM-X library. (We recommend that you check the permissions for the C:\Users\USER\NAME\AppData\Local\Temp folder.)
62	LMX_SERVER_VERSION_TOO_LOW	License server version is lower than the client.
63	LMX_VENDOR_NAME_MISMATCH	License vendor names do not match.
64	LMX_SECURITY_CONFIG_NOT_EMBEDDED	LM-X security configuration file is not embedded.
65	LMX_FEATURE_DUPLICATED	Duplicated feature found during LMX_GetLicenseInfo call.
66	LMX_NOT_INITIALIZED	LM-X is not initialized.
67	LMX_USER_SESSION_NOT_FOUND	No user session is found for the user being removed from the license server.
68	LMX_MINIMUM_REMOVAL_TIME_NOT_PASSED	Minimum time for removing the user from the license server has not yet passed.
69	LMX_USER_REMOVAL_DISABLED	User removal from license server is disabled.
70	LMX_TOKEN_DOUBLE_SHARING	Tokens and their dependencies cannot both be shared at the same time.

71	LMX_LICENSE_IS_TOO_NEW	The license key version is newer than the client can understand.
----	------------------------	--

## Developer solutions

The following sections offer solutions to help you with your particular license management needs. You can also find additional training topics in the [LM-X Training](#) section of the [X-Formation Knowledgebase](#).

- [Building LM-X for multiple platforms](#)
- [Building LM-X SDK for both 32-bit and 64-bit .NET framework](#)
- [Choosing between a floating and node-locked license](#)
- [Deciding whether to lock a license to one or multiple HostIDs](#)
- [Determining which HostID to use](#)
- [Determining which IP address to lock your license to](#)
- [Detecting a real license](#)
- [Implementing licensing for critical-use applications](#)
- [Managing license expiration for licenses running non-stop](#)
- [Using CPU cores](#)
- [Providing demo versions of your software](#)
- [Restricted access licensing](#)
- [Upgrading LM-X](#)
- [Using CPU cores](#)
- [Using dates for version numbers](#)
- [Rehosting or blacklisting a license](#)

## Building LM-X for multiple platforms

To build LM-X for multiple platforms:

1. Install one of the distributions, and make sure the security configuration file is in place.
2. Copy or move the platform-specific directory (for example, win32\_x86 or solaris\_sparc) to the root directory of LM-X. (Only platform-specific files are stored in the platform-specific directory. All shared files exist in the include directory, which is also located in the LM-X root directory.)
3. From the platform-specific directory, compile the platform-specific files.
  - *On Windows*, type: nmake (for example, C:\lm-x\_distrib\win32\_x86> nmake). Alternatively, you can use Visual Studio solutions lmx\_vs2010.sln, or lmx\_vs2012.sln, lmx\_vs2013.sln.
  - *On Unix*, type: make (using GNU make).

(Note that most Unix platforms use the command "make"; however, some platforms may use "gmake.")

## Building LM-X SDK for both 32-bit and 64-bit .NET framework

If you want to deploy your application to be able to use both 32-bit and 64-bit .NET, you need to build liblmxnet.dll with the 32-bit and 64-bit .NET compiler. Note that both copies of this file should be shipped with your application.

In order to successfully create two versions of liblmxnet.dll, you must use the same security key pair when creating them.

The following steps will let you know how to proceed:

1. Compile the LM-X SDK as usual for x64:

```
C:\Program Files\X-Formation\LM-X SDK v4.8.4 win64_x64>(configure your environment for 64-bit compilation)
C:\Program Files\X-Formation\LM-X SDK v4.8.4 win64_x64>nmake
```

2. Copy the generated C:\Program Files\X-Formation\LM-X SDK v4.8.4 win64\_x64\win64\_x64\x-formation\_pair.snk to the equivalent directory of the 32-bit SDK:

```
C:\Program Files (x86)\X-Formation\LM-X SDK v4.8.4 win32_x86\win32_x86\x-formation_pair.snk
```

3. Proceed with compiling the LM-X SDK for x86:

```
C:\Program Files (x86)\X-Formation\LM-X SDK v4.8.4 win32_x86>(configure your environment for 32-bit compilation)
C:\Program Files (x86)\X-Formation\LM-X SDK v4.8.4 win32_x86>nmake
```

## Choosing between a floating and node-locked license

A [floating license](#) allows an application to be used on a network, whereas a [node-locked license](#) allows a single instance of an application to run on a specific machine. Determining which type of license to use depends on your end user's needs and the method of software distribution that will work best for you.

The following table lists advantages and disadvantages of using each type of license.

Type	Advantages	Disadvantages
Floating	<ul style="list-style-type: none"><li>• Your end user can share any number of software licenses between multiple workstations.</li><li>• When your end user closes the application, the license is released back to the central pool so it can be checked out by another user.</li><li>• You have control over the number of licensed copies allowed to run concurrently.</li></ul>	<ul style="list-style-type: none"><li>• Your end user can experience workstation crashes and temporary network outages. Solutions: <a href="#">High Availability Licensing (HAL)</a> and grace licenses.</li><li>• Your end user cannot work on laptop away from the network. Solutions: <a href="#">Borrowing a license</a> and VPN connections.</li><li>• This license type does not work when environment or operating system is blocking socket or network activity (such as with a Linux web server restricted user account). Solution: Use a node-locked licensing instead.</li></ul>
Node-locked	<ul style="list-style-type: none"><li>• This type of license does not need the connection to the license server.</li></ul>	<ul style="list-style-type: none"><li>• An unlimited number of copies to run on a specified host.</li><li>• You need multiple licenses if a customer is using the software on more than one computer.</li></ul>

# Deciding whether to lock a license to one or multiple HostIDs

Locking your licensed application to an Ethernet card, BIOS or harddisk HostID provides enough security to meet most needs. If you require more security, you may lock to multiple HostIDs using one of the following methods.

Method	Usage
Multiple HostIDs with 1:1 matching	With this method, all HostIDs must match exactly for the license to work. For example, if you lock the license to the license server machine hostname, Ethernet adapter, and BIOS HostIDs, all three HostIDs must match for the license to work.
Multiple HostIDs with advanced comparison	With this method, you use the HostID compare function, <a href="#">LMX_OPT_HOSTID_COMPARE_FUNCTION</a> , to specify multiple HostIDs in the license file and use them under custom conditions. For example, you might create a condition to lock to the Ethernet card if it is available, and if it is not available, lock to the harddisk. Another example is to include all available information about the machine in the license file, and then allow the software to run if the harddisk and Ethernet HostIDs match, and ignore all other information.
Multiple HostIDs using HostID match rating	With this method, you use the HostID match rating setting, <a href="#">HOSTID_MATCH_RATE</a> , to specify the percentage required for a successful match, from 0 (no matching required) to 100 (exact matching required). For example, SETTING HOSTID_MATCH_RATE="50" indicates that a 50% or better match must be achieved in order for the license to work. A strict 1:1 match provides optimal security, while a less strict match provides more flexibility, allowing users to run the software if, for example, only two out of three HostID values are valid.

For information about supported HostIDs and which is best for your needs, see [Determining which HostID to use](#).



# Determining which HostID to use

*The information on this page refers to LM-X v4.9 and newer, which added HostIDs for Google Compute Engine and Azure. If you are using an older version of LM-X, refer to [documentation for earlier versions](#).*

The following table lists all HostID types that are supported by LM-X License Manager and indicates the level of security and flexibility of each type to help you decide which HostID type(s) work best for your needs. We recommend using a HostID that the end user will not change often during the valid period of the license.

**Note:** The values you enter as HostID(s) are case-insensitive (for example, "User" and "USER" are interpreted as the same entries).

HostID Type	Security	Flexibility	Notes
Hostname	Low	High	Easy to move.
Username	Low	High	Easy to move.
IP address	Low	High	Easy to move. This HostID is a good choice if the IP address is static and available.
Windows product ID	Low	High	Easy to move.
Harddisk	High	Low	Difficult to move. For optimal security, locking to the BIOS and/or Harddisk HostID is more resistant to abuse; however, these HostIDs can be acquired only on Windows platforms.
BIOS	High	Low	Difficult to move. For optimal security, locking to the BIOS and/or Harddisk HostID is more resistant to abuse; however, these HostIDs can be acquired only on Windows platforms.
Long (Unix-specific locking)	High	Low	Difficult to move.  <b>Note:</b> Mac OS X machines which have undergone extensive hardware modifications, for example had their internal hardware components replaced, may irreversibly lose their serial number (Long HostID). This can cause LM-X to fail to check out the license. Therefore, users without a well-defined serial number or those users who lost their serial number are recommended to choose a HostID type other than Long HostID for their licensing purposes.
Ethernet card (MAC address of the Ethernet card)	Medium	Low	Difficult to move; however, it is possible to change the MAC address, which somewhat compromises security.  Generally a good choice for locking either a node-locked license or a floating license, because almost all machines have an Ethernet card, and information about the Ethernet HostID is available on all platforms.
Hasp HL dongle	High	High	Dongles have superior security and flexibility; however, because the dongle is a physical device, it has the risk of being lost or broken.
Custom	High	High	Lets you develop your own algorithm to lock the application to custom hardware.
AWS Instance ID	High	High	May be somewhat time-consuming when collected in environments other than AWS; therefore, this HostID should be disabled if not being used by the client.
Google Compute Engine (GCE)	High	High	May be somewhat time-consuming when collected in environments other than GCE; therefore, this HostID should be disabled if not being used by the client.
Azure	High	High	May be somewhat time-consuming when collected in environments other than Azure; therefore, this HostID should be disabled if not being used by the client.

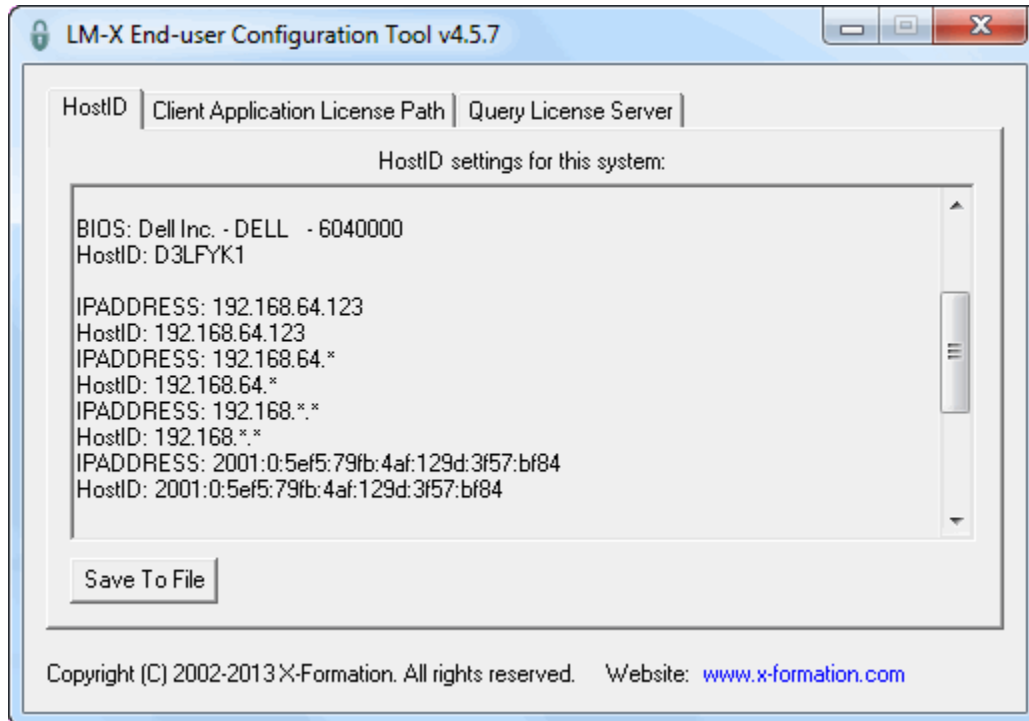
**Note:** For the highest level of security, we recommend that you use as many HostID types as possible. We also recommend using [HostID matching](#) to define custom configuration settings more accurately.

# Determining which IP address to lock your license to

The following sections describe how to determine the IP address to lock your license to for Windows and Linux.

## Determining the IP address for Windows

To find the IP address for a Windows machine, run the [LM-X End-user Configuration Tool](#) (lmxconfigtool.exe) to see your machine's HostIDs, as shown in the following example.



In the above example, you can see that you can lock your license to IP address 192.168.64.123 using its corresponding HostID, 192.168.64.123 (the HostID may not be identical to the IP address).

The IP addresses that include asterisks may be thought of as wildcards. You can use the HostIDs of these IP addresses to lock licenses to a subnetwork if needed, allowing wider access.

## Determining the IP address for Linux

To find the IP address for a Linux machine, run the [LM-X end-user utility](#) using the -hostid option (lmxendutil -hostid) at a command line to see the list of your machine's HostIDs.

## Detecting a real license

This section outlines simple guidelines that you can follow to detect a real license, and if you don't find one, detect a trial license.

For example, say you have chosen a [trial license](#) as a way of providing your potential customer with [demo versions of your software](#), but you might first want to check whether a real license is also available. (Note that there is no need to check or use a trial license if your end user has already bought a real one.) For this purpose, you can use the following code example to check out a license from a server, and when it is not available, to check out a trial license. If no licenses at all are available, the user will see the "Unable to checkout from trial license:" message, as shown below.

```
#include <stdio.h>
#include <stdlib.h>

#include "lmx.h"

LMX_HANDLE LmxHandle;

void exit_on_error(LMX_STATUS s)
{
    if (s != LMX_SUCCESS)
    {
        if (LmxHandle != NULL)
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessage(LmxHandle));
            LMX_Free(LmxHandle); // Release memory allocated by LM-X
        }
        else
        {
            fprintf(stderr, "%s\n", LMX_GetErrorMessageSimple(s));
        }
        fflush(stderr);
        exit(1);
    }
}

int main()
{
    exit_on_error(LMX_Init(&LmxHandle));

    if (LMX_Checkout(LmxHandle, "f2", 1, 0, 1) != LMX_SUCCESS)
    {
        printf("Unable to checkout from server license:\n");
        printf("%s\n", LMX_GetErrorMessage(LmxHandle));

        // If we cannot checkout a license from the server, we set the trial option and try to checkout a trial
        license
        LMX_SetOption(LmxHandle, LMX_OPT_TRIAL_USES, (LMX_OPTION) 2);
        if (LMX_Checkout(LmxHandle, "f2", 1, 0, 1) != LMX_SUCCESS)
        {
            printf("Unable to checkout from trial license:\n");
            printf("%s\n", LMX_GetErrorMessage(LmxHandle));
            LMX_Free(LmxHandle);
            return 1;
        }
    }

    LMX_Free(LmxHandle);
    return 0;
}
```

## Implementing licensing for critical-use applications

A [floating license model](#) provides an easy way to validate the end user's rights to use the software. However, floating licenses depend on a stable network connection between the client machine, on which the licensed software runs, and the license server, which delivers the license to the client.

If a machine is not connected to a network or the network has frequent failures, the most obvious solution is to use a [node-locked license model](#) instead of a floating license model. With node-locked licenses, the license file must be generated and copied onto the specific machine. The [environment variable](#) LMX\_LICENSE\_PATH must be set to allow the software to "see" the license. The XML license template file for a node-locked license has some restrictions, as described in [Application licensing strategies](#).

If you decide to use network licensing with an unstable network connection or with applications that are crucial to keep running, you should consider two solutions that will help with the reliability of license checkout for end users: [grace licensing](#) and [High Availability Licensing](#) (HAL).

For details on how to configure HAL, see [HAL\\_SERVERS](#).

## Managing license expiration for licenses running non-stop

Normally, the license expiration date is controlled while performing the checkout operation. However, in cases where a feature remains checked out continuously for a long period of time, this method of checking for license expiration may allow a license to be used past its expiration date.

To avoid license over-usage for such applications, we recommend that you check the value returned by the [LMX\\_GetExpireTime](#) API function periodically. This function gives you information about the number of hours remaining before the feature expires. When the value becomes -1, the feature is expired and you can stop the software. To continue running the software, the user must obtain a new license.

## Providing demo versions of your software

This section lists some, but far from all, ways of providing your potential customers with demo versions of your software. Depending on the level of security you need, you can choose a method that you may want to employ; in some cases you may want to use a single method, whereas sometimes combining multiple methods will provide the best possible protection for your application.

Method	Description
Trial Licenses	<p><a href="#">Trial licenses</a> let end users use an application without requiring a license. You can limit the use of trial licenses by:</p> <ul style="list-style-type: none"> <li>• A number of days (see <a href="#">LMX_OPT_TRIAL_DAYS</a> in <a href="#">LMX_SetOption</a>).</li> <li>• A number of uses (see <a href="#">LMX_OPT_TRIAL_USES</a> in <a href="#">LMX_SetOption</a>).</li> </ul> <p>By default, trial licenses are not allowed on virtual machines, as described in <a href="#">Licensing for virtual machines and cloud computing</a>. However, you can allow trial licenses to run on a virtual machine using <a href="#">LMX_OPT_TRIAL_VIRTUAL_MACHINE</a>, described in <a href="#">LMX_SetOption</a>.</p> <p><b>Note:</b> You can use LM-X API functions to gather useful feedback information for the trial licenses you are using. For example, <a href="#">LMX_GetExpirationTime</a> provides you with expiration information that you can use to remind the end user to fully license your product.</p>
Restricted Usage	<p>LM-X makes it easy to restrict application functionality for trial use. Simply omit some or all <a href="#">features</a> from the license file that you distribute. Since the feature is not specified in the license file, the corresponding module in your application will be unavailable to the user.</p>
Visual Reminders	<p>You might wish to encourage full licensing of your software by having your program show a popup periodically or when the license is a few days from expiration. You can also implement such means as switching the position of a button users must press to continue opening the demo version.</p>
Watermarks	<p>You may not want your prospective customers to use a demo version of your application for distribution or publication purposes. To discourage such use, you can add watermarks to printed output; for example, by superimposing "EVALUATION" in large text on printed pages.</p>

# Restricted access licensing

LM-X License Manager enables you to precisely control which users are authorized to access your application. Software integrated with LM-X is bound to a license file that enforces usage restrictions and is designed to accommodate standard industry scenarios, such as the following.

## Node-locked/machine-specific restrictions

Locking a license to an exact hardware identifier (hostid) is the most widely used restriction. Standalone licenses allow an application to run only on the specific machine for which they were granted. Likewise, network licenses that run on a license server prevent duplication by locking the license to a specific host. The result is an efficient copy protection mechanism: software and corresponding licenses that end up in the hands of unauthorized users simply will not run. LM-X supports a wide variety of hostids. To learn more about which type of hostid will work best for your needs, see [Determining which HostID to use](#).

## Named user restrictions

Local licenses can be locked to a specific username, while network licenses can be both user- and host-based. You can specify the number of “seats” in a license by setting the USERBASED and/or HOSTBASED directives and naming valid users and/or hosts in the license server configuration file. You have complete control over the user list, although there are restrictions on how frequently names can be changed.

## Time zone restrictions

Time-zone-restricted licenses let you control license usage based on geographical location, ensuring that licenses are used only in the country or region you intend. You can limit license usage to specific time zones for both local and network licenses by setting the TIME\_ZONES directive.

## Dongle restrictions

You can lock your application with an X-Formation or 3rd-party dongle using the special hostid type DONGLE\_HASPHL. With both local and network licenses, background checking can be done every few minutes to ensure that users do not move the dongle to other machines.

# Using CPU cores

Specifying CPU cores for node-locked or floating licenses can be useful in cases such as high-tech computing and application virtualization. This license model enables you to base licensing on your customer's computing resource requirements, increasing the pricing depth for your software offerings. For example, since a 4-CPU machine is double the speed of a 2-CPU machine, licensing the 4-CPU machine would be double the cost of licensing the 2-CPU machine.

Detailed instructions for using CPU cores for both node-locked and floating licenses are described in the following sections.

## Using CPU cores for node-locked licenses

To base node-locked licensing on CPU cores (example is for a Windows environment):

1. Start a new project in your IDE. Base your code on the example in *LM-X\_installation\_directory\examples\local\local.c*.
2. Include the LM-X library into your application (see [License your first application](#) for more information).
3. [Generate a node-locked license file using xmllicgen](#), located in *LM-X\_installation\_directory\platform-specific\_directory*; for example, *C:\LM-X\Win32\_x86*.
4. Using the [OPTIONS setting](#), specify the number of processor cores that the license will be limited to. For example, "CPU=1" limits the license to one processor core, "CPU = 2" limits the license to two processor cores, etc. (We are using the syntax "CPU=x," but you may use different syntax if desired.)

An example [XML template](#) for a node-locked license looks like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LICENSEFILE>
<FEATURE NAME="F1">
<SETTING MAJOR_VERSION="1" />
<SETTING MINOR_VERSION="0" />
<SETTING END="2015-01-01" />
<SETTING OPTIONS="CPU=1" />
<CLIENT_HOSTID>
<SETTING ETHERNET="D71D90A3763DD3BF" />
</CLIENT_HOSTID>
</FEATURE>
</LICENSEFILE>
```

5. To take the number of cores into account in your code (based on the local.c example), change the checkout parameter using [LMX\\_Checkout](#):

```
LMX_Checkout(LmxHandle, "F1", 1, 0, 1)
```

For processor-based licensing that sets the count to the number of logical CPUs, change the checkout parameter to:

```
LMX_Checkout(LmxHandle, "F1", 1, 0, LMX_LOGICAL_CPU_COUNT)
```

For processor-based licensing that sets the count to the number of physical CPUs, change the checkout parameter to:

```
LMX_Checkout(LmxHandle, "F1", 1, 0, LMX_PHYSICAL_CPU_COUNT)
```

6. To access the license setting OPTIONS "CPU=1" use *F1.szOptions*, which is available after calling [LMX\\_GetFeatureInfo](#)(*LmxHandle*, "F1").

Retrieve the integer from string *F1.szOptions* and compare it with *F1.nUsedLicCount*, which is the actual number of CPUs. If the comparison does not match your limitation, you can force the program to exit (remember to use [LMX\\_Free](#)(*LmxHandle*) upon exit to free allocated memory). Otherwise, license restrictions are fulfilled and you can continue.

7. Compile and run your application to test that it works as expected.

## Using CPU cores for floating licenses

To base floating licenses on CPU cores (example is for a Windows environment):

1. Start a new project in your IDE. Base your code on the example in *LM-X\_installation\_directory\examples\network\network.c*.
2. Include the LM-X library into your application (see [License your first application](#) for more information).



3. [Generate a floating license file using xmllicgen](#), located in *LM-X\_installation\_directory\platform-specific\_directory*; for example, C:\LM-X\Win32\_x86.

An example [XML template](#) for a floating license looks like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LICENSEFILE>
<SETTING END="2015-01-01" />
<FEATURE NAME="F1" />
<SETTING MAJOR_VERSION="1" />
<SETTING MINOR_VERSION="0" />
<SETTING COUNT="5" />
<SERVER_HOSTID>
<SETTING ETHERNET="C8A516AD01AFC9FA" />
</SERVER_HOSTID?>
</FEATURE>
</LICENSEFILE>
```

4. To take the number of cores into account in your code (based on the local.c example), change the checkout parameter using [LMX\\_Checkout](#):

```
LMX_Checkout(LmxHandle, "F1", 1, 0, 1)
```

For processor-based licensing that sets the count to the number of logical CPUs, change the checkout parameter to:

```
LMX_Checkout(LmxHandle, "F1", 1, 0, LMX_LOGICAL_CPU_COUNT)
```

For processor-based licensing that sets the count to the number of physical CPUs, change the checkout parameter to:

```
LMX_Checkout(LmxHandle, "F1", 1, 0, LMX_PHYSICAL_CPU_COUNT)
```

5. Compile your application and [start the license server](#) to test that your application works as expected.

# Using dates for version numbers

To create a license that will work only up to a specific date, you can use dates for major and minor version numbers as an alternative to specifying software release numbers.

Since LM-X requires by default that the version number in the license file be greater than or equal to the version number specified in the application, the license file will determine how long users can continue to use new updates of the software. If the software version date is later than the date specified in the license file, that version of the software will not work without an updated license.

To use dates for version numbers:

1. In the license file, set the major version number to yyyy and the minor version number to mmdd, as shown in the following example, which specifies the license file will work until 2014, October 3:

```
<?xml version="1.0" encoding="utf-8"?>
<LICENSEFILE>
<FEATURE NAME="F1">
<SETTING MAJOR_VERSION="2014" />
<SETTING MINOR_VERSION="1003" />
</FEATURE>
</LICENSEFILE>
```

2. Set the version number in the application to the current compile date (which must be earlier than or equal to the license file date), as shown in the following example, which specifies the compile date 2014, January 1:

```
int nYear = 2014;
int nMonth = 01;
int nDay = 01;

int nMajor = nYear;
int nMinor = nMonth * 10 + nDay;

LMX_Checkout(LmxHandle, "F1", nMajor, nMinor, 1);
```

3. When you create a future release that specifies a compile date later than the license file date, as shown in the following example, [LMX\\_Checkout](#) will return an error, and the user will require a new license file to run the new version of the software.

```
int nYear = 2014;
int nMonth = 12;
int nDay = 30;

int nMajor = nYear;
int nMinor = nMonth * 10 + nDay;

LMX_Checkout(LmxHandle, "F1", nMajor, nMinor, 1);
```

# Using LmxLicenseProvider as an implementation of the abstract base class provided by .NET LicenseProvider

If you've ever used .NET LicenseProvider for your licensing model, you probably know that the default implementations of this abstract class offer only basic features that are unlikely to satisfy today's needs for security and flexibility. This is where LmxLicenseProvider can help you to make your licensing model more robust.

LmxLicenseProvider lets you use LM-X License Manager's capabilities in your .NET licensing model, providing solutions for existing or home-made implementations, such as automatic server discovery, license borrowing, redundant server capabilities, and many more [LM-X features](#). In addition, the LM-X .NET API offers extensive functionality that you may find useful for modifying the behavior of LmxLicenseProvider.

The easiest and fastest way to integrate LmxLicenseProvider into your application is to compare the provided sample code, `lmxtest_licenseprovider.cs`, located under the `dotnet examples` directory in your LM-X distribution, with a clear example.

One such example, <http://msdn.microsoft.com/en-us/library/system.componentmodel.licfilelicenseprovider.aspx>, shows you how to use `LicFileLicenseProvider`, one of the default implementations of `LicenseProvider`.

The blue lines in the example below show the changes you would make to `LicFileLicenseProvider` to integrate `LmxLicenseProvider` into your application. As the example shows, the basic steps for enabling the licensing schema for your component or control class are:

1. Apply a `LicenseProviderAttribute` to the class.
2. Add a `License` field to the class.
3. Validate the license in the constructor.
4. Dispose of the granted license in the finalizer of the class or before it is called.

```
using System;
using XFormation;
using System.ComponentModel;
using System.Windows.Forms;

// Adds the LicenseProviderAttribute to the control.
[LicenseProvider(typeof(LmxLicenseProvider))]

public class MyControl : Control {
    // Creates a new, null license.
    private License license = null;
    private string strFeatureName = "feature";
    private int nMajorVer = 1;
    private int nMinorVer = 0;
    private int nCount = 1;
    public MyControl () {
        // Adds Validate to the control's constructor.
        if (nMajorVer >= 0 && nMinorVer >= 0 && nCount >= 0 && strFeatureName.Length > 0)
            license = LicenseManager.Validate(typeof(MyControl), this);
        else throw new LicenseException(typeof(MyControl), this, "Fields are not properly set.");
        // Insert code to perform other instance creation tasks here.
    }
    protected override void Dispose(bool disposing) {
        if (disposing) {
            if (license != null) {
                license.Dispose();
                license = null;
            }
        }
    }
}
```

As you can see, modifying a `LicenseProvider` base class to use `LmxLicenseProvider` requires only a few minor changes and new lines of code, making it quick and easy for you to realize the significant benefits LM-X can bring to your .NET licensing model.

# Advanced topics

Advanced topics include:

- [Using LM-X with multithreaded applications](#)
- [IPv6 support considerations](#)
- [Performing a silent installation](#)
- [MacOS on ARM support](#)

# Multithreading

LM-X can be used with multithreaded applications.

By default, you can access the LM-X API from multiple threads simultaneously. However, this default behavior may not always be sufficient.

For example, you might have a situation in which one of two threads that call [LMX\\_Checkout\(\)](#) could fail. If you subsequently wish to print out the last error message with [LMX\\_GetErrorMessage\(\)](#), the results may have been updated by the second [LMX\\_Checkout\(\)](#) call.

To avoid such cases, you can use your own mutual exclusion to provide the necessary synchronization.

In most cases, you can simultaneously call multiple functions with the same LM-X handle. If there is an error, LM-X will return [LMX\\_API\\_NOT\\_REENTRANT](#).

# IPv6 support

LM-X supports IPv6 with both the license client and license server.

LM-X supports IPv4/6 dual stack which means that it's able to communicate in both IPv4 and IPv6 without having separate versions of the applications.

When a client application wants to connect to an IPv6-enabled server, the IP address needs to be enclosed in brackets ([ ]). For example, to set environment variables for IPv6 license server:

```
LMX_LICENSE_PATH = @[1:2:3:4:5:6:7:8]
```

or

```
LMX_LICENSE_PATH = @[::1]
```

IPv6 is limited to connectivity between client and license servers, you cannot use IPv6 IP addresses in the license server configuration file.

Also note that automatic server discovery is limited to IPv4.

# Performing a silent installation

*The information on this page refers to LM-X version 4.5.2 and newer, which added silent installation for Linux. Previous versions support silent installation for Windows only.*

You can perform silent (that is, unattended) installations on Windows and Linux as described below.

## Performing a silent installation of End-user Tools and the license server on Windows

LM-X End-user Tools can be installed silently using msixec (for details on using msixec, refer to the Microsoft Developer Network at <http://msdn.microsoft.com/en-us/library/windows/desktop/aa367988>).

When performing a silent installation, it is recommended to use reduced UI level, which can be triggered by using msixec with the /qr flag (although the software can also be installed using other /q flags as well).

You can then configure the installation using the following public properties:

Property	Description	Default
INSTALLDIR	Path under which LM-X End-user Tools will be installed.	x86: C:\Program Files (x86)\X-Formation\LM-X End-user Tools version x64: C:\Program Files\X-Formation\LM-X End-user Tools version x64
INSTALLSERVER	If set to 1, LM-X License Server will be installed alongside the End-user Tools.	1
VENDORDLLPATH	Path to liblmxvendor.dll that will be copied to INSTALLDIR. Set to empty string to disable copying the dll. Has no effect if INSTALLSERVER <> 1.	N/A
INSTALLSERVICE	If set to 1, LM-X License Server will be <a href="#">installed as a service</a> . Has no effect if INSTALLSERVER <> 1.	1
SERVICEDISPLAYNAME	Display name of installed LM-X License Server service. Has no effect if INSTALLSERVICE or INSTALLSERVER <> 1.	LM-X License Server version

For example:

```
C:\lmx-sdk-v4.4.6>msiexec /i lmx-enduser-tools_v4.4.6_win32_x86.msi VENDORDLLPATH=C:\lmx-sdk-v4.4.6\win32_x86\liblmxvendor.dll INSTALLDIR=C:\lmx-enduser-v4.4.6 /qr
```

## Performing a silent installation of the SDK and End-user Tools on Linux

The LM-X SDK and End-user Tools can be installed silently on Linux as described below.

### LM-X SDK installation

You can run a silent (that is, unattended) installation of the LM-X SDK at a command line using the following options.

Option	Description
-c <i>security_config_filename</i>	Use a security configuration file from a previous installation.
-b	Compile the LM-X SDK after installation.
-t	Install the LM-X End-User Tools after LM-X SDK compilation.
-k <i>activation_key</i>	An activation key.
-f <i>license_file_path</i>	License file path.
-e accept   reject	Accept or reject the End User License Agreement.
-i <i>installation_directory</i>	Installation directory.
-h	Print this help message.

#### Example 1

The following example installs the SDK without compiling it or installing End-User Tools:

```
sudo ./lmx-sdk_v4.5.2_linux_x64.sh -- -e accept -i /opt/lmx-sdk-4.5.2 -f /home/vendor/Download/lmx.lic
```

**Example 2**

The following example:

- Installs and compiles the SDK
- Installs the End-User Tools to the path `/opt/lmx-sdk-4.5.2`
- Registers the LM-X License Server Service under the default name “lmx-serv-4.5.2.”

```
sudo ./lmx-sdk_v4.5.2_linux_x64.sh -- -e accept -i /opt/lmx-sdk-4.5.2 -f /home/vendor/Download/lmx.lic -b -t
```

**LM-X End-User Tools installation**

You can run a silent (that is, unattended) installation of the LM-X End-User Tools at a command line using the following options:

Option	Description
-s	Start the LM-X License Server after installation.
-l <i>library_path</i>	LM-X vendor library path.
-n <i>service_name</i>	Service name (defaults to <code>lmx-serv-version</code> ).
-r	Overwrite existing service.
-u <i>username</i>	Run as the specified user.
-e accept   reject	Accept or reject End User License Agreement.
-i <i>installation_directory</i>	Installation directory.
-f	Overwrite any existing files.
-p	Postpone providing vendor library.
-h	Print this help message.

**Example 1**

The following example:

- Accepts the EULA
- Installs the LM-X License Server and End-User Tools to `/opt/lmx-enduser`
- Registers the LM-X License Server Service under the default name “lmx-serv-4.5.2.”

```
sudo ./lmx-enduser-tools_v4.5.2_linux_x64.sh -- -e accept -i /opt/lmx-enduser -l /home/enduser/Download/liblmxvendor.so
```

**Example 2**

The following example:

- Accepts the EULA
- Installs the LM-X License Server and End-User Tools to `/home/enduser/lmx`
- Makes the “enduser” user an owner
- Registers the LM-X License Server Service under the name “lmxserv”
- Starts the service after installation.

```
sudo ./lmx-enduser-tools_v4.5.2_linux_x64.sh -- -e accept -i /home/enduser/lmx -l /home/enduser/Download/liblmxvendor.so -u enduser -n lmxserv -s
```



# MacOS on ARM support

*The information on this page refers to v5.1 and newer, which added support for MacOS on ARM.*

LM-X supports MacOS for both Intel-based and Apple silicon Mac computers. During LM-X SDK installation, support for ARM 64-bit architecture is verified and, if possible, LM-X tools and libraries are built as universal binaries, which contain two separate versions of the compiled code for Intel and Apple silicon Mac platforms. At runtime, the system automatically selects the correct version to run on the current platform.

## Providing a Developer ID Application Certificate

Mac computers with Apple silicon require that executables and libraries built during the installation of LM-X SDK be signed with a Developer ID Application Certificate. The Developer ID Application Certificate will be used to sign the LM-X End-user Utility and LM-X License Server. You can provide a Developer ID Application Certificate in one of the following ways:

1. [Provide the Developer ID Certificate during installation.](#)
2. Edit the `platform.mk` file and assign the Developer ID Certificate to the `DEVELOPER_ID_CERTIFICATE` variable, then recompile the LM-X SDK.
3. Use the MacOS `codesign` tool.

For example:

```
codesign --sign DEVELOPER_ID_CERTIFICATE --timestamp --options runtime FILENAME,
```

where `DEVELOPER_ID_CERTIFICATE` is your Developer ID Certificate and `FILENAME` is the name of the file to be signed.

Note the following:

- For the signing to be successful, the Developer ID Certificate must be in the MacOS keychain.
- Signing via SSH requires unlocking the keychain that contains the certificate, which can be done using the `security` command; for example:

```
security unlock-keychain keychain-name
```

- If the executable is not properly signed, a "Killed: 9" error may appear when trying to run it. If you get this error, verify the signature using the `code sign` tool; for example:

```
codesign -vv ./lmx-serv
```

The above command should return the following message:

```
/lmx-serv valid on disk
```

```
/lmx-serv, satisfies its Designated Requirement
```

# LM-X Developer's FAQ

This FAQ section addresses the most commonly-asked questions about LM-X License Server. We try to regularly update this section with new information to help you quickly and easily find the information you're looking for. If you have a question that we have not answered here, please [contact our support team](#).

Setup
<a href="#">Does LM-X offer protection against reverse engineering and tampering with executable content?</a>
<a href="#">Activating licenses using License Activation Center</a>
<a href="#">Does LM-X support Java?</a>
<a href="#">How can I obtain LM-X License Manager dongle drivers?</a>
<a href="#">How long does it take to get started with LM-X License Manager?</a>
<a href="#">Why do I get the error "LMX_BAD_SECURITY_CONFIG"?</a>
<a href="#">How do I download the latest software?</a>
<a href="#">Do our users have to upgrade their license servers after we upgrade to a new LM-X License Manager release?</a>
<a href="#">How can I lock my software to a host when using a cloud service provider?</a>
<a href="#">Compiling LM-X using Visual Studio Express Edition</a>
Training
<a href="#">Using a custom setting in a node-locked license</a>
<a href="#">Using a custom setting in a floating license</a>
<a href="#">Using the license start date to improve protection against over-usage and clock tampering</a>
<a href="#">How to implement licensing for critical-use applications</a>
<a href="#">How to use an additive license</a>
<a href="#">How do I provide license queuing to end users?</a>
<a href="#">How to do custom comparison of hostids?</a>
<a href="#">How to use the LM-X License Manager pay-per-use licensing model</a>
<a href="#">Is it possible to simultaneously run several license servers from different vendors on the same host?</a>
<a href="#">Where can I download MinGW installers?</a>
<a href="#">Why do I get a "software not allowed to run on terminal server client" error?</a>

Usage
<a href="#">How do I prevent loading multiple license files by a single instance of a license server?</a>
<a href="#">How are files protected by client store are stored?</a>
<a href="#">LM-X UI and demo do not work with Chrome version 42 and later</a>
<a href="#">Is it possible to simultaneously run several license servers from different vendors on the same host?</a>
<a href="#">Where can I download MinGW installers?</a>
<a href="#">Does LM-X have to be implemented as part of an executable file?</a>
<a href="#">Does LM-X offer protection against reverse engineering and tampering with executable content?</a>
<a href="#">How to use LmxLicenseProvider as an implementation of the abstract base class provided by .NET LicenseProvider?</a>
<a href="#">Does LM-X License Manager support vendor defined strings or custom data fields?</a>
<a href="#">Do I need multiple connections to the license server (LMX_HANDLE)?</a>
<a href="#">Why does LM-X report first-chance exception in Visual Studio?</a>
<a href="#">What Linux compilers does LM-X License Manager support?</a>
<a href="#">Do you support internet activation in LM-X License Manager?</a>
<a href="#">Should I lock my license to only one hostid or use multiple hostids?</a>
<a href="#">Why do I get the error "Unable to use license server" when I run a .NET application on a 64-bit machine?</a>
<a href="#">Can end users administer and monitor their licenses?</a>
<a href="#">We use custom hardware and can access the serial number information from this equipment. Is there a way to lock LM-X licenses to this hardware device?</a>
<a href="#">How do I renew my software?</a>
<a href="#">Why should I renew my software maintenance?</a>
<a href="#">LM-X UI and demo do not work with Chrome version 42 and later</a>
<a href="#">We use custom hardware and can access the serial number information from this equipment. Is there a way to lock LM-X licenses to this hardware device?</a>

# Setup

## Does LM-X offer protection against reverse engineering and tampering with executable content?

LM-X License Manager does not provide executable code protection against reverse engineering and accidental or deliberate tampering with the executable content. If you want to use software that provides protection from reverse engineering, you will need to use a third party solution.

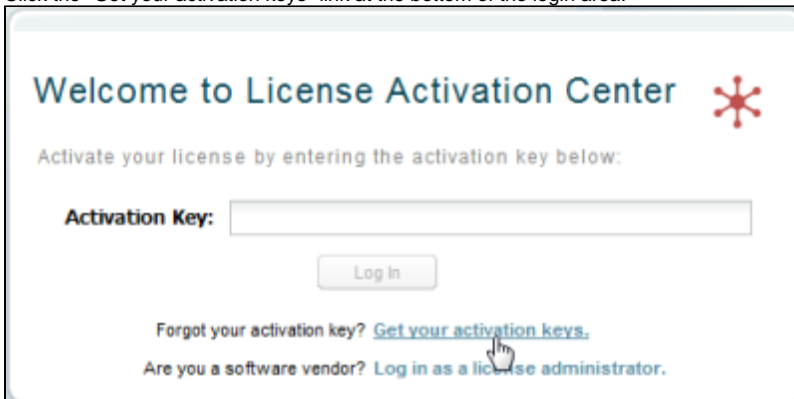
Please note that LM-X License Manager allows you to perform a clock check on the client side to ensure users have not backdated or tampered with their system clock.

For more information about protection against time tampering, see [System clock check](#).

## Activating licenses using License Activation Center

Activating your license using License Activation Center is simple:

1. Go to the [License Activation Center login page](#).
2. Click the "Get your activation keys" link at the bottom of the login area.



3. Enter the email address where you want to receive your activation keys. (If you get a message saying the email address you entered does not exist, please contact us for assistance.)
4. When you receive the email containing the activation keys, follow our [instructions for activating your license online](#). For LM-X License Manager, the license file download should be named lmx.lic and should be saved in the proper location as described in the [LM-X License Manager Quick Start](#).  
You can return to License Activation Center at any time to view your license details or re-download your license.

## Does LM-X support Java?

LM-X License Manager supports Java.

## How can I obtain LM-X License Manager dongle drivers?

On top of Sentinel HASP HL Pro dongles available at X-Formation for some time, now you can also buy 2 GB Flash driverless Sentinel HASP Max dongles.

For more information about downloading the latest Sentinel HASP drivers, go to [End User Tools Downloads](#). You can also [visit our blog](#) to learn more about **driverless** Flash Sentinel HASP Max dongles.

**Note:** If an LM-X dongle is plugged into a machine that does not have the dongle drivers installed, the Windows "Found New Hardware" dialog will appear. You can follow the prompts in this dialog to install the drivers using Windows update.

You may also download the dongle drivers attached to the bottom of this article. If desired, software vendors can ship the drivers to customers along with the dongle.

[HASPUserSetup.exe](#)

## How long does it take to get started with LM-X License Manager?

X-Formation products are designed to get you up and running quickly without a lengthy and complicated installation process or lots of documentation to go through.

See [LM-X License Manager Quick Start](#) for instructions on how to get started with LM-X.

## Why do I get the error "LMX\_BAD\_SECURITY\_CONFIG"?

If you see the error "LMX\_BAD\_SECURITY\_CONFIG" during checkout, you may have multiple security configuration files (security\_config.lmx). The license may be using one security configuration file, while the application is using a different security configuration file. Both the license and application must use the same security configuration file.

To resolve the problem:

1. Check for multiple configuration files and review the contents to determine which should be used.
2. Ensure that the security configuration file you want to use is placed in *LM-X SDK path/config/security\_config.lmx*.
3. Clean the LM-X SDK of objects from the previous compilation: For Linux, run "make clean." For Windows, run "nmake clean" or use the Clean Solution option in Visual Studio.
4. Remove the unneeded security configuration file.
5. If you have two versions of the SDK installed on different machines (for example, one for 32-bit Windows and another for 64-bit Windows), copy the security configuration file from one machine to the other.
6. Recompile the SDK.

## How do I download the latest software?

To download the latest version of LM-X, to the [LM-X License Manager download page](#).

See [LM-X License Manager Quick Start](#) for more details.

## Do our users have to upgrade their license servers after we upgrade to a new LM-X License Manager release?

When software vendors upgrade to a new LM-X License Manager release, their end users must also upgrade their LM-X license server. If they don't do this, the new server will not work with existing licenses.

See [Version compatibility](#) for more compatibility-specific information.

## How can I lock my software to a host when using a cloud service provider?

With Amazon EC2, you can use the Instance ID for the HostID (see [HostID values](#)). However, for other cloud service providers, such as Microsoft Azure, virtual machines do not have HostIDs.

See [Licensing for virtual machines and cloud computing](#) for further information on licensing software for cloud applications.

## Compiling LM-X using Visual Studio Express Edition

You may see the following error message when attempting to compile the LM-X distribution using Visual Studio Express Edition:

```
'devenv' is not recognized as an internal or external command, operable program or batch file.
```

This error is due to different file names in Visual Studio versus Visual Studio Express. The file devenv.exe is the name of the IDE executable for Visual Studio editions. For Visual Studio Express editions, each IDE executable has a different name, specifically: Visual C++ 2005/2008/2010 Express Edition - vcexpress.exe.

To solve this problem, copy the attached batch file (scroll down for attachment), devenv.bat, to a location such as C:\Windows where it will be added to the system path and recognized from the command line. LM-X will then compile normally.

# Training

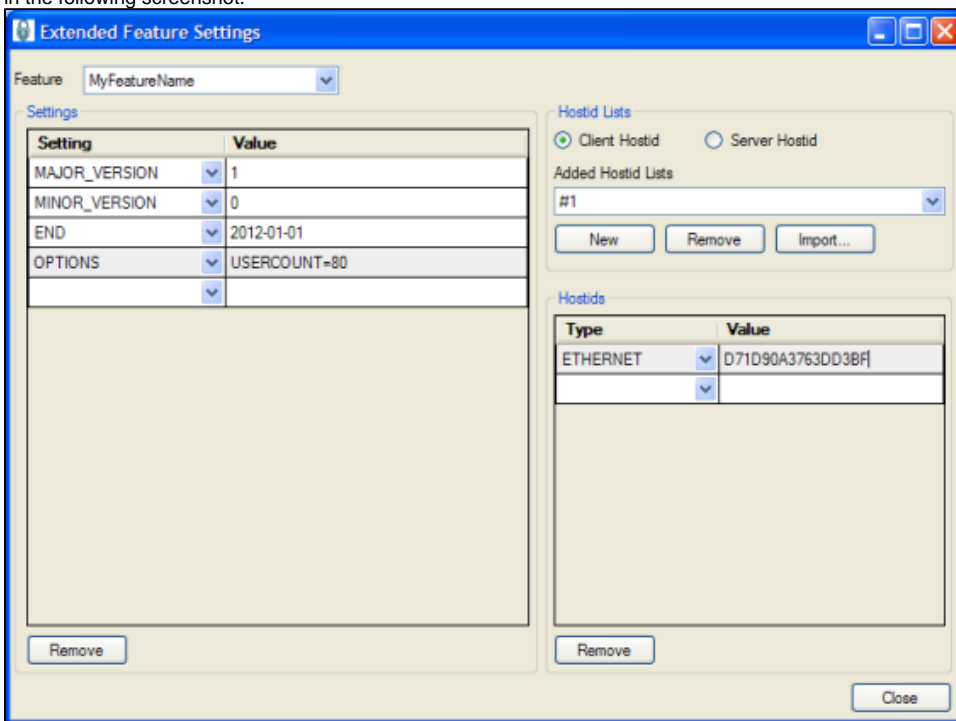
## Using a custom setting in a node-locked license

LM-X License Manager lets you specify custom settings in your license file, allowing for extensive flexibility and increased pricing depth for your software offerings. A custom setting lets you base license checkouts on certain factors that exist at the user site during run time. A custom setting may be the number of CPU's on the client's machine, currently monitored users, current connections, or similar factors. This license model enables you to tailor licensing to your target customers and characteristics of your software.

For example, say you want to offer a discounted small business license that allows a maximum of 80 concurrent users. This means the custom setting, which we'll call USERCOUNT, cannot exceed a value of 80.

To base node-locked licensing on the custom USERCOUNT setting (example made in Windows environment):

1. Start a new project in your IDE. Base your code on the example in LM-X root directory\examples\local\local.c.
2. Include the LM-X library into your application (see the LM-X Quick Start Guide for more information).
3. Generate a node-locked license file using xmllicgengui (for Windows only; use xmllicgen for other platforms), located in LM-X root directory\platform-specific directory, for example, C:\LM-X\Win32\_x86.
4. Using the OPTIONS setting, specify the custom parameter that the license will be limited to, in our example, "USERCOUNT=80" (you may use a different syntax such as "PARAM1=text1,text2,text3 PARAM2=value"). An example of the feature settings for this node-locked license are shown in the following screenshot.



An example XML template for this node-locked license looks like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LICENSEFILE>
  <FEATURE NAME="MyFeatureName">
    <SETTING MAJOR_VERSION="1" />
    <SETTING MINOR_VERSION="0" />
    <SETTING END="2012-01-01" />
    <SETTING OPTIONS="USERCOUNT=80" />
    <CLIENT_HOSTID>
      <SETTING ETHERNET="D71D90A3763DD3BF" />
    </CLIENT_HOSTID>
  </FEATURE>
</LICENSEFILE>
```

You may create multiple templates that use such a custom setting. For example, you might have a template called "Small Business" with USERCOUNT=80, another called "Corporate" with USERCOUNT=200, etc. When the end-user upgrades the license from Small Business to Corporate, all you need to do is ship a new license file. (Note that this example also includes additional security features: the HOSTID setting, which provides copy protection by allowing end-users to run the application on only one machine, and an expiration date.)

5. In your code (based on the local.c example), access the license setting OPTIONS "USERCOUNT=80" using Fl.szOptions, which is available after calling LMX\_GetFeatureInfo(LmxHandle, "MyFeatureName").

Retrieve the integer from string Fl.szOptions and compare it with the actual value in your application. If the comparison does not match your limitation, you can force the program to exit (remember to use LMX\_Free(LmxHandle) upon exit to free allocated memory). Otherwise, license restrictions are fulfilled and you can continue. Repeat the comparison whenever your parameter may change.

6. Compile and run your application to check how it works.  
The information about your custom parameter is encrypted in the license key and, by default, is visible in the license file. If you don't want the custom parameter displayed in the license file, use the KEYCOMMENT setting instead of the OPTIONS setting.

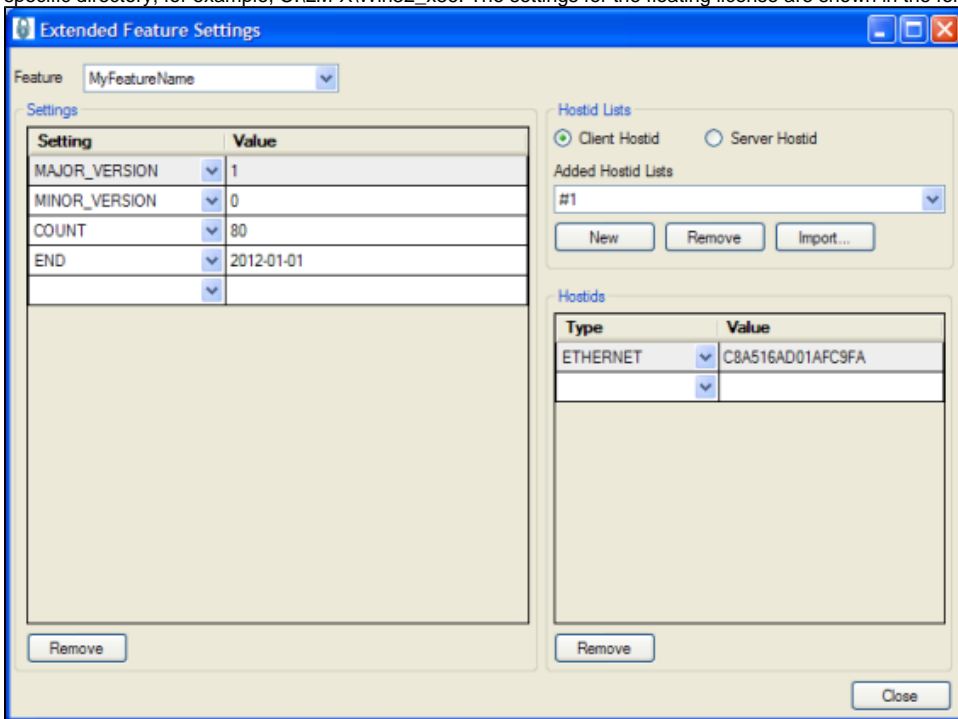
## Using a custom setting in a floating license

LM-X License Manager lets you specify custom settings in your license file, allowing for extensive flexibility and increased pricing depth for your software offerings. A custom setting lets you base license checkouts on certain factors that exist at the user site during run time. A custom setting may be the number of CPU's on the client's machine, currently monitored users, current connections, or similar factors. This license model enables you to tailor licensing to your target customers and characteristics of your software.

For example, say you want to offer a discounted small business license that allows a maximum of 80 concurrent users. This means the custom setting, which we'll call USERCOUNT, cannot exceed a value of 80.

To base floating licensing on the custom USERCOUNT setting (example made in Windows environment):

1. Start a new project in your IDE. Base your code on the example in LM-X root directory\examples\network\network.c.
2. Include the LM-X library into your application (see the LM-X Quick Start Guide for more information).
3. Generate a floating license file using xmllicgengui (for Windows only; use xmllicgen for other platforms), located in LM-X root directory\platform-specific directory; for example, C:\LM-X\Win32\_x86. The settings for the floating license are shown in the following screenshot:



An example XML template for a floating license looks like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LICENSEFILE>
<SETTING END="2012-01-01" />
<FEATURE NAME="MyFeatureName" />
<SETTING MAJOR_VERSION="1" />
<SETTING MINOR_VERSION="0" />
<SETTING COUNT="80" />
<SERVER_HOSTID>
<SETTING ETHERNET="C8A516AD01AFC9FA" />
</SERVER_HOSTID>
</FEATURE>
</LICENSEFILE>
```

4. The following example assumes that your custom parameter in the application code is an integer called `MyCustomParameter`. To take the custom parameter into account in your code (based on `network.c` example), change the checkout parameter:

```
LMX_Checkout(LmxHandle, "MyFeatureName", 1, 0, 1)
```

to:

```
LMX_Checkout(LmxHandle, "MyFeatureName", 1, 0, MyCustomParameter)
```

This will take into account the initial value of the custom parameter. We can also assume the custom parameter might change during the program runtime. For this reason, we'll define a variable called `MyCustomParameterDifference`, which will be the difference between the previous and current value. Whenever the custom parameter changes, the `MyCustomParameterDifference` value must be updated and `LMX_Checkout` or `LMX_Checkin` called depending on the negative or positive change. If the difference is positive we'll call:

```
LMX_Checkout(LmxHandle, "MyFeatureName", 1, 0, MyCustomParameterDifference)
```

If the difference is negative we'll call:

```
LMX_Checkin(LmxHandle, "MyFeatureName", ( (-1)*MyCustomParameterDifference) )
```

5. Compile your application. To check how it works, remember to start the license server (refer to the LM-X Quick Start Guide for more information).

In the example illustrated above, the protected application takes a number of licenses from the license pool according to a parameter within your application. During the application runtime, it can take more licenses or release licenses according to the parameter changes.

Another licensing model where one application uses more than one license is token-based licensing. The primary use for token-based licensing is to let users purchase a number of features that each require one or more real licenses. This gives users a "pool" of licenses they can draw upon for license checkout requests, providing the flexibility to use various feature combinations as needed.

## Using the license start date to improve protection against over-usage and clock tampering

You can set the starting date that your application is available for checkout by using the `START` setting in your [XML license template file](#). By combining the `START` and `END` date settings (for example, `<SETTING START="2010-09-01"/>` and `<SETTING END="2011-09-01"/>`), you can specify the exact period of time that the license is active.

In this way, you can distribute the license far earlier than the start date without risk of over-usage. Setting the start date in a license you distribute earlier than the start date also helps you avoid any possible issues with users turning the clock back before their first use of the license. The system clock tampering will be detected immediately and the license will not be available for checkout until the clock is set to the proper date.

## How to implement licensing for critical-use applications

[LM-X License Manager](#) provides two effective solutions that help with the reliability of license checkout for end users if you decide to use network licensing with an unstable network connection or with applications that are crucial to keep running.

Please refer to [Implementing licensing for critical-use applications](#) for more information.

## How to use additive licenses

Adding licenses for software features that a customer buys later than the initial purchase of a license can be easily accomplished by using multiple additive licenses, which don't require that you change the original license. For details on using additive licenses, see [KEYTYPE](#).

## How do I provide license queuing to end users?

If you allow license queuing for your end users (it is enabled by default), you can test that it works properly by taking the following steps:

1. Enable queuing for Client A. (For information about enabling queuing, see [Queuing licenses](#).)
2. Perform a checkout of the maximum number of available licenses from Client B. (This client does not have to have queuing enabled.)
3. Attempt to checkout one or more licenses from a Client A. Client A should receive an `LMX_QUEUE` return status (that is, the value of `LMX_STATUS` returned by the unsuccessful `LMX_Checkout` operation) when trying to checkout a license, since all the licenses are used.
4. Check one or more licenses from Client B. The checkout attempt from Client A should now be successful.

Note that you should call `LMX_Checkout` again to see if a license is available in the case that `LMX_QUEUE` is returned.

## How to do custom comparison of hostids?

This tutorial describes how to do custom comparison of hostids. This is useful when you want to implement custom verification. For example, you can specify match rules that allow a license to succeed when 2 of 3 hostids are valid, rather than requiring strict 1:1 matching.

Start by setting a custom hostid function. This allows you to make custom hostids in license files with type HOSTID\_CUSTOM:  
`LMX_SetOption(LmxHandle, LMX_OPT_CUSTOM_HOSTID_FUNCTION, (LMX_OPTION) MyHostid);`

Next line lets you customize comparison of hostids in order to allow partial matching instead of 1:1 matching.  
`LMX_SetOption(LmxHandle, LMX_OPT_HOSTID_COMPARE_FUNCTION, (LMX_OPTION) HostidCompare);`

MyHostid and HostidCompare functions must be implemented, obviously:

```
LMX_STATUS LMX_CALLBACK MyHostid(LMX_HOSTID *pHostid, int *npHostids) {...}
LMX_STATUS LMX_CALLBACK HostidCompare(int nKeyHostidType, const LMX_HOSTID pLicenseHostid[], int nLicenseHostids, const LMX_HOSTID
pSystemHostid[], int nSystemHostids) {...}
```

More detailed code example is available on request.

## How to use the LM-X License Manager pay-per-use licensing model

The LM-X pay-per-use licensing model lets you bill your customers based on their actual usage of your software. To determine the usage, you ask the customer to write the usage information to a pay-per-use SQLite database each month (or other period) and send the database file to you. (You can find out more about SQLite, a free open source database, at <http://www.sqlite.org/>.)

Customers can also import pay-per-use databases into License Statistics to monitor their license usage.

To enable pay-per-use, the user must specify the database path in the USAGE\_DATABASE entry in the license server configuration file (lmxserv.cfg). The user can also specify the detail level included in the pay-per-use data by editing the USAGE\_LEVEL entry. Detailed information about these settings can be found in the LM-X End User Guide.

The database is authenticated. You can check for any tampering with the data by running the following command using the lmutil utility:

```
lmxendutil -readusagedb usage.db
```

Although you can check that the database contains valid data, accurate billing for pay per use is entirely dependent on the customer writing their usage data to the database. LM-X does not enforce the use of the pay-per-use database, so you should use pay per use only with customers who you trust to reliably report their usage information.

For more information about the Pay Per Use feature, see the Pay Per Use chapter in the LM-X End Users Guide. If you have further questions about this or other LM-X features, please [contact X-Formation support](#).

## Is it possible to simultaneously run several license servers from different vendors on the same host?

LM-X License Manager is designed to let you simultaneously run multiple instances of different license servers from different vendors on the same host. (Both license servers will share UDP port 6200. )

You should also ensure that each server runs on a unique TCP port.

## Where can I download MinGW installers?

You can [compile the LM-X SDK](#) on a Windows machine using MinGW, a minimalist development environment for Microsoft Windows applications.

Click [here](#) to download MinGW for 32 and 64-bit Windows.

## Why do I get a "software not allowed to run on terminal server client" error?

A "software not allowed to run on terminal server client" error indicates that the SHARE = TERMINALSERVER directive has not been included in the license file.



# Usage

## How do I prevent loading multiple license files by a single instance of a license server?

To prevent the license server from loading multiple [license files](#), you should embed an ID string into the COMMENT field against each feature in a single license file. This way, all features in one file will then have the same ID, but each license file will have a different id; therefore, if a license file is remade, it will be assigned a new ID.

With LmxServerStartup override function, we will validate all of the license features of the server to make sure they all have the same ID which will prevent a user from using multiple license file with a single instance of the server and also from copying feature lines from one license file to another.

## How are files protected by client store are stored?

Client store is one of the methods used by LM-X to protect your application. Client store uses [secure store](#) by supplying a means of storing user content more securely. The default locations where the files are stored differ depending on an operating system. There is no way of removing all files, but you can delete individual pieces of content stored using LM-X API.

Please note that determining whether a file is corrupted is impossible.

See [Client store](#) for more information.

## LM-X UI and demo do not work with Chrome version 42 and later

As of Chrome version 42, NPAPI is disabled by default. If LM-X online demo or UI do not work with your Chrome version 42 and later browser, you probably need to enable the NPAPI plug-in, as recommended by [Java online help resources](#).

To enable NPAPI:

1. Type "[chrome://flags/#enable-npapi](#)" into your Chrome browser bar.
2. Click the **Enable** link for the **Enable NPAPI** configuration option.
3. Click the **Relaunch** button at the bottom of your configuration page.

## Is it possible to simultaneously run several license servers from different vendors on the same host?

LM-X License Manager is designed to let you simultaneously run multiple instances of different license servers from different vendors on the same host. (Both license servers will share UDP port 6200. )

You should also ensure that each server runs on a unique TCP port.

See [License server issues](#) for more information.

## Where can I download MinGW installers?

You can [compile the LM-X SDK](#) on a Windows machine using MinGW, a minimalist development environment for Microsoft Windows applications.

Click [here](#) to download MinGW for 32 and 64-bit Windows.

## Does LM-X have to be implemented as part of an executable file?

LM-X License Manager is not an executable wrapper, and therefore has to be included into the source code prior to compilation.

## Does LM-X offer protection against reverse engineering and tampering with executable content?

LM-X License Manager does not provide executable code protection against reverse engineering and accidental or deliberate tampering with the executable content. If you want to use software that provides protection from reverse engineering, you will need to use a third party solution.

Please note that LM-X License Manager allows you to perform a clock check on the client side to ensure users have not backdated or tampered with their system clock.

See [System clock check](#) for more information about protection against time tampering.

## How to use LmxLicenseProvider as an implementation of the abstract base class provided by .NET LicenseProvider?

The default implementations of .NET LicenseProvider for your licensing model offer only basic features that are unlikely to satisfy today's needs for security and flexibility. This is where LmxLicenseProvider can help you to make your licensing model more robust.

For more information about how LmxLicenseProvider can boost LM-X License Manager's capabilities in your .NET licensing model, see [Using LmxLicenseProvider as an implementation of the abstract base class provided by .NET LicenseProvider](#).

## Does LM-X License Manager support vendor defined strings or custom data fields?

LM-X supports vendor-defined strings and custom data fields, and makes it possible for you to extend LM-X with additional information in your licenses.

You can set the DATA field in the license to specify a comment, such as additional licensing options or restrictions, as described in [FEATURE settings](#).

## Do I need multiple connections to the license server (LMX\_HANDLE)?

You need only one connection to the license server (LMX\_HANDLE), which is done when you call LMX\_Init. You should call LMX\_Init only once, and continue to use the single handle throughout the lifetime of your application.

See [LMX\\_Init](#) for more information.

## Why does LM-X report first-chance exception in Visual Studio?

See [Installation issues](#) for more information about this problem and steps to resolve it.

## What Linux compilers does LM-X License Manager support?

For Linux (x86 and x64), LM-X supports the use of GCC (GNU Compiler Collection). Other compilers that use output compatible with GCC, including custom compilers, as well as Clang, will also work but are not officially supported.

## Do you support internet activation in LM-X License Manager?

By using our [License Distribution Service](#) you can enable activations directly from within your application. This means you can allow purchasing users to process their orders and let evaluating users obtain an evaluation license with no additional work on your part.

## Should I lock my license to only one hostid or use multiple hostids?

Locking your licensed application to an Ethernet card, BIOS or harddisk HostID provides enough security to meet most needs. If you require more security, you may lock to multiple HostIDs using one of methods described in [Deciding whether to lock a license to one or multiple HostIDs](#).

## Why do I get the error "Unable to use license server" when I run a .NET application on a 64-bit machine?

When you have a 32-bit .NET application protected with a 32-bit LM-X library (lmxnet.dll), the application will run successfully on a 32-bit machine, but if the application is run on a 64-bit machine the following error might occur:

```
"Unable to use license server."
```

The problem is that .NET attempts to run the 64-bit version of the .NET framework when the application is attempted to be used on a 64-bit machine.

To resolve this problem, do one of the following:

- If it is important to provide 64-bit support, build your application with a 64-bit version of the lmxnet.dll.
- If providing 64-bit support is not necessary, force your application to be 32-bit by using the switch `/platform:x86` when compiling your code. This solution requires virtually no work and will solve the problem.

## Can end users administer and monitor their licenses?

With LM-X License Manager, software vendors can provide end users with a license file they can use as a standalone license or with a license server. Our license monitoring software, [License Statistics](#), lets endusers monitor the use of floating or standalone licenses.

## We use custom hardware and can access the serial number information from this equipment. Is there a way to lock LM-X licenses to this hardware device?

You can lock LM-X to your hardware device by specifying a custom hostid and using the LMX\_HOSTID\_CUSTOM hostid type together with LMX\_Hostid or LMX\_HostidSimple to retrieve the LM-X hostid. For this to work, you must write a custom hostid callback function as shown in the hostid example that comes with LM-X, and then register this function using:

```
LMX_SetOption(LmxHandle, LMX_OPT_CUSTOM_HOSTID_FUNCTION, (LMX_OPTION) MyHostid);
```

After registering the custom callback function, LMX\_Hostid and LMX\_HostidSimple will call the function and retrieve the LM-X hostid. Note that the LM-X Config Tool and LM-X End-user Utility will not print out custom hostids. To let users send back custom hostids, you must make your own hostid utility and send it to users.

## How do I renew my software?

There is no need to contact us regarding your software renewal. When your X-Formation software product is nearing expiration, we'll send you an automatically generated email with a Proforma invoice for license renewal.

- For 60-day payment terms, the renewal invoice will be sent to you 90 days prior to the software expiration date.
- For 30-day payment terms, the renewal invoice will be sent to you 60 days prior to the software expiration date.

If you require your Purchase Order number to be included in the invoice, you can add it using the form at <http://www.x-formation.com/sa/set-po-number.html>. You will need your activation key and quote/Proforma number to complete this form. For your convenience, your emailed quote includes a personalized link that automatically fills in your activation key and quote/Proforma number on this form.

Please note that X-Formation's invoicing system is completely automated; therefore, if you require a different method of invoicing other than the automatic Proforma invoice or require a printed invoice, a fee of 50 EUR will be charged for administrating the invoice.

## Why should I renew my software maintenance?

There are many important benefits to keeping your X-Formation software maintenance up to date.

All perpetual licenses include 1 year of updates and support (any licenses purchased after the original purchase are added to your current maintenance plan). After the first year, maintenance is available for just 25% of the current, regular (non-discounted) price. This is a significant savings over allowing maintenance to lapse and then repurchasing the software at a later date.

Renewing your software maintenance gives you continued access to the best product support in the industry, which includes:

- Prompt responses to your technical problems by phone or email.
- Participation in [Customer Driven Development](#).
- Access to your existing license files using [License Activation Center](#).

Most importantly, you'll continue to have uninterrupted access to new versions of the software as they're released. We're constantly improving our products, based directly on feedback from customers like you, so you won't want to miss out on our upcoming releases. (Note that your maintenance must be up to date to purchase any add-on software product.)

## LM-X UI and demo do not work with Chrome version 42 and later

As of Chrome version 42, NPAPI is disabled by default. If LM-X online demo or UI do not work with your Chrome version 42 and later browser, you probably need to enable the NPAPI plug-in, as recommended by [Java online help resources](#).

To enable NPAPI:

1. Type "[chrome://flags/#enable-npapi](#)" into your Chrome browser bar.
2. Click the **Enable** link for the **Enable NPAPI** configuration option.
3. Click the **Relaunch** button at the bottom of your configuration page.

## We use custom hardware and can access the serial number information from this equipment. Is there a way to lock LM-X licenses to this hardware device?

You can lock LM-X to your hardware device by specifying a custom hostid and using the LMX\_HOSTID\_CUSTOM hostid type together with LMX\_Hostid or LMX\_HostidSimple to retrieve the LM-X hostid. For this to work, you must write a custom hostid callback function as shown in the hostid example that comes with LM-X, and then register this function using:

```
LMX_SetOption(LmxHandle, LMX_OPT_CUSTOM_HOSTID_FUNCTION, (LMX_OPTION) MyHostid);
```

After registering the custom callback function, LMX\_Hostid and LMX\_HostidSimple will call the function and retrieve the LM-X hostid. Note that the LM-X Config Tool and LM-X End-user Utility will not print out custom hostids. To let users send back custom hostids, you must make your own hostid utility and send it to users.

# Release Notes

LM-X License Manager Release Notes provide information on the improvements and bug fixes introduced in each release.

The following pages include release notes for both major and minor (bugfix) releases of the latest version of X-Formation's [LM-X License Manager](#).

# LM-X License Manager 5 Release Notes

These links will take you to the Release Notes for the following major and minor (bug fix) releases of LM-X License Manager 5:

- [LM-X License Manager v5.2.1 Release Notes](#)
- [LM-X License Manager v5.2 Release Notes](#)
- [LM-X License Manager v5.1.2 Release Notes](#)
- [LM-X License Manager v5.1.1 Release Notes](#)
- [LM-X License Manager v5.1 Release Notes](#)
- [LM-X License Manager v5.0.1 Release Notes](#)
- [LM-X License Manager v5.0 Release Notes](#)

# LM-X License Manager v5.2.1 Release Notes

LM-X **License Manager** version 5.2.1 includes the enhancements and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).


## Enhancements

LM-X v5.2.1 includes the following enhancements.

Issue #	Description
LMX-4162	License server now logs a warning when an unlimited license will be reserved.

## Fixes

LM-X v5.2.1 includes the following fixes.

Issue #	Description
LMX-4319	Resolved issue with using <a href="#">secure store</a> between different LM-X versions. <div> The issue with secure store exists in versions 5.2 and 5.1.2. We recommend that you do not use these versions if you are using secure store.</div>
LMX-4293	Changed the default installation path on MacOS (for both SDK and EUT).
LMX-4291	Fixed problem with missing file information for the vendor library on Windows.

# LM-X License Manager v5.2 Release Notes

LM-X **License Manager** version 5.2 includes the enhancements and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v5.2 includes the following enhancements.

Issue #	Description
LMX-4267	Improved installer description for providing Developer Certificate ID on MacOS.
LMX-4263	Added support for Visual Studio 2022.
LMX-4254	Added support for dongle HostIDs on MacOS on ARM (Apple Silicon).
LMX-4152	Added support for Linux on ARM.
LMX-4205	Made improvements to reduce SDK compilation time.

## Fixes

LM-X v5.2 includes the following fixes.

Issue #	Description
LMX-4294	Removed Developer ID from the MacOS SDK installer.
LMX-4279	Improved error message for cases where the LM-X internal license is too old.
LMX-4268	Improved implementation of HTTP-based HostIDs (GCE, Azure, AWS).
LMX-4266	Fixed problem with duplicated "-arch" options during SDK build on MacOS.
LMX-4244	Fixed problem with system clock check on Windows.

# LM-X License Manager v5.1.2 Release Notes

LM-X **License Manager** version 5.1.2 includes the enhancements and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v5.1.2 includes the following enhancements.

Issue #	Description
LMX-4227	Made several improvements to LMX_GetLicenseInfo performance, including: <ul style="list-style-type: none"><li>• Removed network message size limit</li><li>• Added new setting for denial storage period, which facilitates faster denial handling during license information response generation</li><li>• Added new dynamic thread pool for handling client connections</li><li>• Improved socket handling on the client side</li></ul>
LMX-4196	Added new error code ("LM-X license is too new") for cases where the license key version is newer than the client can understand.

## Fixes

LM-X v5.1.2 includes the following fixes.

Issue #	Description
LMX-4251	Resolved problem with glibc internal error on netlink descriptors.
LMX-4249	Fixed signing issue for MacOS.
LMX-4247	Fixed problem with uncaught exception when file descriptor limit was reached during license server startup.



# LM-X License Manager v5.1.1 Release Notes

LM-X **License Manager** version 5.1.1 includes the enhancement and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v5.1.1 includes the following enhancement.

Issue #	Description
LMX-4235	Added new option to the license server configuration file that lets you specify the number of concurrent threads handling client connections.

## Fixes

LM-X v5.1.1 includes the following fixes.

Issue #	Description
LMX-4226	Added missing flag to Linux SDK.
LMX-4222	Fixed issue with license checkin order when sharing additive licenses.
LMX-4218	Fixed problem with missing .NET documentation in the SDK installer.

# LM-X License Manager v5.1 Release Notes

LM-X **License Manager** version 5.1 includes several enhancements and fixes. The improvements made in this **release**, listed below, were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v5.1 includes the following enhancements and changes.

Issue #	Description
LMX-4207	Improved feature picking algorithm to enable more sophisticated licensing scenarios.
LMX-4199	Banned sharing both token and its dependency at the same time to avoid confusion.
LMX-4144	Added <a href="#">support for MacOS on ARM</a> .

## Fixes

LM-X v5.1 includes the following fixes.

Issue #	Description
LMX-4201	Added missing dependency to LM-X service on Windows.
LMX-4198	Fixed a problem with the init script running as a daemon user.
LMX-4130	Removed duplicate license server shutdown message in Windows.
LMX-4123	Fixed issue with creating the license server file lock directory.

# LM-X License Manager v5.0.1 Release Notes

LM-X License Manager version 5.0.1 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v5.0.1 includes the following fixes.

Issue #	Description
LMX-4161	Fixed an issue with running the license server init script as a daemon user.
LMX-4172	Fixed socket leak in automatic server discovery procedure.
LMX-4184	Fixed an issue with upgrade licenses that caused the license server to hang.

# LM-X License Manager v5.0 Release Notes

LM-X **License Manager** version 5.0 includes several enhancements and fixes. The improvements made in this **release**, listed below, were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v5.0 includes the following enhancements and changes.

Issue #	Description
LMX-4129	Added ability to share token-based licenses.
LMX-4125	Added ability to queue token-based licenses.
LMX-4124	Added an "upgrade" license type, which lets you add licenses for software features that a customer buys after the initial purchase of a license without changing the original license, and improves on the "additive" license type.

## Fixes

LM-X v5.0 includes the following fixes.

Issue #	Description
LMX-4160	Fixed bug with limiting all users.
LMX-4151	Fixed issue with disregarding change to Pay Per Use interval.

# LM-X License Manager 4 Release Notes

These links will take you to the Release Notes for the following major and minor (bug fix) releases of LM-X License Manager 4:

- [LM-X License Manager v4.9.25 Release Notes](#)
- [LM-X License Manager v4.9.24 Release Notes](#)
- [LM-X License Manager v4.9.23 Release Notes](#)
- [LM-X License Manager v4.9.22 Release Notes](#)
- [LM-X License Manager v4.9.21 Release Notes](#)
- [LM-X License Manager v4.9.20 Release Notes](#)
- [LM-X License Manager v4.9.19 Release Notes](#)
- [LM-X License Manager v4.9.18 Release Notes](#)
- [LM-X License Manager v4.9.17 Release Notes](#)
- [LM-X License Manager v4.9.16 Release Notes](#)
- [LM-X License Manager v4.9.15 Release Notes](#)
- [LM-X License Manager v4.9.14 Release Notes](#)
- [LM-X License Manager v4.9.13 Release Notes](#)
- [LM-X License Manager v4.9.12 Release Notes](#)
- [LM-X License Manager v4.9.11 Release Notes](#)
- [LM-X License Manager v4.9.10 Release Notes](#)
- [LM-X License Manager v4.9.9 Release Notes](#)
- [LM-X License Manager v4.9.8 Release Notes](#)
- [LM-X License Manager v4.9.7 Release Notes](#)
- [LM-X License Manager v4.9.6 Release Notes](#)
- [LM-X License Manager v4.9.5 Release Notes](#)
- [LM-X License Manager v4.9.4 Release Notes](#)
- [LM-X License Manager v4.9.3 Release Notes](#)
- [LM-X License Manager v4.9.2 Release Notes](#)
- [LM-X License Manager v4.9.1 Release Notes](#)
- [LM-X License Manager v4.9 Release Notes](#)
- [LM-X License Manager v4.8.13 Release Notes](#)
- [LM-X License Manager v4.8.12 Release Notes](#)
- [LM-X License Manager v4.8.11 Release Notes](#)
- [LM-X License Manager v4.8.10 Release Notes](#)
- [LM-X License Manager v4.8.9 Release Notes](#)
- [LM-X License Manager v4.8.8 Release Notes](#)
- [LM-X License Manager v4.8.7 Release Notes](#)
- [LM-X License Manager v4.8.6 Release Notes](#)
- [LM-X License Manager v4.8.5 Release Notes](#)
- [LM-X License Manager v4.8.4 Release Notes](#)
- [LM-X License Manager v4.8.3 Release Notes](#)
- [LM-X License Manager v4.6.5 Release Notes](#)
- [LM-X License Manager v4.6.4 Release Notes](#)
- [LM-X License Manager v4.6.3 Release Notes](#)
- [LM-X License Manager v4.6.2 Release Notes](#)
- [LM-X License Manager v4.6.1 Release Notes](#)
- [LM-X License Manager v4.6 Release Notes](#)
- [LM-X License Manager v4.5.8 Release Notes](#)
- [LM-X License Manager v4.5.7 Release Notes](#)
- [LM-X License Manager v4.5.6 Release Notes](#)
- [LM-X License Manager v4.5.5 Release Notes](#)
- [LM-X License Manager v4.5.4 Release Notes](#)
- [LM-X License Manager v4.5.3 Release Notes](#)
- [LM-X License Manager v4.5.2 Release Notes](#)
- [LM-X License Manager v4.5.1 Release Notes](#)
- [LM-X License Manager v4.5 Release Notes](#)
- [LM-X License Manager v4.4.9 Release Notes](#)
- [LM-X License Manager v4.4.8 Release Notes](#)
- [LM-X License Manager v4.4.7 Release Notes](#)
- [LM-X License Manager v4.4.6 Release Notes](#)
- [LM-X License Manager v4.4.5 Release Notes](#)
- [LM-X License Manager v4.4.4 Release Notes](#)
- [LM-X License Manager v4.4.3 Release Notes](#)
- [LM-X License Manager v4.4.2 Release Notes](#)
- [LM-X License Manager v4.4.1 Release Notes](#)
- [LM-X License Manager v4.4 Release Notes](#)

- [LM-X License Manager v4.8.2 Release Notes](#)
- [LM-X License Manager v4.8.1 Release Notes](#)
- [LM-X License Manager v4.8 Release Notes](#)
- [LM-X License Manager v4.7.6 Release Notes](#)
- [LM-X License Manager v4.7.5 Release Notes](#)
- [LM-X License Manager v4.7.4 Release Notes](#)
- [LM-X License Manager v4.7.3 Release Notes](#)
- [LM-X License Manager v4.7.2 Release Notes](#)
- [LM-X License Manager v4.7.1 Release Notes](#)
- [LM-X License Manager v4.7 Release Notes](#)

# LM-X License Manager v4.9.25 Release Notes

LM-X License Manager version 4.9.25 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.25 includes the following enhancements.

Issue #	Description
LMX-4115	Error messages related to "allow" and "deny" rules now include the feature name.
LMX-4116	Implemented customer-suggested LM-X SDK makefiles enhancement.
LMX-4117	Made remote access password requirements consistent.
LMX-4142	Implemented new error codes displayed on failure when using the <code>lmxendutil -removeuser</code> command to remove a user from the license server: <code>LMX_USER_SESSION_NOT_FOUND</code> , <code>LMX_MINIMUM_REMOVAL_TIME_NOT_PASSED</code> and <code>LMX_USER_REMOVAL_DISABLED</code> .

## Fixes

LM-X v4.9.25 includes the following fixes.

Issue #	Description
LMX-4113	Fixed issue with simultaneous use of license limit and license queuing.
LMX-4114	Fixed issue with simultaneous use of dynamic reservations and license queuing.
LMX-4119	Fixed duplicated log message in client log.
LMX-4126	Fixed issue with permissions for the <code>lacutil</code> command line utility.
LMX-4128	Fixed issue with multiple initializations of <code>LMX_HANDLE</code> .
LMX-4131	Fixed issue with simultaneous use of static reservations and license queuing.

# LM-X License Manager v4.9.24 Release Notes

LM-X License Manager version 4.9.24 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.24 includes the following enhancements.

Issue #	Description
LMX-4036	Added ability to specify a feature name when removing a user/host from the license server using the <code>lmxendutil -removeuser</code> command.

## Fixes

LM-X v4.9.24 includes the following fixes.

Issue #	Description
LMX-4051	Fixed issue with silent installation on Linux.
LMX-4065	Added version information to .NET library.
LMX-4068	LM-X configuration tool no longer crashes while querying the license server when host is empty.
LMX-4070	Group names and group members created in the license server configuration file are no longer case-sensitive.
LMX-4072	Fixed inconsistency in IP address groups on the license server.
LMX-4073	Fixed typo in SDK header files.
LMX-4093	Banned R_386_GOT32X relocations to support older version of GCC on Linux.
LMX-4110	Fixed issue with LMX_Init hanging on startup when a client library was used as a DLL.
LMX-4112	Fixed bug in Java client that prevented the removal of uploaded licenses.



# LM-X License Manager v4.9.23 Release Notes

LM-X License Manager version 4.9.23 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).



## Important

This release of LM-X requires rebuilding the liblmxvendor library.

## Enhancements

LM-X v4.9.23 includes the following enhancements.

Issue #	Description
LMX-3412	Enabled continuous HostID checking to be configured by the vendor.
LMX-4024	LMX_GetLicenseInfo now returns LMX_FEATURE_DUPLICATED when encountering duplicated local features.
LMX-4034	REPLACE property of the license template now allows white-space characters. This enables copy/paste of a license KEY from a previously generated license file.
LMX-4050	LM-X API returns LMX_NOT_INITIALIZED when LMX_Init has not yet been called.

## Fixes

LM-X v4.9.23 includes the following fixes.

Issue #	Description
LMX-3988	Fixed issue with returning additive licenses.

# LM-X License Manager v4.9.22 Release Notes

LM-X License Manager version 4.9.22 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).


## Fixes

LM-X v4.9.22 includes the following fixes.

Issue #	Description
LMX-3903	LMX_GetLicenseInfo no longer makes redundant calls to the license server.
LMX-4021	Fixed issue with trailing slash missing in TMPDIR environment variable.
LMX-4027	LMX_Putenv now works with the _dupenv_s function on Windows.
LMX-4033	LM-X SDK no longer requires Advapi32.lib.

# LM-X License Manager v4.9.21 Release Notes

LM-X License Manager version 4.9.21 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).



Due to improvements to 64-bit support in this release, Pay-Per-Use databases created using a previous version of the LM-X license server no longer work in v4.9.21 and newer.

You can still use Imxendutil from prior versions for verifying databases created prior to v4.9.21; however, the OS version (32-bit or 64-bit) of Imxendutil and the database must match. That is, you must use a 32-bit version of Imxendutil with a database generated by a 32-bit license server, and a 64-bit version of Imxendutil with a database generated by a 64-bit database.

Databases created by LM-X v4.9.21 or newer should be verified by the new Imxendutil. We also recommend that you specify a new file for the database, since the database that was created by the older version and is still being used in LM-X v4.9.21 and newer won't be verified by Imxendutil.

## Enhancements

LM-X v4.9.21 includes the following enhancement.

Issue #	Description
LMX-3987	Validation of LM-X vendor names has been made stricter.

## Fixes

LM-X v4.9.21 includes the following fixes.

Issue #	Description
LMX-3953	The LM-X installer no longer requires root privileges.
LMX-3965	HyperV detection no longer returns false positives.
LMX-3986	Pay-per-use database created on a 32-bit machine can now be used on a 64-bit machine.
LMX-4007	Fixed issues with replacing licenses. License replace string is now part of license key.

# LM-X License Manager v4.9.20 Release Notes

LM-X License Manager version 4.9.20 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.20 includes the following enhancements.

Issue #	Description
LMX-3863	The LMX_GetLicenseInfo field "szStartDate" now includes the start date of trial licenses.
LMX-3666	A new API function, <a href="#">LMX_GetSystemClockTime</a> , has been added.

## Fixes

LM-X v4.9.20 includes the following fixes.

Issue #	Description
LMX-3875	Improved VM detection for HyperV to fix a bug that was causing false positive results.

# LM-X License Manager v4.9.19 Release Notes

LM-X License Manager version 4.9.19 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.19 includes the following enhancements.

Issue #	Description
LMX-3894	Due to Apple's requirements, we are now signing our CLI tools. Customers should also sign our DLLs. (Read more at <a href="https://developer.apple.com/documentation/xcode/notarizing_macos_software_before_distribution">https://developer.apple.com/documentation/xcode/notarizing_macos_software_before_distribution</a> .)
LMX-3865	Refactored license checkout process to improve speed and license prioritization.

## Fixes

LM-X v4.9.19 includes the following fixes.

Issue #	Description
LMX-3905	Internal license parsing is now more restrictive. Unsupported tokens will fail license validation.
LMX-3897	Fixed problem with case-sensitive feature name comparison.
LMX-3882	Several problems with the ordering of shared additive license checkouts have been fixed.

# LM-X License Manager v4.9.18 Release Notes

LM-X License Manager version 4.9.18 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.9.18 includes the following fixes.

Issue #	Description
LMX-3879	Fixed system clock check issue.
LMX-3878, LMX-3862	Fixed issue with xmllicgen that resulted in errors using LM-X trial license.

# LM-X License Manager v4.9.17 Release Notes

LM-X License Manager version 4.9.17 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.17 includes the following new features and enhancements.

Issue #	Description
LMX-3828	Updated minimum version of MacOS supported by LM-X to 10.13.
LMX-3801	Removed dependencies of MinGW libraries.

## Fixes

LM-X v4.9.17 includes the following fixes.

Issue #	Description
LMX-3857	Fixed issue to ensure HOSTID_MATCH_RATE=0 disables HostID validation.
LMX-3845	Fixed issue with license sharing.
LMX-3842	Fixed issue with removing temp files.
LMX-3841	Fixed issue with uninstalling LM-X SDK.
LMX-3838	Fixed issue with invalid temporary directory.
LMX-3831, LMX-3821	Fixed issue with license checkout order.
LMX-3819	Fixed issue with LM-X Administrator API returning LMX_AUTH_ERROR when password was incorrect.
LMX-3810	Fixed issue in extended log that showed the incorrect number of licenses when LMX_OPT_ALLOW_CHECKOUT_LESS_LICENSES was turned on.
LMX-3808	Added "szUniqueID" for local licenses to LMX_GetLicenseInfo.
LMX-3267	Fixed issue with borrowing different versions of the same feature.

# LM-X License Manager v4.9.16 Release Notes

LM-X License Manager version 4.9.16 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.16 includes the following new features and enhancements.

Issue #	Description
LMX-3795	Improved internet clock checking mechanism.
LMX-3772	LM-X no longer specifies default calling convention.
LMX-3741	Introduced vendor name validation for local licenses.
LMX-3710	Added ability to create groups in license server configuration file.
LMX-1480	Added Administrator API functions for restarting and shutting down license server.

## Fixes

LM-X v4.9.16 includes the following fixes.

Issue #	Description
LMX-3798	Fixed error related to lack of embedded security configuration file.
LMX-3776	Fixed issues with file permissions on Linux.
LMX-3765	Fixed format of error message that can occur during license checkout.



# LM-X License Manager v4.9.15 Release Notes

LM-X License Manager version 4.9.15 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.15 includes the following enhancements.

Issue #	Description
LMX-3735	Removed VC++ Redistributables from LM-X installer.
LMX-2026	Set proper product description during signing of executables on Windows.
LMX-1819	Added followup instructions on last page of LM-X SDK installer.
LMX-1739	Added ability for LM-X client's extended log to print out all LM-X environment variables.
LMX-1311	Added ability for license server to print out the name of a feature that could not be returned because it was not borrowed.

## Fixes

LM-X v4.9.15 includes the following fixes.

Issue #	Description
LMX-3745	Fixed issue with internet clock checker in LM-X trial license.
LMX-3740	Fixed typo in LM-X EULA.
LMX-3712	Removed Java examples from Java wrapper.
LMX-3711	Fixed issue with LM-X SDK installer prompting twice for Administrator password on Windows.
LMX-3703	Fixed incorrectly formatted error message in LicServerClient.
LMX-3688	Fixed issue with SDK build succeeding in cases when examples failed to build.
LMX-3633	Fixed issue with license server logging data to client's extended log.
LMX-1503	Fixed issue with license queuing for HAL configuration.

# LM-X License Manager v4.9.14 Release Notes

LM-X License Manager version 4.9.14 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.9.14 includes the following fixes.

Issue #	Description
LMX-3662	Fixed issue with incorrect handling of HTTP PUT and DELETE requests.
LMX-3657	Fixed issue with logging the lack of a dongle as an error.
LMX-3650	Corrected name of callback function from retryFailure to retryFeature.
LMX-3639	Fixed issue with node-locked licenses being detected as floating licenses.
LMX-3631	Fixed issues with Google Cloud HostId.
LMX-2751	Fixed issue with restarting license server properly after adding invalid license.
LMX-2091	Fixed an issue with the wrong error code being returned when there were static reservations on a feature.
LMX-1619	Fixed issue with license checkout denial when client HostIDs were duplicated.

# LM-X License Manager v4.9.13 Release Notes

LM-X License Manager version 4.9.13 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.13 includes the following enhancements.

Issue #	Description
LMX-3667	Added size of log file on download button in LicserverClient.

## Fixes

LM-X v4.9.13 includes the following fixes.

Issue #	Description
LMX-3632	Fixed an issue with LM-X Windows installer unable to install LM-X license server as a service.
LMX-3658	Fixed an issue with LicserverClient failure to cleanup log viewer when server was changed.

# LM-X License Manager v4.9.12 Release Notes

LM-X License Manager version 4.9.12 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.12 includes the following enhancements.

Issue #	Description
LMX-3654	There is now a system clock check performed any time a date (start date, end date, maintenance start date or maintenance end date) is present in the license.
LMX-3618	The LM-X license field "VENDOR= <i>VENDOR_NAME</i> " is no longer allowed.
LMX-3605	HASP HL error messages are now visible in extended logs.
LMX-3600	LicserverClient will now update its server list for every change.

## Fixes

LM-X v4.9.12 includes the following fixes.

Issue #	Description
LMX-3652	Fixed missing build number in version.txt.
LMX-3636	Fixed NullPointerException in LicserverClient.
LMX-3630	Fixed bug with incorrect configuration for Visual Studio 2017 solution.
LMX-3623	MinGW builds now properly detect the Windows version.
LMX-3603	Fixed issue with failure to create missing directories for LicserverClient server list.
LMX-3594	Fixed bug with incorrect server priority on HAL restart.
LMX-2792	Internal license validator didn't print errors into logs.

# LM-X License Manager v4.9.11 Release Notes

LM-X License Manager version 4.9.11 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).



LM-X 4.9.11 is the last version that will support the use of the LM-X license field "*VENDOR=VENDOR\_NAME*."

LM-X 4.9.11 no longer supports Visual Studio 2013. This release supports Visual Studio 2015 and newer, including support for Visual Studio 2019.

## Enhancements

LM-X v4.9.11 includes the following enhancements.

Issue #	Description
LMX-3569	Added support for Visual Studio 2019 and discontinued support for Visual Studio 2013.

## Fixes

LM-X v4.9.11 includes the following fixes.

Issue #	Description
LMX-3624	Included .NET Framework Tools directory in Visual Studio 2019 solution (due to a bug in Visual Studio).
LMX-3617	Fixed an issue with forbidden use of LM-X license field " <i>VENDOR=VENDOR_NAME</i> ."
LMX-3606	Removed reboot prompt from LM-X SDK installer.
LMX-3595	Fixed a bug related to the LM-X Enduser Tools installer that was preventing installation of the server executable.

# LM-X License Manager v4.9.10 Release Notes

LM-X License Manager version 4.9.10 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.10 includes the following enhancements.

Issue #	Description
LMX-3552	Added new queue example in SDK.
LMX-3548	Added new return code for when the client application is trying to connect to an older, unsupported version of the license server.
LMX-3501	Added error message for mismatched License Server Client and license server versions when attempting to <a href="#">connect the License Server Client to an LM-X license server</a> .
LMX-3473	Implemented user anonymization for Pay-Per-Use database.

## Fixes

LM-X v4.9.10 includes the following fixes.

Issue #	Description
LMX-3557	Fixed an incompatibility issue when using Harddisk Hostid type with <a href="#">License Activation Center</a> .
LMX-3422	Fixed an issue with setting maximum license server version for vendor DLL.

# LM-X License Manager v4.9.9 Release Notes

LM-X License Manager version 4.9.9 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.9.9 includes the following fixes.

Issue #	Description
LMX-3553	Fixed crash caused by missing permissions to boost_interprocess folder on Windows.
LMX-3536	Removed the deprecated checkbox for "Run Visual Studio solution" from installer.

# LM-X License Manager v4.9.8 Release Notes

LM-X License Manager version 4.9.8 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.8 includes the following enhancements.

Issue #	Description
LMX-3490	Update VC Redistributables 2015 in LM-X installer.
LMX-3482	Update NTP servers list.
LMX-3453	Improve LM-X End-user Tools installer to support silent installation without server.
LMX-3355	Define calling conventions for LM-X API.

## Fixes

LM-X v4.9.8 includes the following fixes.

Issue #	Description
LMX-3506	Fix error preventing more than one LM-X End-user Tools installation per machine.
LMX-3483	Catch NullPointerException in Java client.
LMX-3476	Fix issues with building SDK using older linkers by disabling new relocations.
LMX-3474	Fix issue with printing out an incorrect license when the license server cannot read the license file.
LMX-3472	Fix issue with init script incorrectly stopping all LM-X servers running on the same machine instead of stopping only one specified server.
LMX-3465	Fix KVM false-positive detection.
LMX-3463	Fix possible out-of-range exception when gathering AWS HostID.
LMX-3459	Fix init script on Mac OS.
LMX-3458	Remove Mac exports from Linux SDK.
LMX-3432	Fix bug in Java wrapper with comparing custom HostIDs.
LMX-3427	Fix incorrect value for nActualBorrowHours when using LMX_GetLicenseInfo.
LMX-3319	Fix bugs reported by Microsoft Application Verifier.
LMX-357	Prevent xmllicgen from generating floating licenses when the SHARE directive is used with the TERMINALSERVER option.



# LM-X License Manager v4.9.7 Release Notes

LM-X License Manager version 4.9.7 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.7 includes the following enhancements.

Issue #	Description
LMX-3420	Created new SDK for MinGW x64.
LMX-3417	Updated License Activator.
LMX-3409	Removed MinGW libraries from Windows SDK. (MinGW now has its own SDK.)
LMX-3405	Updated HASP HL libraries.

## Fixes

LM-X v4.9.7 includes the following fixes.

Issue #	Description
LMX-3441	Implemented Upgrade Table in LM-X installer.
LMX-3440	Fixed several installer bugs.
LMX-3435	Fixed installer messages that contained empty placeholders.
LMX-3419	Removed old links from LM-X installer.
LMX-3416	Fixed issue with server unable to start when attempting to reserve non-existing feature.
LMX-3411	Fixed .NET documentation issue due to at character (@) not escaping.
LMX-3397	Fixed bug with gathering BIOS HostID.
LMX-3393	Fixed license server crash due to optimizations during VM detection.
LMX-3380	Many improvements on LicserverClient.
LMX-3359	Fixed issue with license server not shutting down when system clock is changed.

# LM-X License Manager v4.9.6 Release Notes

LM-X License Manager version 4.9.6 includes the fix detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.9.6 includes the following fix.

Issue #	Description
LMX-3391	Fixed a license server crash caused by KVM detection.

# LM-X License Manager v4.9.5 Release Notes

LM-X License Manager version 4.9.5 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.5 includes the following enhancements.

Issue #	Description
LMX-3348	Added support for KVM/QEMU virtualization detection.
LMX-1039	Added ability to Imxendutil that allows specification of all three servers for querying HAL.

## Fixes

LM-X v4.9.5 includes the following fixes.

Issue #	Description
LMX-3320	Fixed a problem with quotation marks in comment section of LM-X license file.
LMX-2687	Fixed an issue with Java client updating the LM-X license server configuration file with an incorrect value.

# LM-X License Manager v4.9.4 Release Notes

LM-X License Manager version 4.9.4 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.9.4 includes the following fixes.

Issue #	Description
LMX-3333	Removed message box display in LM-X to resolve an issue with the messages blocking client's API.
LMX-3321	Added missing constant (LMX_ALL_FEATURES) to .NET library.
LMX-3300	Changed the way Java callbacks are called to avoid classloader issues.

# LM-X License Manager v4.9.3 Release Notes

LM-X License Manager version 4.9.3 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.3 includes the following enhancements.

Issue #	Description
LMX-3303	Use "poll" function instead of "select" function to avoid file descriptors limit.
LMX-3284	Provide new debug version (liblmxnetd.dll) of .NET library file (liblmxnet.dll) in LM-X SDK.
LMX-3279	Check the license path for server availability before cached servers are checked.
LMX-3262	Additional validation for reservation token.

## Fixes

LM-X v4.9.3 includes the following fixes.

Issue #	Description
LMX-3298	Excluded makepp directory from LM-X SDK.
LMX-3297	Fixed problem with running 32-bit and 64-bit clients on the same machine simultaneously.
LMX-3296	Fixed inability of SDK installer to copy security configuration due to insufficient permissions.
LMX-3288	Updated NTP server list.
LMX-3287	Fixed problem with sharing client counter when umask was set (Linux).
LMX-3286	Fixed several bugs in Java Client.
LMX-3285	Fixed issue with xmllcgen not adding escaped quotation marks when a quotation mark is the first or last character in comment field.
LMX-3278	Fixed an issue with LM-X attempting to change security context even when SE Linux was disabled.

# LM-X License Manager v4.9.2 Release Notes

LM-X License Manager version 4.9.2 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.2 includes the following enhancement.

Issue #	Description
LMX-3252	Added the following new administrative functions into Java API: <ul style="list-style-type: none"><li>• LMX_Admin_Reserve</li><li>• LMX_Admin_ReserveEarlyReturn</li><li>• LMX_Admin_UploadLicense</li></ul>

## Fixes

LM-X v4.9.2 includes the following fixes.

Issue #	Description
LMX-3270	Updated NTP servers list to resolve issues with time servers.
LMX-3264	Fixed LM-X error code mapping in Java API.
LMX-3256	Fixed bug with lmxendutil -hostid not returning HARDDISK HostID information for virtual machines.
LMX-3255	Fixed problem with Nessus security scanner reporting critical error.
LMX-3253	Fixed problem starting two applications compiled with LM-X on the same machine.
LMX-3234	Resolved compilation errors that occurred while building SDK.

# LM-X License Manager v4.9.1 Release Notes

LM-X License Manager version 4.9.1 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9.1 includes the following new enhancement.

Issue #	Description
LMX-3232	Added HAL information to the LMX_LICENSE_INFO API structure.

## Fixes

LM-X v4.9.1 includes the following fixes.

Issue #	Description
LMX-3220	Fixed issue with correctly storing statistics for features on license server.
LMX-3219	Fixed license server crash that sometimes occurred when an incomplete HTTP request was made.
LMX-3210	Fixed license server crash that occurred when a license was in the queue for an extended time.
LMX-3202	Fixed problems with incorrect return codes being issued during HAL license server queuing and heartbeat checks.

# LM-X License Manager v4.9 Release Notes

LM-X License Manager version 4.9 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.9 includes the following new features and enhancements.

Issue #	Description
LMX-3204	Added new HostID for Microsoft Azure.
LMX-3203	Added new HostID for GCE (Google Compute Engine).
LMX-3199	Improved performance and fixed several bugs for LM-X Java Client.

## Fixes

LM-X v4.9 includes the following fixes.

Issue #	Description
LMX-3200	Fixed ServerFunction in Java wrapper.
LMX-3183	Created separate limit for each feature when BORROW_LIMIT_COUNT_ALL is used.
LMX-3076	Ensured that "xf-dll" folder is deleted after all client instances are closed.



# LM-X License Manager v4.8.13 Release Notes

LM-X License Manager version 4.8.13 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.13 includes the following enhancement.

Issue #	Description
LMX-3175	Improved LM-X Java wrapper (added LMX_HOSTID into FEATURE_INFO structure).
LMX-3078	Added support for Visual Studio 2017 in LM-X SDK, and removed support for Visual Studio 2012.

## Fixes

LM-X v4.8.13 includes the following fixes.

Issue #	Description
LMX-3178	Resolved handling exception in LM-X Java client.
LMX-3169	Prevented attempts to get license twice from the same license server when the server is down.
LMX-3165	Corrected display of buttons and checkbox in LM-X Enduser Tools installer.

# LM-X License Manager v4.8.12 Release Notes

LM-X License Manager version 4.8.12 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.12 includes the following enhancements.

Issue #	Description
LMX-3126	Extended LM-X Java API by adding all remaining LM-X functions, callbacks, and new, detailed Java examples to the LM-X SDK.
LMX-3124	Added the ability to create destination folders recursively.

## Fixes

LM-X v4.8.12 includes the following fixes.

Issue #	Description
LMX-3145	Removed the "Run LM-X web based client" checkbox from LM-X end user tools installer.
LMX-3138	Added support for old licenses in security verification.
LMX-3133	Removed LM-X GUI applet from the LM-X License Server.
LMX-3132	Eliminated segmentation fault occurring when a license template was broken.
LMX-3117	Fixed minor errors occurring when security check was performed while building the LM-X SDK.
LMX-3104	Fixed an issue with custom hostid compare callback not being set.
LMX-3103	Eliminated a problem with LM-X License Server shutting down when a number of custom HostIDs changed despite not being in use.
LMX-3097	Fixed a dongle-specific problem occurring when a dongle was unplugged and then plugged again during the server runtime.
LMX-3089	Made the Usage Statistics graph in the LM-X License Server Client Dashboard less prone to crashing and more responsive.
LMX-2777	Fixed an issue with lmx-serv not deleting its lock file.

# LM-X License Manager v4.8.11 Release Notes

LM-X License Manager version 4.8.11 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Important Considerations

As of this release, old liblmxvendor.dll will not work with new LM-X server and new liblmxvendor.dll will not work with old servers. Please use the same version of the license server and dll file to match.

In addition, please note that with this release we're dropping support for CentOS 5. (For more platform-specific information, see [Supported platforms](#).)

## Enhancements

LM-X v4.8.11 includes the following enhancements.

Issue #	Description
LMX-3105	Created a "verify" method of injecting security configuration into libraries.
LMX-3082	Improved LM-X install script by eliminating the need to extract installation files to an empty directory (which previously couldn't contain any configuration files, license keys, etc.).
LMX-3077	Enhanced method of delivering BIOS HostID information.
LMX-2913	Improved C# callbacks examples in the LM-X SDK.
LMX-2088	Improved LM-X SDK by injecting security keys into the LM-X Java client library (liblmxjav.so/dll), eliminating the need for vendors to build the library themselves.

## Fixes

LM-X v4.8.11 includes the following fixes.

Issue #	Description
LMX-3105	Fixed security flaw with injected security configuration.
LMX-3080	Fixed an issue which resulted in improper display of fonts in the LM-X License Server Client UI.

# LM-X License Manager v4.8.10 Release Notes

LM-X License Manager version 4.8.10 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.10 includes the following enhancements.

Issue #	Description
LMX-3060	Added the ability for the resetsystemclock tool to print out fixed filenames.

## Fixes

LM-X v4.8.10 includes the following fixes.

Issue #	Description
LMX-3063	Implemented a more detailed error message informing that LM-X allows a maximum of five installed instances of end-user tools on a single machine.
LMX-3061	Fixed an issue with installing end-user tools on a Suse Linux Enterprise Server 11 machine.
LMX-3055	Fixed an issue with LM-X license server not printing the header in lmx.serv.log.

# LM-X License Manager v4.8.9 Release Notes

LM-X License Manager version 4.8.9 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.8.9 includes the following fixes.

Issue #	Description
LMX-3051	Fixed uncaught exception while running Visual Studio debugger.

# LM-X License Manager v4.8.8 Release Notes

LM-X License Manager version 4.8.8 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Breaking Changes

With this release, we're introducing enhancements to secure store, allowing for more safety, improved speed, and greater capacity. Please note that secure store might not migrate successfully from an older version of LM-X to a newer version, which can result in data loss.

## Fixes

LM-X v4.8.8 includes the following fixes.

Issue #	Description
LMX-3043	Fixed an issue with secure store which caused the disappearing of borrows after the restart of the license server.
LMX-3035	Fixed an issue with lack of permissions when borrowing licenses on Windows.
LMX-2953	Eliminated a problem with LM-X installer not being able to read symlinks.

# LM-X License Manager v4.8.7 Release Notes

LM-X License Manager version 4.8.7 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.7 includes the following enhancements.

Issue #	Description
LMX-3015	Added the ability to call <a href="#">LMX_GetFeatureInfo</a> from the heartbeat callback functions.
LMX-3006	Enabled printing versions of dynamic reservations in Imxendutil.
LMX-3004	Added a dialog that prompts the user to reboot the machine after successful installation of the LM-X SDK.

## Fixes

LM-X v4.8.7 includes the following fixes.

Issue #	Description
LMX-3021	Fixed LMX_GetLicenseInfo crashing when discovery cache file is damaged.
LMX-3017	Enabled Java libraries to be built on Windows 10 with MinGW compiler.
LMX-3007	Fixed duplicated dynamic reservation in Imxendutil with additive licenses.

# LM-X License Manager v4.8.6 Release Notes

LM-X License Manager version 4.8.6 includes the enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.6 includes the following enhancements.

Issue #	Description
LMX-2994	Removed static reservations user limit.
LMX-2992	Increased the line length limit in the license server configuration file from 1024 characters to unlimited length.



# LM-X License Manager v4.8.5 Release Notes

LM-X License Manager version 4.8.5 includes the fixes and enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

This version changed the minimum gcc version required to build LM-X with MinGW from 4.8.3 to 5.3.0.

## Enhancements

LM-X v4.8.5 includes the following enhancements.

Issue #	Description
LMX-2959	Implemented a new error code, LMX_CANNOT_LOAD_SHARED_LIBRARY, displayed when the internal LM-X library cannot be loaded.
LMX-2940	Updated NTP (Network Time Protocol) server list with working servers.
LMX-2926	Enabled signing the library with the same .NET keypair for both 32-bit and 64-bit .NET framework.

# LM-X License Manager v4.8.4 Release Notes

LM-X License Manager version 4.8.4 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.8.4 includes the following fixes.

Issue #	Description
LMX-2910	Fixed an issue related to a missing privilege for SYSTEM user on Windows which resulted in the failing of borrow requests.
LMX-2901	Eliminated a problem causing secure store to break after the hostname has been changed.
LMX-2865	Fixed an issue for HAL servers which resulted in the master server failing to take over checking out and borrowing licenses from a slave server despite the fact that the master server was up again.
LMX-2863	Eliminated a problem related to a missing library which made it impossible to activate a license from the LM-X License Server Client.

# LM-X License Manager v4.8.3 Release Notes

LM-X License Manager version 4.8.3 includes the fixes and enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.3 includes the following enhancements.

Issue #	Description
LMX-2880	Dropped support for Linux (32 bit) ARM.

## Fixes

LM-X v4.8.3 includes the following fixes.

Issue #	Description
LMX-2893	Fixed a problem with a missing library in LM-X SDK for the Microsoft Visual C++2012.
LMX-2887	Eliminated a bug in the building system which resulted in displaying a popup message in the <a href="#">LM-X License Server Client</a> every few seconds and made it impossible to use it.
LMX-2886	Fixed LM-X build names (previously, all build names were set to "DEVELOPMENT BUILD").
LMX-2726	Enabled LM-X to reload /etc/resolv.conf file before resolving the DNS address in order to prevent the failing of gethostbyname function.

# LM-X License Manager v4.8.2 Release Notes

LM-X License Manager version 4.8.2 includes the fixes and enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.2 includes the following enhancements.

Issue #	Description
LMX-2867	Dropped support for Solaris (64 bit) x64 and Solaris (64 bit) Sparc64.
LMX-2838	Added the path to the temporary directory to the extended client log.
LMX-2831	Enabled downloading sets of data (instead of downloading all data at once) and monitoring the download progress in order to improve the speed of loading graphs in the LM-X License Server Client.
LMX-2826	Enabled sorting license names alphabetically and ordering them from highest to lowest in the LM-X License Server Client.
LMX-2825	Enabled continuous checking of custom HostIDs to eliminate the possibility of moving them to a different machine and simultaneously starting another instance of the license server on that machine.
LMX-2810	Enabled downloading network charts in UI client in the background when connecting to LM-X License Server.
LMX-2807	Added the possibility to parse licenses with quotation marks inserted into the comment section of the license file.
LMX-2805	Stopped making separate releases for Linux on Debian and RHEL 5 and started releasing them as one release now.
LMX-2794	Added the possibility to display two or more features with the same name and version on graphs in the LM-X License Server Client as different features with different counts of licenses instead of displaying them as one feature.
LMX-2727	Improved the speed of loading data for UI charts with long ranges by reducing the volume of the sent data.

## Fixes

LM-X v4.8.2 includes the following fixes.

Issue #	Description
LMX-2837	Fixed an issue which resulted in changing the license file path to the HAL license server in the configuration file to the path of the defined environment variable.
LMX-2835	Fixed an issue where pressing the Delete button in the LM-X License Server Client did not delete a license as expected.
LMX-2834	Fixed an issue for the License Server Client where pressing Enter to confirm the selected LM-X license server's URL did not cause the same action as pressing the OK button.
LMX-2832	Eliminated a problem with saving information about more than 350 borrowed licenses into secure store.
LMX-2790	Fixed an issue for the License Server Client where a user was not required to provide a password to log onto a new license server after successfully logging onto another server using the Administration tab.
LMX-2726	Enabled LM-X License Server Client to check if it has the same version as the LM-X license server that it is connecting to.

# LM-X License Manager v4.8.1 Release Notes

LM-X License Manager version 4.8.1 includes the fixes and enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.8.1 includes the following enhancement.

Issue #	Description
LMX-2784	Extended the range of valid values for HOLD and MIN_CHECKOUT directives from 1 to 86400 seconds (24 hours) to 1 to 31536000 seconds (1 year).

## Fixes

LM-X v4.8.1 includes the following fixes.

Issue #	Description
LMX-2788	Eliminated an issue with a log file not being created when installing LM-X in a directory different than /home.
LMX-2779	Fixed an issue which resulted in LM-X being started as a root user instead of a regular user indicated during the installation.
LMX-2778	Fixed an issue which resulted in a user's inability to run end-user tools because of insufficient permissions (lack of root privileges) even though a non-root user was selected during the installation.
LMX-2736	Made init_script installation unconditional and independent of whether or not a user chose to run LM-X during start-up.

# LM-X License Manager v4.8 Release Notes

LM-X License Manager version 4.8 includes the fixes and enhancements detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

Due to declining usage of NPAPI plugins, Google is carrying out its depreciation plan to remove NPAPI support from Chrome. In September 2015 NPAPI support will be permanently removed from Chrome; therefore, installed Java extensions that require NPAPI plugins will no longer be able to load such plugins. As a result of these changes, LM-X GUI applet has been converted to an independent application, LM-X License Server Client, which is located in the end-user tools directory.

With this release, we're also adding support for Visual Studio 2015 and Windows 10.

As of this release we discontinue shipping builds for the following platforms: AIX PPC64 and HP-UX (64-bit) IA64.

## Enhancements

LM-X v4.8 includes the following enhancements.

Issue #	Description
LMX-2774	Added support for Windows 10.
LMX-2773	Dropped support for AIX PPC64 and HP-UX (64-bit) IA64.
LMX-2769	Considerably improved LM-X License Manager security in regard to generating licenses.
LMX-2759	Improved the speed of loading a blacklist from secure store.
LMX-2756	Added support for Visual Studio 2015 and removed support for Visual Studio 2010.
LMX-2717	Converted LM-X GUI applet to an independent application which can be accessed using LM-X end-user tools.

## Fixes

LM-X v4.8 includes the following fix.

Issue #	Description
LMX-2749	Fixed an issue with logs not being recorded in the log file due to log file path not specified with the LOG_FILE setting in the init script for Linux OS.

# LM-X License Manager v4.7.6 Release Notes

LM-X License Manager version 4.7.6 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.7.6 includes the following fixes.

Issue #	Description
LMX-2706	Improved init loading script to eliminate a problem with reading license files from a directory different than LM-X installation dir when running LM-X license server as a service.
LMX-2705	Fixed an issue with inconsistent behavior of token-based licenses with the same token names which resulted in an inability to check out the last feature.

# LM-X License Manager v4.7.5 Release Notes

LM-X License Manager version 4.7.5 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#)

## Fixes

LM-X v4.7.5 includes the following fixes.

Issue #	Description
LMX-2696	Fixed LM-X client library crashing on Windows without indicating any errors due to permission problems.
LMX-2694	Eliminated a problem with installing LM-X on Windows with FIPS mode enabled.
LMX-2678	Changed the default LogPath setting of the LM-X license server service from command line arguments to the config file.



# LM-X License Manager v4.7.4 Release Notes

LM-X License Manager version 4.7.4 includes the fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Fixes

LM-X v4.7.4 includes the following fixes.

Issue #	Description
LMX-2679	Fixed a null pointer crash of LM-X client library under OS X.
LMX-1676	Made fixes to UNKNOWN@UNKNOWN, which was logged for the client consuming reservation that expired in the meantime.

# LM-X License Manager v4.7.3 Release Notes

LM-X License Manager version 4.7.3 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Breaking Changes

As of this release, expired licenses are no longer loaded on license servers. This change was introduced, because using expired licenses with duplicate licenses resulted in their inconsistent behavior.

## Enhancements

LM-X v4.7.3 includes the following enhancements.

Issue #	Description
LMX-2661	Added versioning for all dll files to enable distinguishing between different components.
LMX-2561	Improved logging procedure by preventing chcon errors from being displayed in non-SELinux environment.

## Fixes

LM-X v4.7.3 includes the following fixes.

Issue #	Description
LMX-2650	Fixed incorrect end dates in <a href="#">example XML license templates</a> in the LM-X SDK.
LMX-2647	Fixed an issue that caused inconsistent loading of licenses on a license server.
LMX-2638	Fixed a bug in license server callback function.
LMX-2637	Fixed <a href="#">automatic server discovery</a> failing to find servers on the network and enabled it to work with the -vendor flag specified.
LMX-2629	Fixed lmxendutil to exit with non-success error code on failure.
LMX-2343	Removed Chrome Frame for LM-X web-based UI.

# LM-X License Manager v4.7.2 Release Notes

LM-X License Manager version 4.7.2 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.7.2 includes the following enhancement.

Issue #	Description
LMX-2641	Added an <a href="#">example of using a token-based licensing model</a> to the LM-X SDK.

## Fixes

LM-X v4.7.2 includes the following fix.

Issue #	Description
LMX-2644	Fixed interactive questions in silent mode for the End-user Tools installer.

# LM-X License Manager v4.7.1 Release Notes

LM-X License Manager version 4.7.1 is a patch release to fix issues found in previous versions, as described below.

## Fixes

LM-X v4.7.1 includes the following fixes.

Issue #	Description
LMX-2624	Fixed Java document generation for JDK 1.8.
LMX-2600	Fixed LM-X hanging when heartbeat callback contains exit(int) function.
LMX-2590	Fixed handling of Dongle HostID in LM-X License Server.
LMX-2588	Fixed LM-X log that contained erroneous information about additional, non-existent denied users.
LMX-2582	Fixed LM-X SDK building with strict options with MinGW.

# LM-X License Manager v4.7 Release Notes

LM-X License Manager version 4.7 includes the enhancements and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see the article, "[Customer-driven development](#)."

## Important Changes

As of this release, you no longer need to use a separate mingw32 installer to compile the LM-X SDK using MinGW, because it has been merged with regular Windows distribution that originally contained only Visual Studio libraries. However, if you want to use both MinGW and Visual Studio libraries, you should compile a single LM-X SDK twice: first using `nmake` to initialize Visual Studio libraries, and then with `mingw32-make` to initialize MinGW libraries, as described in [Compile the LM-X SDK on Windows](#).

Furthermore, this release introduces new libraries for 64-bit MinGW platforms. Both 32-bit and 64-bit libraries were produced using updated versions of version 4.8.3 of `gcc`.

## Enhancements

LM-X v4.7 includes the following enhancements.

Issue #	Description
LMX-2458	Added support for Microsoft Visual Studio 2013.
LMX-2450	Unified Visual Studio and MinGW SDKs for Windows. Introduced libraries for 64bit MinGW.

## Fixes

LM-X v4.7 includes the following fix.

Issue #	Description
LMX-2589	Fixed a problem with <code>lmx-serv</code> crashing due to permission problems on Windows.

# LM-X License Manager v4.6.5 Release Notes

LM-X License Manager version 4.6.5 includes the enhancement and fixes detailed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.6.5 includes the following enhancement.

Issue #	Description
LMX-2028	Enhanced text messages with more details for permission errors

## Fixes

LM-X v4.6.5 includes the following fixes.

Issue #	Description
LMX-2539	Fixed an issue causing Boost.interprocess to fail with "Access denied" error on 64-bit applications, and not on 32-bit ones.
LMX-2553	Fixed a client deadlock when LMX_Checkout was called too often.

# LM-X License Manager v4.6.4 Release Notes

LM-X License Manager version 4.6.4 is a patch release to fix issues found in previous versions, as described below.

## Fixes

LM-X v4.6.4 includes the following fixes.

Issue #	Description
LMX-2553	Fixed an issue where removing client temporary files by dynamically loaded DLL linked with Imxclient failed when the application itself was not linked with Imxclient.
LMX-2535	Provided a more detailed error message displayed when using the dlopen() function causes the LMX_Init() function to return LMX_UNKNOWN_ERROR.

# LM-X License Manager v4.6.3 Release Notes

LM-X License Manager version 4.6.3 is a patch release to fix issues found in previous versions, as described below.

## Fixes

LM-X v4.6.3 includes the following fixes.

Issue #	Description
LMX-2525	Filtered out a few GCC switches to avoid compilation problems with older GCCs.
LMX-2512	Fixed pthread_mutex_unlock() function called on freed mutex triggered by LMX_UnregisterDll.
LMX-2510	Fixed a problem with HAL server paths not loading properly from the configuration file.



# LM-X License Manager v4.6.2 Release Notes

LM-X License Manager version 4.6.2 is a patch release to fix issues found in previous versions, as described below.

## Enhancements

LM-X v4.6.2 includes the following enhancements.

Issue #	Description
LMX-2483	Replaced hard-coded tmp directory with the system call.

## Fixes

LM-X v4.6.2 includes the following fixes.

Issue #	Description
LMX-2500	Fixed a problem with an installer not working on Solaris.
LMX-2487	Fixed a problem where passing a NULL value to LMX_OPT_AUTOMATIC_HEARTBEAT_INTERVAL resulted in truncating LMX_OPT_LICENSE_STRING.
LMX-2463	Fixed an issue with clean target removing wrong files under Linux.
LMX-2299	Fixed inconsistent command-line flags for the following LM-X tools: lmxdev, xmllicgen, lmxresetsystemclock, lmxresettrial, lmxendutil. All the completed commands have a new flag.
LMX-2289	Fixed a problem with lmxendutil reporting 0 instead of error when forwarding the stat response from master to slave failed for a HAL license server.

# LM-X License Manager v4.6.1 Release Notes

LM-X License Manager version 4.6.1 is a patch release to fix issues found in previous versions, as described below.

As of this release:

- We are shipping builds for FreeBSD version 10.0.
- We have discontinued shipping builds for the following 32-bit platforms: AIX PPC32, Solaris x86, HP-UX IA64 ILP32, and Solaris Sparc.

## Fixes

LM-X v4.6.1 includes the following fixes.

Issue #	Description
LMX-2452	Fixed a problem with subsequent checkouts partially ignoring static reservations.
LMX-2451	Handled a case where setting up IPv4 sockets failed on Windows XP.
LMX-2449	Eliminated LM-X.NET client library crashes on Windows Server 2008 R2/Amazon.
LMX-2448	Fixed invalid paths in the LM-X SDK makefiles for Windows.
LMX-2443	Fixed a problem with collecting Bios HostID information from the system.

# LM-X License Manager v4.6 Release Notes

LM-X License Manager version 4.6 includes the enhancements and fixes detailed below. Most of these improvements were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Breaking Changes

As of this release, `LMX_AtExit` function is removed to eliminate its inconsistent behavior across platforms

Furthermore, beginning with this release, ARM executables and libraries are no longer built using softfp floating point ABI. ARM executables are now built using a hard floating point ABI.

## Enhancements

LM-X v4.6 includes the following enhancements.

Issue #	Description
LMX-2424	Improved UI in terms of its visuals and usability.
LMX-2383	Enabled checkout of all available licenses if the license server has fewer licenses than requested.
LMX-2309	Added configuration option to ignore local borrow restrictions for selected users.

## Fixes

LM-X v4.6 includes the following fixes.

Issue #	Description
LMX-2429	Fixed "make clean" to fully purge the files from the previous SDK compilation.
LMX-2427	Eliminated LM-X client crashes when starting multiple clients at the same time.
LMX-2256	Fixed license server failure to start immediately after installation on Windows platforms despite the appropriate option to start the server having been selected during installation.

# LM-X License Manager v4.5.8 Release Notes

LM-X License Manager version 4.5.8 includes the enhancements and fixes detailed below. Most of these improvements were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.5.8 includes the following enhancements.

Issue #	Description
LMX-2375	Improved data loading performance for UI chart by enabling loading older data on demand.
LMX-2359	Enabled optional installation of LM-X Server on Unix.

## Fixes

LM-X v4.5.8 includes the following fixes.

Issue #	Description
LMX-2285	Removed inconsistency in handling default values in LMX_SetOption.
LMX-2408	Eliminated trial version crashing whenever version number increased.
LMX-2406	Fixed interprocess resource locking on Windows.

# LM-X License Manager v4.5.7 Release Notes

LM-X License Manager version 4.5.7 includes the enhancement detailed below. This change was made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#)

As of this version, a build dedicated for CentOS 5 is shipped with each release (with the installer filename suffix "\_rhel5" text) to improve compatibility with RHEL 5, as described in [Supported platforms](#).

## Enhancements

LM-X v4.5.7 includes the following enhancement.

Issue #	Description
LMX-2369	Enabled the resources allocated by LMX_Init to be automatically freed upon application exit by eliminating the need to call LMX_Free function.

# LM-X License Manager v4.5.6 Release Notes

LM-X License Manager version 4.5.6 includes the enhancements and fixes detailed below. Most of these improvements were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.5.6 includes the following enhancements.

Issue #	Description
LMX-2349	Improved cleaning of old LM-X libraries from the temp directory on Windows.
LMX-2319	Improved compiler detection support for Windows SDK.
LMX-2311	Adjusted license server UI to integrate with the latest Java Runtime (JRE 7u45).

## Fixes

LM-X v4.5.6 includes the following fixes.

Issue #	Description
LMX-2317	Enabled printing of CHECKOUT messages for TOKEN features in the log file.

# LM-X License Manager v4.5.5 Release Notes

LM-X License Manager version 4.5.5 includes the enhancements and fixes detailed below. Most of these improvements were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.5.5 includes the following enhancements.

Issue #	Description
LMX-2266	Improved xmllicgen to generate licenses for multiple template files.
LMX-2239	Introduced LMX_AtExit API call to ensure calling LM-X-API methods within atexit works properly.
LMX-2163	Improved interprocess resource locking.
LMX-2130	Improved reading of Unicode files.
LMX-2025	Improved license server log messages for removing users using the lmxendutil function.

## Fixes

LM-X v4.5.5 includes the following fixes.

Issue #	Description
LMX-2249	Fixed a case where LM-X client library crashed on .NET platform.
LMX-2235	Improved license server performance on Mac OS X.
LMX-2232	Fixed multithreading issues on AIX that caused license server to crash.
LMX-2223	Fixed an incorrect error message for some HostID requests on Linux.
LMX-2213	Fixed an issue with borrow request for HAL license server.
LMX-2072	Fixed an issue where running "nmake clean" in Windows SDK makefile failed in a platform folder.

# LM-X License Manager v4.5.4 Release Notes

LM-X License Manager version 4.5.4 includes the enhancement and fix listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.5.4 includes the following enhancement.

Issue #	Description
LMX-2236	Update HASP libraries to version 7.0 on Linux and Windows.

## Fixes

LM-X v4.5.4 includes the following fix.

Issue #	Description
LMX-2240	Incorrect validation of user list in Imxendutil.



# LM-X License Manager v4.5.3 Release Notes

LM-X License Manager version 4.5.3 includes the enhancement and fix listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.5.3 includes the following enhancement.

Issue #	Description
LMX-2228	Add ability to enable internet clock check for LM-X vendors.

## Fixes

LM-X v4.5.3 includes the following fix.

Issue #	Description
LMX-2226	The LmxResetSystemClock tool's clock verification logic was reversed, resulting in the tool working only when the clock was incorrect.

# LM-X License Manager v4.5.2 Release Notes

LM-X License Manager version 4.5.2 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#). Note that as of LM-X v4.5.2 we will be shipping:

- The Linux version of LM-X using Debian 5.0 instead of 4.0 as for previous releases
- A single MacOSX build for all versions (10.6, 10.7 and 10.8)

## Breaking Changes

Beginning with this release, the `resetsystemclock` tool now *requires* an internet connection to fix any time problems reported by LM-X.

## Enhancements

LM-X v4.5.2 includes the following enhancements.

Issue #	Description
LMX-2191	Add ability to perform silent installation for Linux. (See Performing a silent installation in the <i>LM-X Developers Manual</i> .)
LMX-2177	Support Internet time verification for <code>lmxresetsystemclock</code> .
LMX-2143	New settings for system clock check to allow time verification using a network time server. The system clock check can now be set to LOCAL, INTERNET or DISABLED.

## Fixes

LM-X v4.5.2 includes the following fixes.

Issue #	Description
LMX-2203	Remove support for Solaris x86.
LMX-2190	Windows makefile did not include JAVA_HOME warning.
LMX-2175	SDK installer fails under Mac OSX.

# LM-X License Manager v4.5.1 Release Notes

LM-X License Manager version 4.5.1 is a patch release to resolve issue LMX-2173, which required updating the build system to make executables compiled by Visual Studio 2012 work on Windows XP and Windows 2003.

## Compatibility Notes

As of LM-X v4.5.1, we are shipping builds for FreeBSD version 9.1 (previously 7.2).

# LM-X License Manager v4.5 Release Notes

LM-X License Manager version 4.5 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Compatibility Notes

As of LM-X v4.5:

- We are now shipping only two versions of the Windows SDKs for Visual Studio: one for x86 (32-bit) and one for x64 (64-bit). These SDKs contain client libraries for all supported MSVC versions. Previously, there was a separate SDK for each MSVC version.
- We have discontinued shipping builds for FreeBSD x86 and Linux IA64, due to a lack of demand for these platforms.

## Enhancements

LM-X v4.5 includes the following enhancements.

Issue #	Description
LMX-2118	When calling LMX_Checkout on an expired trial license, return LMX_FEATURE_EXPIRED.
LMX-2112	Enhance license server logging with detected clock tampering warnings.
LMX-2101	Allow heartbeat re-checkout to succeed if the feature that is being renewed has a version equal to or less than the version of the feature that is currently being served by the server.
LMX-2100	If two or more duplicate license files are loaded, LM-X will use only the first license file read and will ignore the other duplicate license file(s).
LMX-2073	Allow the license server to be run using a different init system without requiring export of LD_LIBRARY_PATH environment variables.
LMX-2040	Clarify borrow return error message.
LMX-2017	Reduce the number of SDK distributions for Windows to one for x64 (64-bit) and one for x86 (32-bit).

## Fixes

LM-X v4.5 includes the following fix.

Issue #	Description
LMX-2119	SystemClockCheck results are unreliable in consecutive runs.

# LM-X License Manager v4.4.9 Release Notes

LM-X License Manager version 4.4.9 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#)

## Enhancements

LM-X v4.4.9 includes the following enhancements.

Issue #	Description
LMX-2106	Improve performance for concurrent handling of LMX_GetLicenseInfo/lmxendutil -licstat calls.
LMX-2105	Improve performance of HAL peer validation.
LMX-2090	New, simplified XML template setting for specifying multiple HostIDs.
LMX-1903	Move LM-X Developers Manual online.

## Fixes

LM-X v4.4.9 includes the following fixes.

Issue #	Description
LMX-2111	DENY_{HOST,USER,*}_ALL must not deny white-listed hosts, users, etc.
LMX-2086	After the connection to the server is lost, heartbeats does not invoke the RetryFeature callback when server is back up.
LMX-2084	When using dynamic license reservations, the license server does not recognize the reservation limit for the first license pool configured.
LMX-2065	After replacing a feature in a floating license, the license server still reads the old feature instead of the new feature.
LMX-2046	Only dynamic reservation limit should be written to Secure Store.
LMX-2045	Borrowed users not displayed correctly in web-based UI.
LMX-2041	The log erroneously reports 0 licenses queued.
LMX-2033	When a license is removed using the web-based UI, the license appears to be removed but is not actually removed until the server is restarted.
LMX-1981	Used license count does not match the count in the user list.

# LM-X License Manager v4.4.8 Release Notes

LM-X License Manager version 4.4.8 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.8 includes the following enhancements.

Issue #	Description
LMX-2027	Do not log disabled HostIDs for extended log.
LMX-2024	Do not log borrow and grace license actions when they're disabled.

## Fixes

LM-X v4.4.8 includes the following fixes.

Issue #	Description
LMX-2062	Manual time change detection does not work properly.
LMX-2045	Borrowed users not displayed correctly in UI.
LMX-2043	Secure Store lock file does not work with Unicode-encoded usernames.
LMX-2042	Windows installer should require v4 .NET runtime.
LMX-2038	Secure Store does not work properly because usernames are case-sensitive.
LMX-2018	HAL servers may crash or be unstable when under a heavy load of LMX_GetLicenseInfo calls.
LMX-1992	Client log does not show proper expiration time.
LMX-1952	Heartbeat thread may hang on join operation during client library unload.
LMX-1882	Statistics graphs in GUI show data in a misleading way.
LMX-1607	Duplicated error messages in LMX_GetErrorMessage().
LMX-1240	Configuration entries were not validated properly, which sometimes resulted in a server crash.

## LM-X License Manager v4.4.7 Release Notes

LM-X License Manager version 4.4.7 includes the fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

Issue #	Description
LMX-2005	Virtual machine detection for Windows 8 and Surface Pro gives false-positive results.
LMX-2020	License server UI not working after Java 7u11 b21 update.
LMX-2029	End-user Tools installer will not install multiple instances on Windows.

# LM-X License Manager v4.4.6 Release Notes

LM-X License Manager version 4.4.6 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Breaking Changes

In this release, we changed the handling of HostIDs for floating licenses. The customer feedback on this functionality was significant enough to warrant breaking existing code that depended on the previous behavior.

Prior to this version, all possible HostID types were computed for every floating license checkout and sent to the license server. This resulted in relatively poor performance compared to LM-X v3, where such HostID types were non-existent.

If you use a client-side HostID for your floating licenses and want to retain the behavior prior to this release, you may want to add the following lines to the LM-X client library configuration section of your code:

```
LMX_SetOption(lmxHandle, LMX_OPT_HOSTID_ENABLED, (LMX_OPTION) LMX_HOSTID_ALL);
LMX_SetOption(lmxHandle, LMX_OPT_CLIENT_HOSTIDS_TO_SERVER, (LMX_OPTION) 1);
```

For more details please see the following sections in the LM-X Developers Manual:

- [2.3 Optimizing license checkout speed](#)
- [Under 4.2.4 LMX\\_SetOption: LMX\\_OPT\\_HOSTID\\_ENABLED and LMX\\_OPT\\_HOSTID\\_DISABLED option\) and LMX\\_OPT\\_CLIENT\\_HOSTIDS\\_TO\\_SERVER options](#)

## Enhancements

LM-X v4.4.6 includes the following enhancements.

Issue #	Description
LMX-1963	Create option to disable sending HostID types over network.
LMX-1962	Improve locking of client library to avoid security flaws.
LMX-1830	Reduce load time for web UI.

## Fixes

LM-X v4.4.6 includes the following fixes.

Issue #	Description
LMX-2008	Visual Studio solution files for SDK are broken.
LMX-1959	Random crashes while running clients in parallel.
LMX-1944	Error logging creates broken text messages.
LMX-1943	Bug with empty HostIDs.
LMX-1939	Slow initial checkout under Windows.
LMX-1936	LMX Client library crashes due to improper error handling/argument validation.
LMX-1921	Single lock is per user instead of per machine under Windows.
LMX-1894	Issue with uploading a license to the server.
LMX-1827	Dongle and extended log causes license server to crash.



# LM-X License Manager v4.4.5 Release Notes

LM-X License Manager version 4.4.5 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.5 includes the following enhancements.

Issue #	Description
LMX-1929	The SDK installer now includes a link to the online quickstart guide, <i>Getting Started with LM-X License Manager</i> .
LMX-1915	Clarified instructions for obtaining liblmxvendor.dll during End-user Tools installation.
LMX-1912	"Clean" target was added to SDK makefiles, also available in Visual Studio's solutions.
LMX-1898	The End-user Tools installer now has a proper default path to liblmxvendor.dll file when there was a previous installation of the LM-X SDK.
LMX-1895	Errors and warnings that occur during LM-X client library load are now obtainable by using LMX_GetErrorMessage.

## Fixes

LM-X v4.4.5 includes the following fixes.

Issue #	Description
LMX-1926	Restarting the license server causes sockets to be closed twice.
LMX-1899	Crash caused by using atexit or static object destructors.
LMX-1896	Usernames containing printf's special characters may cause a server crash or misprints in a log file.
LMX-1889	Calling LMX_Admin_UploadLicense causes License server to crash.
LMX-1876	Installer on *nix platforms not accepting uppercase letters in response to Y/N questions. Both lowercase and uppercase letters are now accepted.
LMX-1869	Race condition on loading LM-X client library.
LMX-1868	HAL crashes when no configuration file is specified.

# LM-X License Manager v4.4.4 Release Notes

LM-X License Manager version 4.4.4 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.4 includes the following enhancements.

Issue #	Description
LMX-1842	Improved performance for acquiring Amazon HostIDs.
LMX-1805	Extended logging for Pay-Per-Use database.

## Fixes

LM-X v4.4.4 includes the following fixes.

Issue #	Description
LMX-1860	Pointer double-free on Linux x64.
LMX-1853	Web server is vulnerable to XSS attack.
LMX-1816	License server deadlocks on concurrent LMX_GetLicenseInfo calls.
LMX-1793	IPv6 address ::1 is erroneously reported as a HostID.
LMX-1742	M-X crashes on CentOS 5.8.
LMX-1729	Cannot reserve licenses for both usernames and hostnames.

# LM-X License Manager v4.4.3 Release Notes

LM-X License Manager version 4.4.3 includes the enhancements and fixes listed below. The changes in this release were made primarily in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.3 includes the following enhancements and changes.

Issue #	Description
LMX-1794	Add ability for vendor to define maximum server version number.
LMX-1636	Do not display IPv6 link-local addresses for HostIDs.
LMX-1624	Add and enhance installation programs, including: Introduce installation program for end-user tools for Unix and Windows; introduce SDK installer for Unix; and improve the existing SDK installer for Windows.
LMX-1618	The Imxendutil and Imxconfigtool end-user tools no longer require embedding the security configuration file. These tools are now generic.

## Fixes

LM-X v4.4.3 includes the following fixes.

Issue #	Description
LMX-1817	LM-X crashes on CentOS 5.8.
LMX-1743	Fixed display of USB HDD HostIDs.

# LM-X License Manager v4.4.2 Release Notes

LM-X License Manager version 4.4.2 includes the enhancements and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.2 includes the following enhancements and changes.

Issue #	Description
LMX-1741	Add support for Mac OS X 10.8.
LMX-1719	Implement path versioning based on secure store version.
LMX-1568	Add new HostID type LMX_HOSTID_AWS_INSTANCE_ID.

## Fixes

LM-X v4.4.2 includes the following fixes.

Issue #	Description
LMX-1642	Fix wrong data type in Java wrapper.
LMX-1677	LMX_Admin_Reserve fails with "Unknown error" when LMX_ALL_LICENSES is used.
LMX-1713	When borrowed licenses expire for multiple clients, LM-X returns the license for only the first client.
LMX-1728	Fix wrong license path order for HAL.
LMX-1734	Secure Store can be accidentally corrupted.

# LM-X License Manager v4.4.1 Release Notes

LM-X License Manager version **4.4.1** includes the enhancements and fixes listed below. The improvements in this release were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

## Enhancements

LM-X v4.4.1 includes the following enhancements and changes.

Issue #	Description
LMX-1683	LM-X no longer requires manually specifying the library search path.
LMX-1600	Visual Studio 2005 is no longer supported.

## Fixes

LM-X v4.4.1 includes the following fixes.

Issue #	Description
LMX-1711	Bug in parsing options field for LM-X internal license.
LMX-1710	Bug in displaying details for LM-X internal license.

# LM-X License Manager v4.4 Release Notes

LM-X **License Manager** version 4.4 includes several enhancements and fixes. The improvements made in this **release**, listed below, were made in response to customer feedback. For more information about how we incorporate customer feedback into our development process, see [Customer-driven development](#).

This release introduces some breaking changes. Please see SDK changes for LM-X v4.4 and newer for details.

## Enhancements

LM-X v4.4 includes the following enhancements and changes.

Issue #	Description
LMX-1625	Change automatic heartbeats to attempt checkout from different servers only on last heartbeat.
LMX-1616	Remove support for Mac OS X 10.5.
LMX-1611	Call HEARTBEAT_CHECKOUT_SUCCESS_FUNCTION or HEARTBEAT_CHECKOUT_FAILURE_FUNCTION callback after every HEARTBEAT_RETRY_FEATURE_FUNCTION callback.
LMX-1604	Provide a new configuration setting for specifying an update interval for Pay Per Use database records.
LMX-1286	Create a separate library (liblmxvendor.dll for Windows or liblmxvendor.so for Unix) to hold vendor-specific code, and remove vendor-specific information from the license server (lmx-serv) executable to make it generic.
LMX-1582	Add support for Visual Studio 2012.
LMX-1575	Ensure that LM-X works properly with Windows 8.
LMX-1555	Disable callbacks for internal LM-X calls.
LMX-1546	Implement self-licensing for LM-X by activating licenses using License Activation Center.
LMX-1420	Enhance lmxendutil to enable it to display recent denials statistics.

## Fixes

LM-X v4.4 includes the following fixes.

Issue #	Description
LMX-1613	LM-X will not exit even if LmxServerStartup function returns LMX_UNKNOWN_ERROR.
LMX-1609	After connection to the server was lost, the HEARTBEAT_CHECKOUT_FAILURE_FUNCTION callback was called even if there was no HEARTBEAT_RETRY_FEATURE_FUNCTION called.
LMX-1602	lmxendutil -licstat -network results are poorly formatted.
LMX-1597	CUSTOM_USERNAME and HOSTNAME do not work during checkout.
LMX-1589	Match rate is not working properly in some cases.
LMX-1571	License Server GUI does not load newest version after upload.
LMX-1553	Refreshing a borrowed license does not work properly on the client side.

# Troubleshooting developer problems

If you have any problems installing or running LM-X, we may ask you to [enable extended logging](#), and send us the [log files](#) from the LM-X log directory.

Troubleshooting for specific issues is covered in the following sections:

[Compilation issues](#)

[Debugging issues](#)

[Enabling extended logging](#)

[HASP issues](#)

[High CPU utilization issues](#)

[Installation issues](#)

[Interprocess issues](#)

[Licensing issues](#)

[Runtime issues](#)

[Virtual machine issues](#)

# Compilation issues

## "LM-X SUBSCRIPTION VERSION: Invalid license key" error

When you [compile the LM-X SDK](#), you can get the following error:

```
LM-X SUBSCRIPTION VERSION: Invalid license key
```

When this error occurs, please [send us](#) your license file from the config directory so that we can check internal settings of your license.



# Debugging issues

## "0x000006BA: RPC server unavailable" message

When debugging an application in which LM-X is integrated, you may see messages similar to the following:

```
0x000006BA: The RPC server is unavailable  
0xC000001D: Illegal Instruction  
0xC0000096: Privileged instruction
```

These messages are part of the virtual machine detection that LM-X performs, and do not indicate any errors.

# Enabling extended logging

If you experience problems with LM-X, we may ask you to enable extended logging, as described below.

## Enabling extended logging for client-side logs

*To enable client-side extended logging in Windows:*

1. Set the environment variable [LMX\\_EXTENDEDLOG](#); for example:  

```
$ set LMX_EXTENDEDLOG=C:\Users\john\lmx-extended.txt  
$ my-application.exe
```
2. Restart your licensed software.

*To enable client-side extended logging in Linux:*

1. Set the environment variable [LMX\\_EXTENDEDLOG](#); for example:  

```
$ export LMX_EXTENDEDLOG="/var/log/lmx-extended.log"  
$ ./my-application
```
2. Restart your licensed software.

## Enabling extended logging for license server

*To enable extended logging for the license server in Windows or Linux, either:*

- [Enable extended logging](#) in the [license server configuration file](#).
- or
- [Set the extended logging in the web-based UI](#).

For more information about license server logging, see [License server log file](#).

# HASP issues

## Problems with dongle visibility

Due to Gemalto specifications, HASP dongles may not be visible for the LM-X library under the following circumstances:

- when running with terminal servers, including remote desktop connections on Windows OS.
- on some Unix machines when using an SSH connection.
- LM-X for Linux x86 does not recognize dongles running on Linux x64 machines.

If you are planning to use the [LM-X license server](#), make sure it's running as a service to ensure visibility of the dongles.

# Installation issues

## Failure to run nmake on Windows

If nmake fails under Windows, the compiler environment for Visual Studio may not be set up properly.

To set it on the command line, you can run vcvars32.bat, typically located in C:\Program Files\Microsoft Visual Studio *version*\VC\bin. Be sure to use the same console to execute nmake that you used to run the batch file.

## C#(csc.exe) errors with .NET framework

If you get C# (csc.exe) errors, your compiler executable file csc.exe is not recognized as an internal or external command.

To resolve this issue, make sure the .NET framework location is included in your PATH variable (for example, the location for .NET framework is: C:\Windows\Microsoft.NET\Framework\v4.0.30319).

## A "bad key" failure

If you do not reuse your existing [LM-X security configuration file](#) for the lifetime of your application, you may get "bad key" failures between license generators, license server and licensed applications.

This issue occurs mainly when compiling LM-X SDK on multiple platforms. To resolve this problem, be sure to reuse the LM-X security configuration file.

## Problem with JAVA\_HOME environment variable

If your JAVA\_HOME environment variable is set to the JRE rather than the JDK, or it is set to a JDK that is not officially provided by Oracle, your LM-X SDK will be built without Java programming language support. In such cases, when you build the LM-X SDK, you will see an error similar to the following:

```
In file included from lmxjava.c:19:0:
lmxjava.h:2:17: fatal error: jni.h: No such file or directory
#include <jni.h>
^
compilation terminated.
makefile:28: recipe for target 'all' failed
```

To fix this problem, make sure your environment variable is set to the path of version 1.6 or newer of the JDK.

# Interprocess issues

## "Access denied" error during license checkout

Some [LM-X License Server](#) features that you and/or your end user may optionally enable involve storing client data. For example, [borrowing](#) or checking out a floating license with enabled [automatic server discovery](#) require OS-level synchronization between different processes.

If initiating one of the above actions fails under Windows, LM-X may return the error code [LMX\\_SYSTEM\\_INTERPROCESS](#) indicating a resource locking error. Furthermore, if you want to borrow a license by setting the environment variable [LMX\\_BORROW](#), you may see the following error:

```
Unable to checkout:  
LM-X Error: (Internal: 23 Context: 23)  
Access is denied.
```

In most cases, removing `C:\ProgramData\boost_interprocess` should solve this problem.

# Licensing issues

## Problem running more than one license server on a single machine

Each license server will host licenses for a specific vendor. You cannot start multiple instances of the same license server on a single machine. When you make an attempt to start a second instance of a license server hosting the same vendor on the same machine, you may see the following error in the LM-X License Server log file:

```
"FAIL: Unable to create lockfile ...."
```

To fix this problem, you should ask your application vendor to give you a correct version of the license server for the vendor's licenses. If you have two license servers from different vendors, the servers can coexist if you ensure that each server runs on a unique TCP port.

## "Software not allowed to run on terminal server client" error

If the [SHARE = TERMINALSERVER](#) directive has not been included in the license file, you may see the following error when attempting to use an LM-X-protected license on a terminal server:

```
FATAL ERROR - Exploration License: LM-X Error:
```

```
[LOCAL] C:\MSRDS2.0_Cons\bin\karto.lic - Software not allowed to run on a terminal server client
```

Setting the SHARE directive to TERMINALSERVER for a local (node-locked) license specifies that the licensed application will work for remote terminal server clients. When TERMINALSERVER is not specified, remote clients will be blocked, thereby preventing a local license from being shared for multiple users via a terminal server.

**Note:**TERMINALSERVER setting applies only to Windows and is ignored by Unix.

# Runtime issues

## First-time exceptions in Visual Studio

When the debug mode is enabled, the debugger gets the first chance to see all exceptions before the program does. If the debugger allows the program execution to continue, the program will see the exception as usual. If the program does not handle the exception, the debugger gets a second chance to see the exception, normally causing the program to crash if the debugger were not present.

When running an application with a debug mode enabled in Visual Studio, you may see the following message in the output window:

```
"First-chance exception at 0x76c1b727 in yourapp.exe: some_exception_type at memory location 0x0033411c"
```

The above message doesn't indicate that LM-X didn't handle an exception, but only notifies you that the exception occurred. If LM-X handles the exception gracefully, no second chance exception will appear.

Please note that LM-X guarantees exception safety and doesn't leave any exceptions unhandled, even if they occur internally.

## Where does LM-X store its libraries?

Every LM-X protected application checks for the existence of dll files in the following order and uses the first path found:

*On Linux:*

1. The path specified by the [TMPDIR](#) environment variable.
2. The /tmp path.

*On Windows:*

1. The path specified by the TMP environment variable.
2. The path specified by the TEMP environment variable.
3. The path specified by the USERPROFILE environment variable.
4. The Windows directory.

## USERPROFILE locks the licensing process

When re-defining the USERPROFILE environment variable on Windows, the license checkout may lock the licensing process: in such situations closing the application abruptly seems to be the only solution. One workaround to this problem is to restore the value *before* using any LM-X function and then set it again after making LM-X API calls to prevent this issue from occurring.

# Virtual machine issues

## Configuring a license to run on virtual machines when using an SSH connection

Using an SSH connection does not require that you configure your licenses to allow your application to run in a [virtual machine environment](#). (That is, by setting SHARE = VIRTUAL in your feature description.) SSH is simply a communication protocol, while virtual machines include VMware, Microsoft Virtual Server, etc.



# High CPU utilization issues

## High CPU utilization when running LM-X License Server

When you run [LM-X License Server](#), you may experience high CPU utilization (such as 1 hour's worth of 100% CPU time gone to running the license server per day) despite the fact that there is very little activity going on.

This problem occurs because most of the used CPU time is consumed by the network libraries trying to serve licenses to multiple [clients](#) on the network in an inefficient manner. LM-X was originally designed to support the whole set of platforms and therefore the network libraries used are made to be as reusable as possible. Unfortunately this means that these libraries are not making use of the best platform-specific techniques available to gain the best performance.

In the future we will consider rewriting this into more modern technology if the demand is high enough.

However, given cheap computing costs nowadays, this issue should not pose a big problem in most cases.

Learn more about [running the license server as a daemon in the background](#).

# Reporting a bug in LM-X

If you experience problems with [LM-X License Server](#), we may ask you to [enable extended logging for the license sever](#) and send us the [log files](#) from the LM-X log directory for further investigation.

If your request for assistance concerns the client library, you may have to enable [client-side extended logging](#).

To help us assist you most quickly, when reporting a bug in LM-X:

1. Create a reproduction of the bug that includes a small example demonstrating the problem. Include instructions on how to use your reproduction.
2. Send your own application code if it is compilable.
3. Compress and send all files by email to X-Formation technical support.  
**Note:** In case of large amount of data, the files should be uploaded onto [the X-Formation Upload Portal](#). Each affected client will be given unique credentials to the FTP server.
4. Depending on the type of problem you are reporting, additional information may be helpful:
  - *Application-specific issues:* Assistance is not covered under normal LM-X support and must be treated as hourly consultant work. If this is the case, we will inform you so that you can either request our consultant services or do the debugging yourself.
5. You can request our consultant services to obtain a complete implementation of LM-X within your software, where we will assure that the implementation works as expected.

# Supported platforms

Some information on this page refers to LM-X License Manager v5.2 or newer, which added support for Linux ARM. If you are using an earlier version, please refer to the [documentation for releases prior to v4.8.11](#).

The following table lists the platforms currently supported by LM-X License Manager, as well as platform-specific information and limitations.

For Licensed Platform	Download Filename	Requirements/Limitations
FreeBSD (64 bit)	lmx-sdk-version#_freebsd_x64.sh	
Linux x86 (32 bit)	lmx-sdk-version#_linux_x86.sh	<ul style="list-style-type: none"> <li>You cannot build the 32-bit LM-X SDK on Red Hat Enterprise Linux 6 x64; however, you can run LM-X tools, including the license server, on Red Hat Enterprise Linux 6 x64.</li> <li>The x86 and x64 builds are compiled on CentOS 6, which uses glibc v2.12 to improve compatibility with RHEL 6.</li> <li>For SELinux, an LM-X-enabled application must have <code>textrel_shlib_t</code> to run. The LM-X client tries to add this library during runtime, but if the current user does not have permissions to modify the SELinux configuration, an administrator must issue the following command to add <code>textrel_shlib_t</code>: <pre>semanage fcontext -a -t textrel_shlib_t '/tmp/xf-dll/xf-.*\..tmp'</pre> </li> </ul>
Linux x64 (64 bit)	lmx-sdk-version#_linux_x64.sh	
Linux ARM (64 bit)	lmx-sdk-version#_linux_arm.sh	
Mac OS X (Universal) for 10.14+	lmx-sdk-version#_darwin_universal.sh	
Windows (32 bit) x86 compiler	lmx-sdk-version#_win32_x86.msi	<p>All files have been digitally signed, except:</p> <ul style="list-style-type: none"> <li>.NET, Java and dll redistributables</li> <li>example code</li> </ul> <p>These files can be signed with your own digital certificate.</p> <p>The supported Windows versions are 8.1, 10 and 11.</p>
Windows (64 bit) x64 compiler	lmx-sdk-version#_win64_x64.msi	
MinGW32 (with GCC 8.1.0)	lmx-sdk-version#_mingw32_x86.msi	<p><b>MinGW requirements and limitations</b></p> <p>When using MinGW as a compiler, note that the following are currently <i>not</i> supported:</p> <ul style="list-style-type: none"> <li>HARDDISK, BIOS, and DONGLE_HASPHL HostIDs (see <a href="#">LMX_Hostid</a>)</li> <li>Virtual machine detection (SHARE = VIRTUAL; see <a href="#">FEATURE settings</a>)</li> <li>The flags <code>LMX_LOGICAL_CPU_COUNT</code> and <code>LMX_PHYSICAL_CPU_COUNT</code>, used with <a href="#">LMX_Checkout</a>.</li> </ul> <p>GCC version 8.1.0 must be used when building LM-X with MinGW. The exemplary output for <code>GCC -v</code> is as follows (using MinGW32 for our example):</p> <pre>COLLECT_GCC=gcc COLLECT_LTO_WRAPPER=C:/mingw32/mingw32/bin/./libexec/gcc/i686-w64-mingw32/8.1.0/lto-wrapper.exe Target: i686-w64-mingw32 Configured with: ../../src/gcc-8.1.0/configure --host=i686-w64-mingw32 --build=i686-w64-mingw32 --target=i686-w64-mingw32 --prefix=/mingw32 --with-sysroot=/c:/mingw32/i686-530-posix-dwarf-rt_v4-rev0/mingw32 --with-gxx-include-dir=/mingw32/i686-w64-mingw32/include/c++ --enable-shared --enable-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads=posix --enable-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=release --enable-fully-dynamic-string --enable-version-specific-runtime-libs --disable-sjlj-exceptions --with-dwarf2 --disable-lsl-version-check --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=i686 --with-tune=generic --with-libiconv --with-system-zlib --with-gmp=/c:/mingw32/prerequisites/i686-w64-mingw32-static --with-mpfr=/c:/mingw32</pre>

MinGW64 (with GCC 8.1.0)	lmx-sdk-version #_mingw64_x64.msi	<pre> /prerequisites/i686-w64-mingw32-static --with-mpc=/c/mingw530/prerequisites/i686-w64-mingw32-static --with-isl=/c/mingw530 /prerequisites/i686-w64-mingw32-static --with-pkgversion='i686-posix-dwarf-rev0, Built by MinGW-W64 project' --with- bugurl=http://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -I/c/mingw530/i686-530-posix-dwarf-rt_v4-rev0/mingw32/opt/include -I/c/mingw530 /prerequisites/i686-zlib-static/include -I/c/mingw530/prerequisites/i686-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -I/c /mingw530/i686-530-posix-dwarf-rt_v4-rev0/mingw32/opt/include -I/c/mingw530/prerequisites/i686-zlib-static/include -I/c/mingw530/prerequisites /i686-w64-mingw32-static/include' CPPFLAGS= LDFlags='-pipe -L/c/mingw530/i686-530-posix-dwarf-rt_v4-rev0/mingw32/opt/lib -L/c/mingw530 /prerequisites/i686-zlib-static/lib -L/c/mingw530/prerequisites/i686-w64-mingw32-static/lib -Wl,--large-address-aware' Thread model: posix gcc version 8.1.0 (i686-posix-dwarf-rev0, Built by MinGW-W64 project) </pre>
--------------------------	--------------------------------------	--

**Note:** The platforms supported by LM-X are identified by executables compiled with LM-X rather than by the actual OS. For example, Win32 and Win64 are considered different platforms.

The following table specifies the platforms no longer supported by LM-X License Manager.

Platform	Last Supported Version
CentOS 5	LM-X v4.8.10
Linux (32 bit) ARM	LM-X v4.8.2
Solaris (64 bit) x64	LM-X v4.8.1
Solaris (64 bit) Sparc64	LM-X v4.8.1
AIX PPC64	LM-X v4.7.6
HP-UX (64 bit) IA64	LM-X v4.7.6
Windows 2003	LM-X v4.6.5
Windows XP	LM-X v4.6.5
AIX PPC32	LM-X v4.6
HPUX IA64 ILP32	LM-X v4.6
Solaris (32 bit) x86	LM-X v4.6
Solaris (32 bit) Sparc	LM-X v4.6
Linux IA64	LM-X v4.5.1
FreeBSD (32 bit)	LM-X v4.5.1
Mac OS X (Universal) for 10.6	LM-X v4.5.1
Mac OS X (Universal) for 10.7	LM-X v4.5.1

# Glossary

This glossary lists terms that are specific to LM-X License Manager.

Term	Description
Client application	An application program that requests or receives a license.
Contact	A customer's contact person.
Customer	A company that orders licenses from a Vendor.
End user	A machine defined by a set of HostIDs, for example a personal computer, server, virtual machine, or any other equipment running licensed software.
Feature	<p>Any functionality that needs to be licensed. The definition of <i>feature</i> is developer-dependent, according to the application and to the developer's requirements.</p> <p>For example, a feature can consist of:</p> <ul style="list-style-type: none"> <li>• A single program irrespective of its version (for example, MS Office).</li> <li>• Version (one feature can have multiple versions, such as MS Office 2007, 2010, 2013 and so on).</li> <li>• An application system that consists of multiple programs.</li> <li>• A module of a program.</li> </ul>
Floating license	<p>A license that can authorize usage of an application for users on a network. A license server is required to manage a floating license. With this type of license, the number of concurrent users is counted with the licensed application usable by only a specified number of users at any time. A variation of this scheme is a license that can be locked to work both with specific client computers and with a specified server computer.</p> <p>A floating license is sometimes referred to as network licensing.</p>
Heartbeat	A message sent from the client application and acknowledged by the license server to ensure that both the license server and the client are still up and running. This action is triggered periodically.
HostID	A unique machine value that can be used to lock a license file to a specific host.
License	A text file shipped to an end user, which provides one or more features. Features contain information about specific modules in your program that are licensed. Each feature in a license has its own key, where all security information is encrypted.
Imx-serv	A process that serves licenses for clients over a network. Sometimes referred to as a license server.
Log file	One or more ASCII text files written by a license server. A log file contains status information useful for debugging purposes.
Node-locked license	<p>A license that can authorize use of an application running on a single specific machine, as opposed to being on a network. Node-locked licenses do not require a license server because they are uncounted.</p> <p>A node-locked license is sometimes referred to as local licensing.</p>
Security configuration file	LM-X security configuration file. A file, specific to the vendor, which holds cryptographic keys and is used for generating licenses and for generating specific files required for compiling the client application. Created by the developer tool Imxdev.
Terminal Server Client	A Terminal Server Client, also known as a Remote Desktop, is a Windows feature that lets you administrate Windows on a remote host. From a licensing point of view, this may be prone to abuse if a few people are running the application simultaneously.
Vendor name	The name of the vendor that is used in license files.
Unix system	Refers to Linux, Solaris, HP-UX, AIX and FreeBSD platforms. (See <a href="#">PLATFORMS</a> for more platform-specific information).
Windows system	Refers to Win32 or Win64 platform.

## Node-locked version vs full version

LM-X is available as a node-locked-only version for vendors who need only distribute node-locked licenses for their software.

The node-locked version of LM-X identical to the full version, except for the following limitations:

- [LMX\\_Checkout](#) ignores any network licenses.
- LM-X will refuse any attempts to start a license server.
- [Automatic server discovery \(floating licenses only\)](#) is not available.