NG Logic

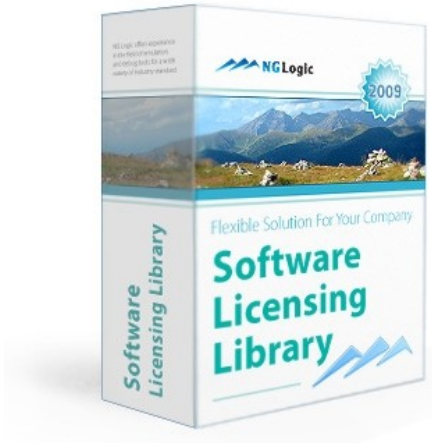
NG Logic

Users Guide Version 1.4

NGL Software Licensing Library

12/4/2008

Contents:



NGL SOFTWARE LICENSING LIBRARY

- your key to increased software profits

1 Introduction

The NGL Software Licensing Library is a complete and flexible solution for your company licensing needs. It uses license keys, which are copy and paste-able text strings, that can be sent via the web or email. Each license key is issued for specific machine and allows it to run specific application with selected features until expiration time. The library can be used for protecting applications targeted at individual users as well as software used in large corporations. The library offers a high degree of security with RSA cipher, different licensing schemes and cross-platform portability.

1.1 Software Licensing Library Features

The most important features of the product are:

- Strong 1024-bit RSA encryption of licenses
- Each license contains application identifier, host id and expiration time
- Licenses can securely carry additional licensing information e.g. application-related features and user-defined custom data
- Portable object-oriented C++ API for client applications
- Automatic host identifier generation based on network adapter MAC and other hardware components. Possibility to provide custom host ids.
- Possibility to limit the number of instances of running application
- Interface in Python and C for automatic license generation, e.g. from a web page or a script (also available other connectors for Java, C# and other technologies)
- License request protocol from client application to a web or mail server that allows it to securely transport user name and password (RSA encrypted).
- Cross-platform License Server application that allows corporation to centralize license management by usage of remote licenses.
- The application code compiled to native assembly code and important data are obfuscated to protect from reverse engineering.
- This package may also work in extra-high security mode with appropriate server and client software
- Multiple license instances in one file
- Operating systems supported: Windows, Linux, Unix, Solaris, MacOS X

The library employs efficient and flexible security model derived from private-public encryption scheme. This means that our customers are the only party that can issue licenses for their software.

The License Server software allows storage of all licenses for organization in one place and license acquisition as needed. The License Server Controller application provides management for the available licenses, handles requests for subsequent licenses and supports problem diagnoses with help of detailed log file.

This product can be easily customized to meet your company licensing needs. In particular, NG Logic can deliver bindings to Java and .NET platform. We can also develop custom license generation automations or help your company to customize the Licensing Library for requirements of your application and used IT technology.

1.2 Requirements

The client application has to meet following requirements:

- C++ compiler (preferably MSVC for Windows platform or gcc for Linux)
- linker that is able to statically link C++ code
- we provide several versions of the library compiled by MS VC .NET 2003, MS VC 8.0 2005 and MS VC 9.0 2008 as well as by gcc (there is also a possibility to recompile the library for you using other compilers and environments)
- in case of using other library connectors such as Python or Java connector there is no need to have access for MS VC or other C++ compiler; we provide you with the complex solution including dynamic shared library (dll/so file) and appropriate Python/Java/C#/other code managing the library

The machine used to run the License Server has to meet following requirements:

- Pentium-class processor
- Windows NT/2000/XP and newer or Linux operating system
- 256 MB RAM

1.3 Distribution

The product is distributed as a SDK (Software Development Kit) containing compiled C++ library and headers, License Generator application for issuing licenses and License Server application. The package also contains sample applications demonstrating the use of licensing API and description of process of generating licenses and using it.

If you are interested in evaluating our product, please contact our sales department as sales@nglogic.com.

1.4 Package Contents

The Licensing Software library contains the following files:

(NOTE: the *VER* text stands for a compiler version which the SDK package has been build with e.g. MS VC 9.0)

Description	**Location relative to the root installation directory of SDK**
A statically linked library built with Microsoft C++ toolchain (lib subdirectory)	/lib*VER*
A set of C++ header files for inclusion in the licensed software	/include
LicenceGenerator application for generation of license keys	../LicenseGenerator
LicenceServer application for management of remote licences	../LicenseServer
Sample license files	/project/licenses
Sample C++ solution featuring:	/project
C++/C source code for sample applications	/project/sources
• Example demonstrating usage of local (node-locked) licenses	LocalLicenseDemo.cpp
• Example demonstrating usage of hardware-locked licenses	HardwareLicenseDemo.cpp
• Example demonstrating usage of floating (remote) licenses	RemoteLicenseDemo.cpp
Compiled executable files coming from above examples	/project/Debug*VER*/
Server-side Python library	/server/RegisServer
Server-side C bindings (provided as Windows dll) for	/server/RegisServerDll

Description	Location relative to the root installation directory of SDK
server Python library	
SDK Developer Guide	/doc/guide.pdf

We also provide the LicenseServer software as separate package with convenient installer. The resident part of the application installs itself as a system service under Windows NT/2000/XP/Vista systems and as a daemon on the Linux and MacOS systems.

2 Installation

Included with your purchase is the installation CD or downloadable archive containing the SDK setup (SoftwareLicencingSDK binary file), LicenseServer application in a separate package (LicenceServerSetup) for distribution for users of floating licensing scheme and a LicenceGenerator application in separate installer (LicenceGeneratorSetup binary file).

Insert the installation CD into your CD ROM drive. If auto detect is enabled in your system environment setup the installation program will launch automatically. Otherwise, double-click desired application setup file to launch installation program.

We can also provide you with other ways of installation such as: RPM/DEB packages for Linux, MSI Windows package intended to be managed by customer companies rather than by individual users or other. If you are interested in evaluating such packages, please contact our sales department as sales@nglogic.com.

3 Using Licensing Library

3.1 Definitions

Target application – the application that is to be secured.

Client library – the library providing licensing services that will be embedded in the target application.

Server library – the library that is able to generate licenses from given data

LicenseGenerator – the GUI and command-line application that enables users to generate licenses interactively.

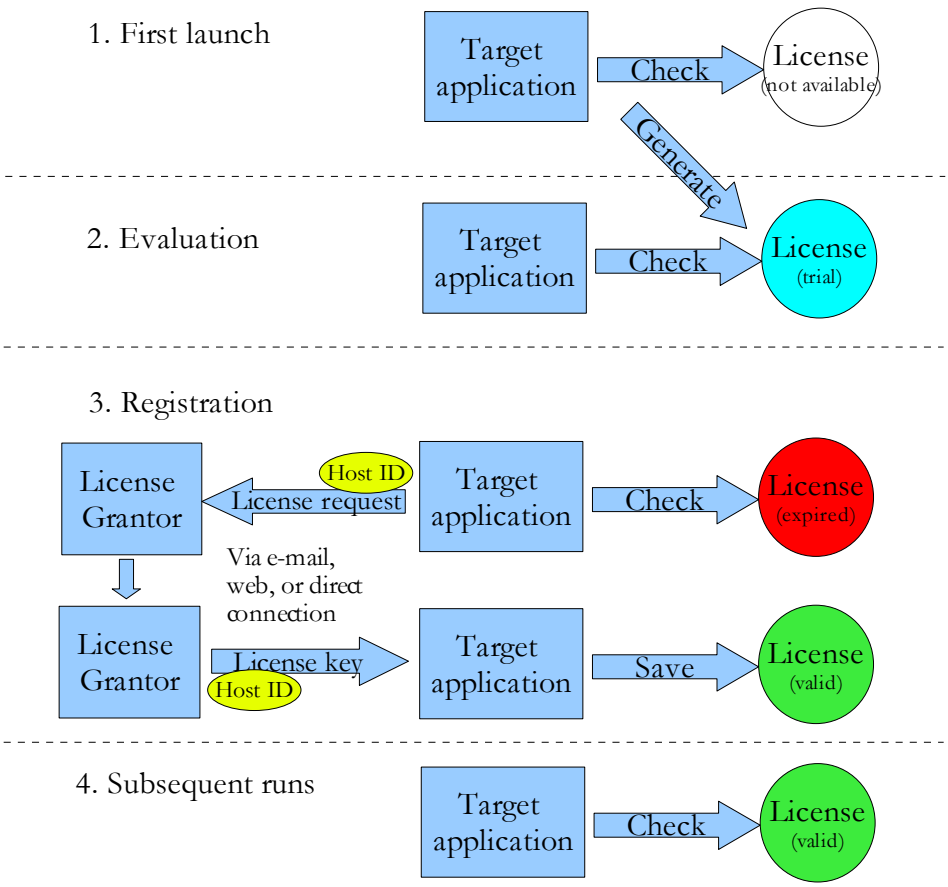
LicenseServer – application holding a pool of licenses and providing them to target applications as needed.

License Grantor – The body that owns the software and is entitled to grant licenses for the target application

3.2 Overview of licensing scheme

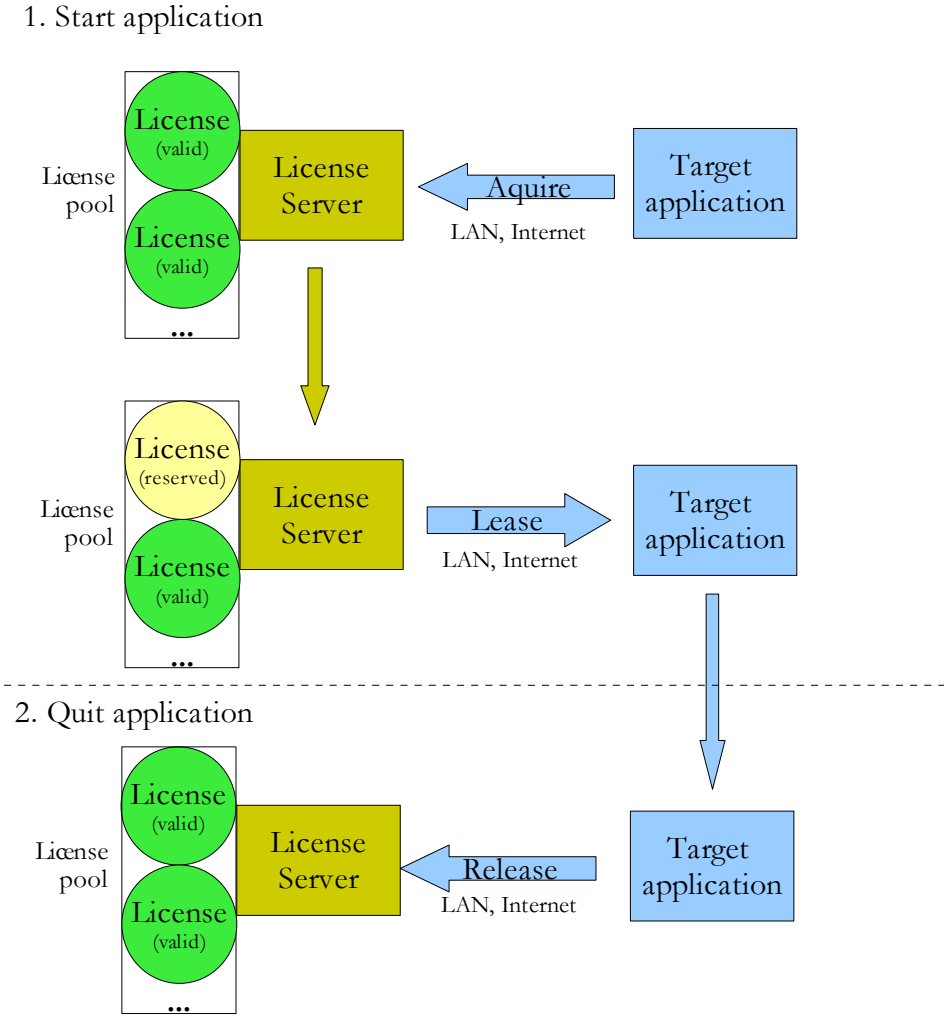
The library provides couple licensing schemes out-of-the-box – **local** and **floating**, but it can be extended to accommodate additional schemes (for example external hardware based scheme). In such case the hardware licensing scheme works the same way as the local scheme apart from one difference – for local licensing we use host ids related to current user machine which is used to run the Target application. If you use a hardware license it contains hardware id related to your hardware key rather than to your PC. It means that containing the hardware key which a license has been generated for you are able to run your application with the key on many computers and for instance work with the Target application at your office as well as at home.

The **local licensing** scheme is illustrated on the following picture:



Each time an application is launched it checks for valid licenses. If this is the first launch, it can not find any licenses. In this case, a temporary evaluation license can be created that will expire in short period of time. After that, the user has to obtain a valid license key by sending license request containing host identification string to the LicenseGrantor, optionally with user name and encrypted password. The transfer method may be either web browser, e-mail, or direct communication with software grantor's server via the Internet. The License Grantor can now process the request and generate license key for specific period of time using a command-line utility, GUI application or using a customized script (also web-based scripts)

Another option, **floating licensing**, is presented on the next illustration (the evaluation phase was omitted here for clarity).



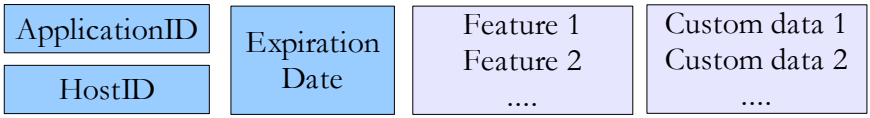
In the floating licensing mode, the target application connects directly to specified license server host to obtain license upon each execution. The license is reserved from the pool of licenses available to LicenseServer instance. Information about the license is returned to the target application that keeps the connection to the server alive until termination.

When it terminates, the server is notified and put the license back to the pool for further use by another application instances.

Licenses to the LicenseServer can be added in the same way as to the target application in local licensing scheme – via e-mail, web, or direct connection. The advantages of the floating licensing include centralized license management suitable for larger organizations and optimal license usage. If you needed a pool containing 5 licenses and there were many more computers in your company that would be the best and the most flexible solution for your, since you would be able to work with any of the computers using the Target application and that would only be limited by the count of active Target applications and amount of the licenses in the pool. In this scheme the License Grantor does not need to know the host ids of machines that Target Application is running on – only the LicenseServer host id.

3.3 License parameters

The licenses contain two types of information: mandatory fields, namely: Application Identifier, Host Identifier, Expiration Time, and optional fields, called Features, that can be used to define the license subject more specifically. Features is a list of ordinary strings containing arbitrary information that Target Application can interpret and use according to the build-in logic. In addition, custom data in either text or binary form can be encoded in the license and then retrieved by Target Application. The license key structure is illustrated on the following picture:



For instance if your Target application is intended to be module-based you can generate a license using Features which describe which module should be turn on and which need to be disabled for your customer. Then using Custom Data you can encode some additional data for a license owner such as a license type description (e.g. educational/students or developer license) or some personal customer data such as his address etc...

4 Technical description and security

The licensing security is implemented using public-private encryption mechanism (RSA). This results in very convenient, flexible and secure licensing schemes.

The basis for software licensing is a private-public RSA key pair in PEM format that is assigned to software provider (License Grantor). The private part of the key should be kept in secure place and never disclosed to third-parties as it allows to create unlimited number of licenses for any host. The RSA keys can be generated with LicenseGenerator software or any other RSA-enabled tool like OpenSSL. This allows for arbitrary key length enabling to increase security level as needed.

The private RSA key is used in conjunction with LicenseGenerator software or server-side library in order to generate new licenses. The public RSA key should be sent to NG Logic to be embedded in the library that is linked with Target Application.

During the license generation, the license information is digitally signed with private key. When the license is loaded into Target Application the signature correctness is checked. This way only the owner of the RSA private key may issue valid licenses.

Each license is tied to specific host, that is identified by host id - a 8 or more-character string. In the Software Licensing Library the host identifier is based on MAC (Media Access Control) of Network Interface Card, but it can be provided by Target Application in different way (e.g. based on hardware that is provided with application). The license key itself can be distributed over non-secure channels, published on web or left on a not-encrypted disk as it is usable only on one specific machine. In order to enable copy-and-paste operation on key it is always base64-encoded so it consists only of letters, symbols and digits.

The license request is a piece of information that goes from Target Application to License Grantor. It contains information needed to issue a license: application id, host id, and requested features. Request can be generated in the Target Application by the provided library or by LicenseServer. Then, request can be parsed by LicenseGenerator application or by server-side library and license can be automatically generated. To enable more control over the process of license generation, the request can also contain a user name and an encrypted password. This allows for different licensing automation scenarios involving custom user database. The license request is also a base64-encoded string that can be transferred over non-secure channels. The password is encrypted with public RSA key that is embedded in the Target Application and can be decrypted only by owner of private key from the pair. The whole request is also digitally signed to avoid malicious changes (for example changing host id in order to get a license for another machine).

The license key along with additional information (user and company name) is stored in special files in the file system. As the licenses are public, the location of the licenses does

not need to be hidden. The library also records the current system time to prevent clock-shifting attack. The time is recorded in various places in the system.

Floating licenses are using the same RSA encryption mechanism as described above. However, the license must contain special features with names starting with Floating prefix in order to be used with LicenseServer application. The LicenseGenerator application allows to set various options of remote licenses. The Target Application has to contact the LicenseServer over TCP/IP (default port 9091) in order to obtain a valid license. The connection is sustained until the Target Application closes and license is released. Floating license keys are tied to the machine that LicenseServer is running on, instead of machine the Target Application machine. Nevertheless, the remote licensing scheme is as secure as the local: all communication between LicenseServer and Target Application is signed and protected from various attacks. In particular, the client communicates with server every couple of minutes to see if it is still alive.

Each remote license has a license identifier assigned during creation and stored in feature list. Its purpose is to allow to load to LicenseServer more than one licence for the same application and allow the licenses seats to sum. Only licenses with different identifiers can be loaded in the same time to the server. Otherwise it would be possible to load additional licenses by duplicating the files license is stored.

5 Software Integration

The Software Licensing Library provides convenient and flexible APIs. For the Target Application an object-oriented API in C++ is available. We would like to meet your requirements, so we can also deliver for you APIs in C, C#, Java or other programming languages.

License Grantor API for managing license request and license key generation is provided in Python and C. The Python version is a Python module that can be simply imported in script or web server application. The C interface is provided as a dynamic loaded library (DLL/SO) with exported functions.

The detailed software integration description can be found in Developers Manual document.

6 Bundled applications

The supplied applications provide convenient GUI for users.

6.1 LicenseGenerator

LicenseGenerator application is a simple utility that allows to manage licensing configurations and generate licenses. A configuration file is a simple INI-like file that stores information about licensed application. It is used by LicenseGenerator to save settings for license generation as well as LicenseServer to generate license requests.

Here is presented a sample config file. The comment lines starting with colon will explaing meaning of each option.

```
[Application]
;Target Application unique ID
ApplicationID=Sample App
;Keyfile used to sign the license
KeyFile=config/samplekey.pem
;Descriptive text
Description="Application Sample v. 1.0"
;Version is currently not used
Version=
;URLs used in LicenseServer
PurchaseURL=http://sample.com
PurchaseEmail=order@sample.com

;File containing custom data
;You can store any data in here, also in binary form
;(but without \0's)
;This file will be splitted into lines, each line will become
;a separate custom data item
;Contents of this file will be appended to custom data
specified ;in this file at the end
CustomDataFile=config/custom.dat

;LicenseScheme controls the type of license to generate:
;currently supported values: Local, Hardware, Floating
;you can specify more than one value separated by spaces
;the default is Local
LicenseScheme=Local

;Defaults parameters for various license schemes
;Default time for which a license is valid when not specified
directly, in days
DefaultTimeDelta=720
;Default hardware type identifier for hardware keys
```

```
DefaultHardwareType=USBKey
;Default license identifier for remote licenses
DefaultLicenseID=1
;Default number of seats of the single remote license
DefaultSeats=1

;List of features to embed in the license
[Features]
Feature1=Feature_cool
Feature2=Foo
Feature3=Big

; Description of features to be placed in License Server
[FeatureDescriptions]
Feature1=Very cool feature
Feature2=Another feature
Feature3=Very big feature
Feature4=Server (floating) license

;additional custom data specified in the name=value form
[CustomData]
URL=http://www.nglogic.com
Support=support@nglogic.com
```

The licenses can be generated by our command line tool ***LicGen***, capable of generating all kinds of keys used by the library based on config file and command-line options. Please run *LicGen* binary file without parameters or with *"--help"* switch to receive information about the usage.

Note:

On demand we can provide you with a GUI application to manage your config.ini files, create new ones and edit their settings.

There is also a GUI version of LicenseGenerator. The upper left part of the screen contains list of available licenses and buttons to add, remove, save and reload a configuration. The upper right part of the screen allows to view and edit license configuration information: application id, version, default time the license is granted for, web address and email for submission of license requests (used in LicenseServer to help the user purchase new licenses), private key file and a list of possible features and their descriptions presented in the user interface.

The bottom part of the application screen allows to generate new licenses according to selected configuration. You can choose the features to be placed in the license, the expiration time, mark the license as floating and set license id and number of seats. You

should also supply the host identifier. After that pushing the Generate button will show the generated license key.

The configurations saved by this application can be also used in the LicenseServer application for convenient ordering of licenses. You may find the *.cfg files describing the configuration in the config subdirectory of the LicenseGenerator application installation.

6.2 LicenseServer

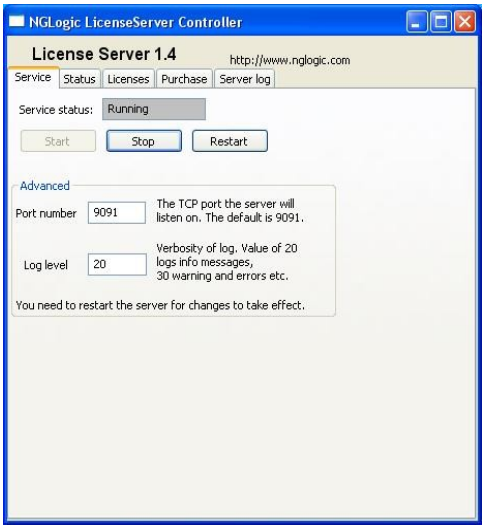
The LicenseServer application itself is installed as a system service and started automatically when system starts. However, you can also start it from command line using command:

```
LicenseServer [port] [log_level]
```

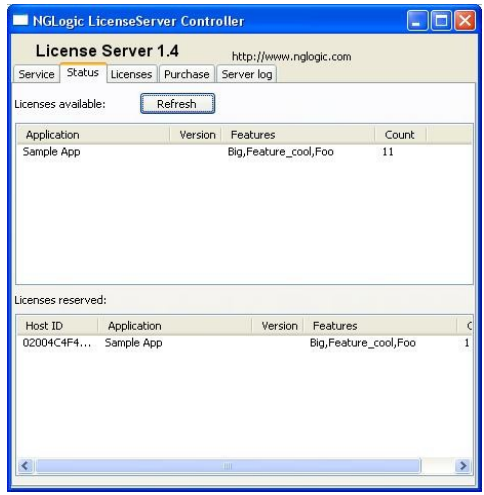
where port is the TCP port number for the server to listen on (default: 9091) and log_level controls the verbosity of log: 20 stands for INFO messages, 25 for LICENSE, 30 for WARNING, 40 for ERROR (default: 20). You can stop the server by hitting Ctrl-Break keys.

In order to control the service a LicenseServer Controller application is provided. You can launch it using an icon in Start Menu. The application consists of five tabs: *Service*, *Status, Licenses, Purchase, Server Log*.

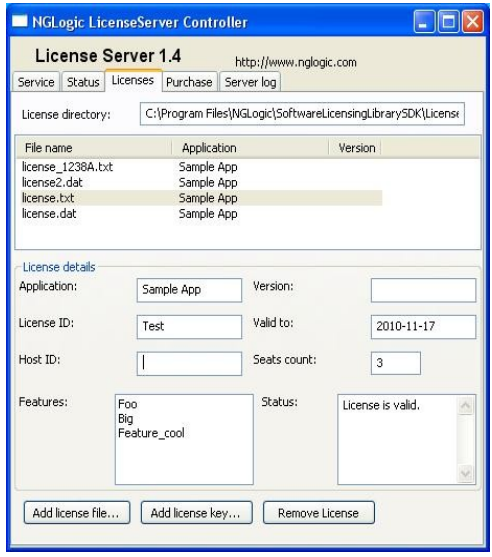
The *Service* page allows to start, stop and restart the LicenseServer service and control the server port and log verbosity:



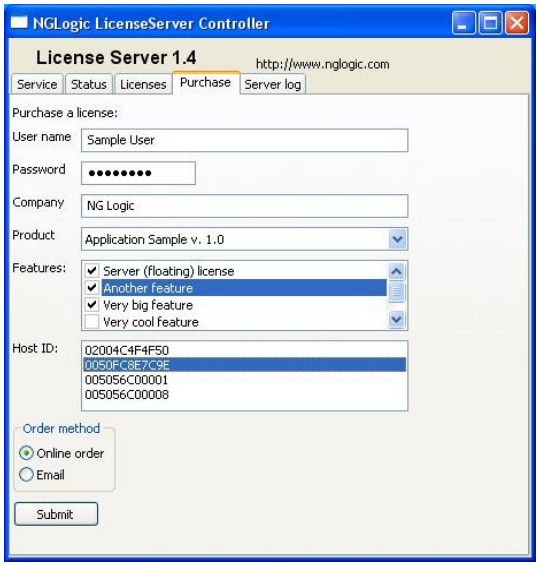
The next tab, *Status*, allows to review currently reserved and available licenses:



The licenses available can be inspected in detail using the *Licenses* tab. It is a file-centric view: You can inspect licenses stored in each file in the licenses subdirectory of your LicenseServer installation. You may also add new license keys or files and remove the expired licenses.

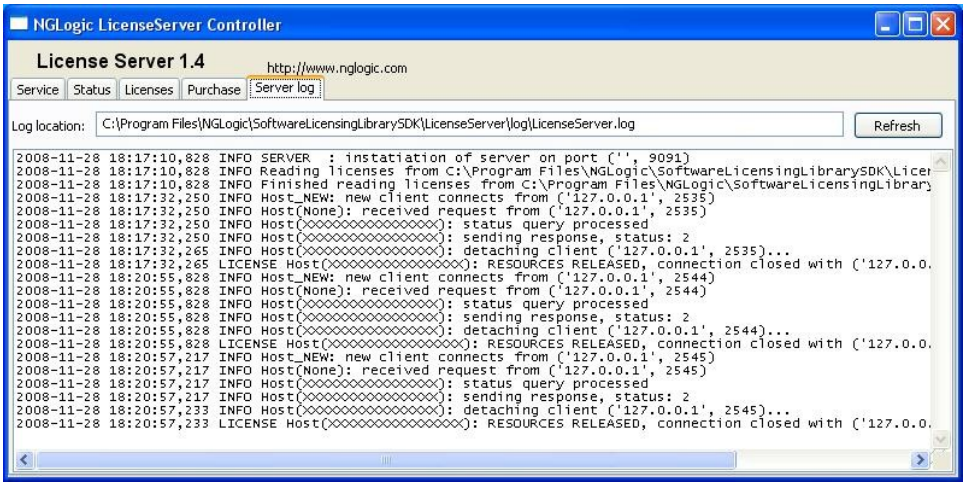


The *Purchase* tab allows to add (order) more licenses for that LicenseServer instance. You can enter user name, optional password, company, specify product and its features, one of host identifiers and sent it to License Grantor either by email or web. The next steps depend on the procedure adopted by the grantor.



The list of applications and its features comes from configuration files put into the config subdirectory of your LicenseServer application. The format of the file is the same as in the LicenseGenerator application, so you can safely create the config file in the LicenseGenerator and then copy to the LicenseServer. NG Logic can prepare a customized installer for LicenseServer with your config included.

The last tab, *Server log*, allows to view server log that contains information about functioning of the server. You can inspect the log in case there are some problems in functioning of the server. The log file is stored in the log subdirectory of your local LicenseServer application.



There are four categories of log messages:

Name	*Level*	*Description*
INFO	20	Informational messages displaying normal server operation
LICENSE	25	This messages are issued when license is reserved or released
WARNING	30	Inform about unusual situations the server encountered
ERROR	40	These are error messages

You can set the logging level on the *Service* tab.

Currently for the Unix-based systems and MacOS X the GUI License Server controller is not available yet. However if you need such tool, please contact us and we will provide it for you. If you are interested in how to run and use the LicenseServer under Unix, please go to the *"LicenseServer"* directory and read appropriate detailed documentation.

7 Sample Programs

Several examples have been supplied in the SDK.

7.1 Target Application API

There are three command line applications in the /project/Debug*/ subdirectory of the SDK installation. The first application (LocalLicenseDemo) loads license file located in the ../license subdirectory relative to the current directory and prints its parameters. The second one (RemoteLicenseDemo) connects to local LicenseServer on port 9091 and tries to acquire license for Sample Application (configuration for this application is included in both LicenseGenerator and LicenseServer). The last one is a sample of hardware key locking. It does not require hardware key to run though – there is a stub implementation of the to run – please contact us if you are interested in evaluation of hardware key options. Those application source code contains enough comments to start integrating the API to your own application simply by copy&paste to your source and modifyng relevant constants. For more information about using sample applications go to main *readme.txt* file or execute demo programs with *"--help"* option.

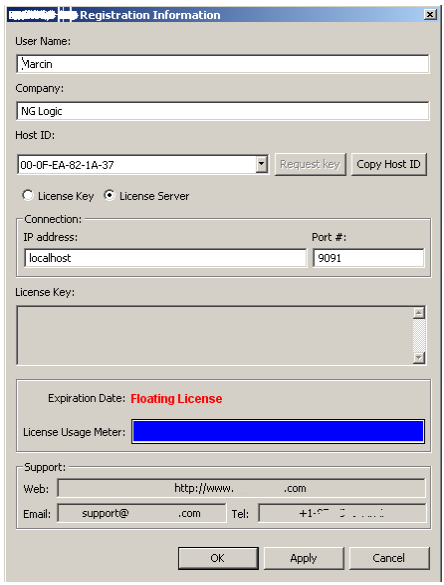
7.2 Server API

In the server subdirectory of SDK installation you can find examples of usage of server-side API. To be able to run the examples, you will need Python interpreter version 2.4 or later and PyCrypto and TlsLite libraries installed. Current PyCrypto version can be downloaded from http://www.amk.ca/python/code/crypto website while TlsLite from http://trevp.net/tlslite/ website.

The example includes Python source files (in RegisServer subdirectory) for command line utility included in LicenseGenerator (RegisServer.py) and a sample CGI script that generates the license after verification of a password (CGIDoGenLicense.py) and accompanying PHP web page (html/index.php).

7.3 Sample registration window in target application

Below you can find a sample window in a target application that displays licensing information and allows to enter license key or connect to LicenseServer. NG Logic can provide source for this window on demand.



Note:

This window has been developed in Java 1.4 using SWT technology. However we can provide for you similar windows implemented in other programming languages or using different GUI technologies (for instance SWING or AWT for Java, MFC or wxWindows for C++ etc...). On demand we also offer developing a license window or other license-related components customized for your company and designed in a way you will provide.

NG Logic, December 2008
sales@nglogic.com

Tel.: +48 505091662

http://www.nglogic.com
email: nglogic@nglogic.com