

LICENSE4J LICENSE MANAGER USER GUIDE

VERSION 4.7.2

LICENSE4J
www.license4j.com

Table of Contents

Getting Started.....	4
Managing Products.....	6
Create Product.....	6
Edit Product.....	7
Refresh, Delete Product.....	8
Disable and Enable Product Use.....	8
Change Product Category.....	10
View Key Pair.....	10
Product Update Notifications.....	12
Managing Categories.....	13
Create, Edit Categories.....	13
Refresh, Delete Categories.....	14
Backup and Restore Products.....	14
Managing Licenses.....	15
Generate New License.....	18
Generate New License in Bulk.....	32
View License.....	33
Modify License.....	33
Clone License.....	34
Delete License.....	34
Export License.....	34
Export Licenses to CSV File.....	34
Search License.....	35
Test License.....	36
License Deactivation.....	37
Manage License Templates.....	38
Manage Activations.....	38
Manage Online Basic Key Usage.....	41
Automatic License Generation Settings.....	43
Display License Generation URL.....	47
Display HTML Form Source for License Generation.....	48

Send License E-Mail.....	50
License Messages.....	52
License Tags.....	54
License Use Update (last time used & total use count).....	54
License Use Tracking.....	55
Managing Modification Keys.....	56
Modification Key Generation.....	58
Modification Key Generation in Bulk.....	61
Manage Modification Key Templates.....	62
Manage Modification Key Auto Generation Settings.....	64
Customer List.....	67
Tools and Options.....	67
Database Storage Options.....	68
Online.License4J Settings.....	69
Online Storage Connection Options.....	69
Account Information.....	70
Log Viewer.....	71
Hardware ID Use.....	72
License4J Auto License Generation and Activation Server.....	73
License4J Runtime Library Integration.....	79
ValidationStatus.....	79
ActivationStatus.....	82
ModificationStatus.....	84
License Validation Code.....	86
Easy License Validation Code.....	86
Basic and Cryptographically Secure License Key Validation.....	88
License Text Validation.....	90
Floating License Text Validation.....	90
Online License Key Floating Over Internet Validation.....	94
Auto License Activation.....	98
Manual License Activation.....	100
Auto License Deactivation.....	101

Manual License Deactivation.....	101
Manual License Modification.....	102
License Availability (Blacklist) Check.....	103
Product Update Check.....	104
Product Update Message Check.....	104
License Message Check.....	105
License Modification.....	105
License Use Information Update Code.....	106
License Use Tracking Code.....	106
License4J Runtime Library Obfuscation.....	108
LicenseValidator Debug Log.....	108
SSL Verification and TrustManager.....	108
License4J Development Library Integration.....	110
License4J License Manager License.....	112
License Manager GUI Tool Error Logging.....	116

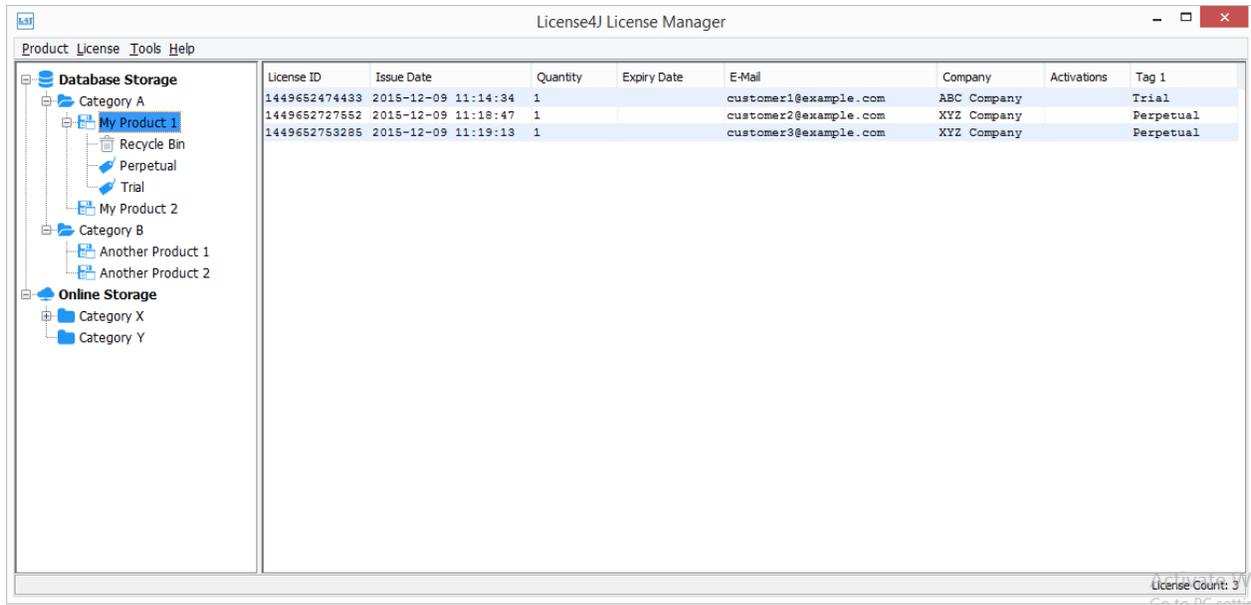
Getting Started

License4J License Manager supports generation of license text, license key, and floating license text. License Manager stores all products, templates, licenses, and settings in a database. Database can be on a local embedded Derby database or on database servers: MySQL, PostgreSQL, and Microsoft SQL Server. Online.License4J system is an online storage for products and licenses, it is a license hosting system.

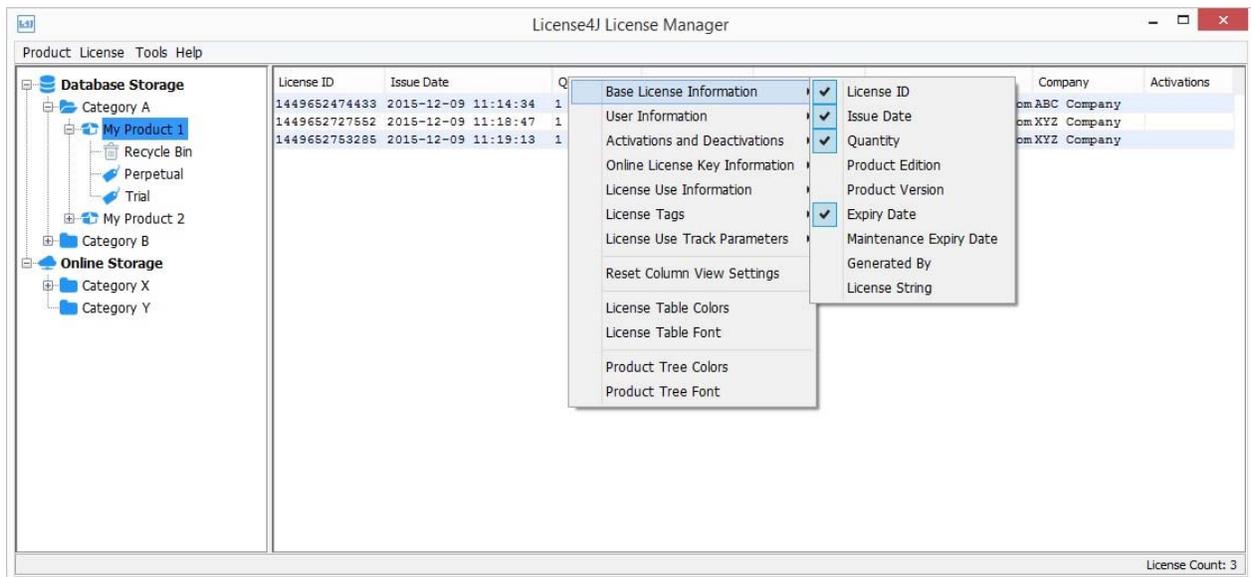
License4J runtime library includes necessary methods to validate and activate a license. It is a free library and is distributed within any software product for license validation and activation.

Download packages on www.license4j.com/download/ includes License Manager Setup file, Floating License Server Setup file, Library jar files, examples, and documentation. Oracle Java 1.7+ is required to use all features, if Java 1.6 is used secure license key and if defined activation codes cannot be generated. Basic license key, license text and floating license text can be generated and validated with Java 1.6.

License Manager GUI is an easy to use software to manage products and licenses. Main GUI window includes a left tree panel for navigation on products and categories; and the table on the right displays licenses for the selected product.



Main GUI window size and location are saved to local configuration file on application startup, therefore next time it is started, size and location will be same. Another setting which is saved to configuration file is license table column visibility settings. License table columns can be selected with a pop up menu which comes when clicked on table header.



Visible column setting and column width values is saved on application shutdown. License table and left product tree colors and font properties are customized on the same table header pop up menu.

Managing Products

License Manager allows definition of products to categorize and manage generated licenses. Public and private key pair is generated when product is created, and it cannot be changed later.

Create Product

Create New Product menu item brings a window as below for creation of a new product in license manager. Unique ID is for license manager database usage and generated automatically. ID is a required field because it will be used in license validation, product cannot be created without an ID.

Available key pair algorithms are RSA and DSA with 1024 and 2048 bit key size. For generation of license keys EC algorithm is used.

Create New Product

Product

Unique ID: 1424335445918

Name: My Product

ID: my-product-12345678

Edition:

Version:

Key Pair

Create New RSA Key size: 1024

Load From File

Save Cancel

If there is a key pair saved from another product previously, it can load that key pair from selected file. Therefore two different products with same key pair (so that public key and private key) can be created.

Edit Product

Product information can be changed with *Edit* menu item. Key pair settings cannot be changed because generated licenses for this product becomes invalid if it changes. If ID is changed all generated licenses before will fail in license validation since license validation checks for product id given in validate method of runtime library.

Edit Product

Product

Unique ID: 1424335445918

Name: My Product

ID: my-product-12345678

Edition:

Version:

Key Pair

Create New RSA Key size: 1024

Load From File

Save Cancel

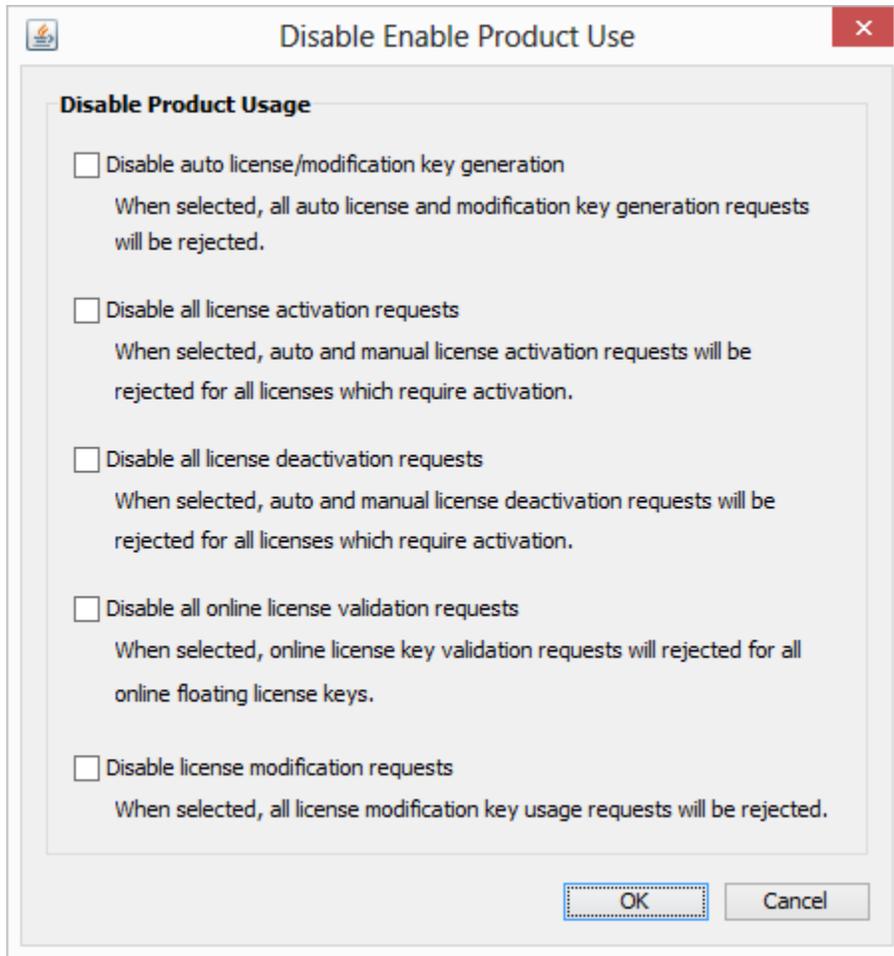
Refresh, Delete Product

If a remote database or Online.License4J is used as the storage and there are other users connecting to the same database, *Refresh* menu item can be used to see changed licenses on database. It refreshes product information and licenses.

Products can be deleted with *Delete* menu item; deleting a product will also delete all licenses, activations, templates, and license generation settings for that product.

Disable and Enable Product Use

Activation, deactivation, validation, generation and modification can be disabled for a product. Product menu and context menu has a menu item as *Disable Enable Product Use*, and it brings a window as in the screenshot below. When any of the product feature is disabled, product item on the left tree is displayed in red color.



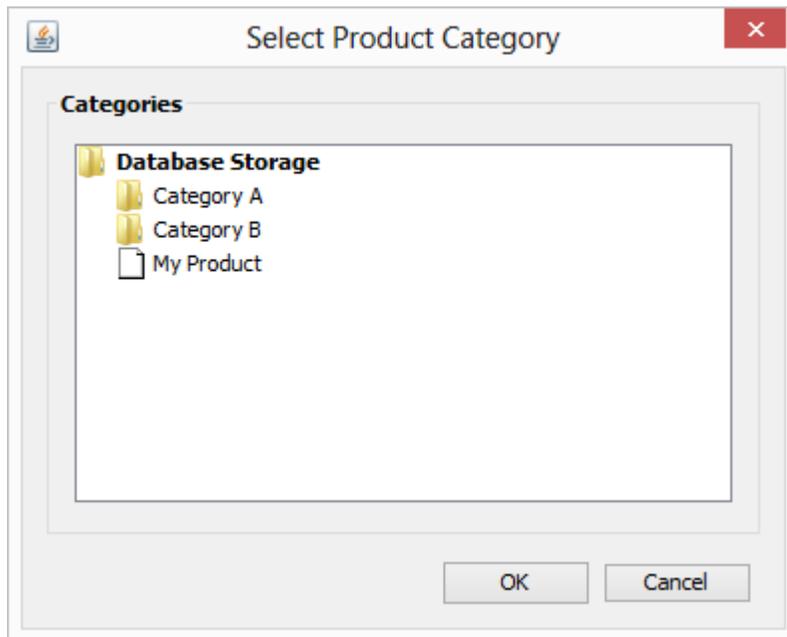
License and modification key auto generation can be completely disabled with the first checkbox. All license activation and requests can be rejected with the second checkbox. All license deactivation requests can be rejected with the third checkbox. Online license key validation can be disabled with fourth checkbox. When last checkbox is selected all license modification requests are rejected.

When any of the feature is disabled, all generated licenses are affected, so it may be used to temporarily disable a product and make modifications, updates etc.

Change Product Category

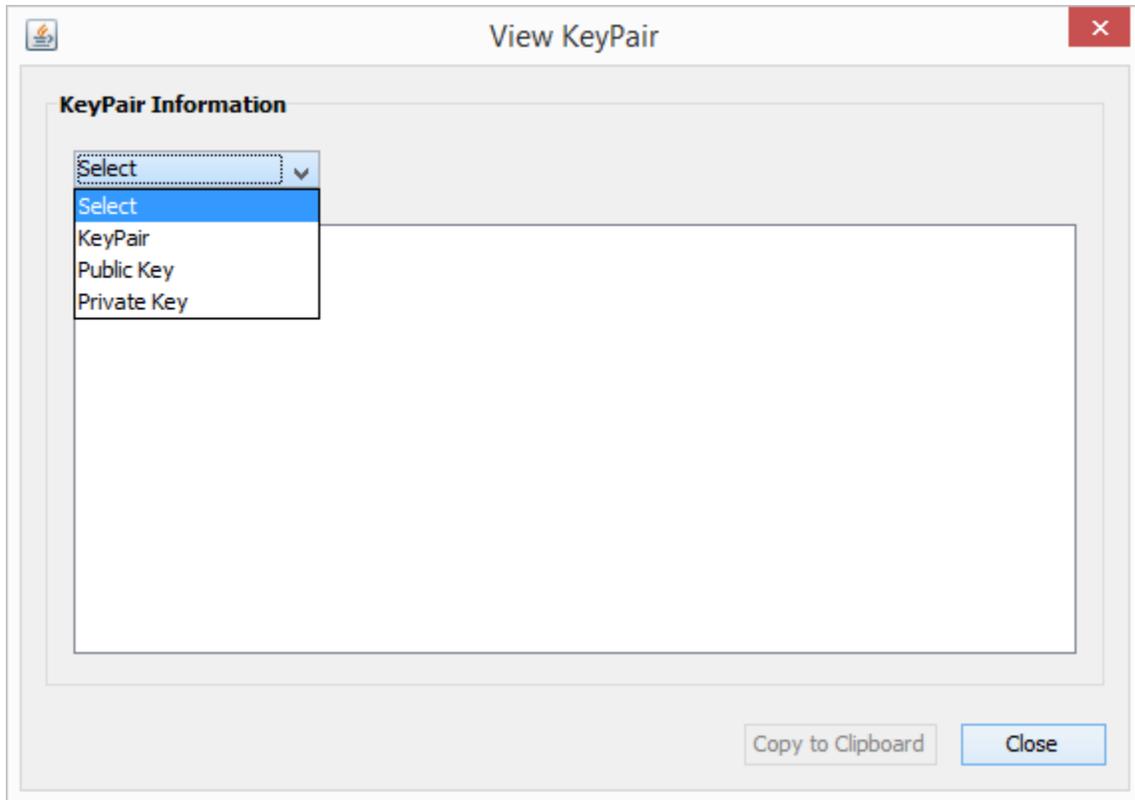
Products can be categorized with creating categories. Unlimited number of categories and subcategories can be created. It is possible to change a product category with *Change Product Category* menu item.

Following window displayed with a list of products and categories, product will be moved to selected category.

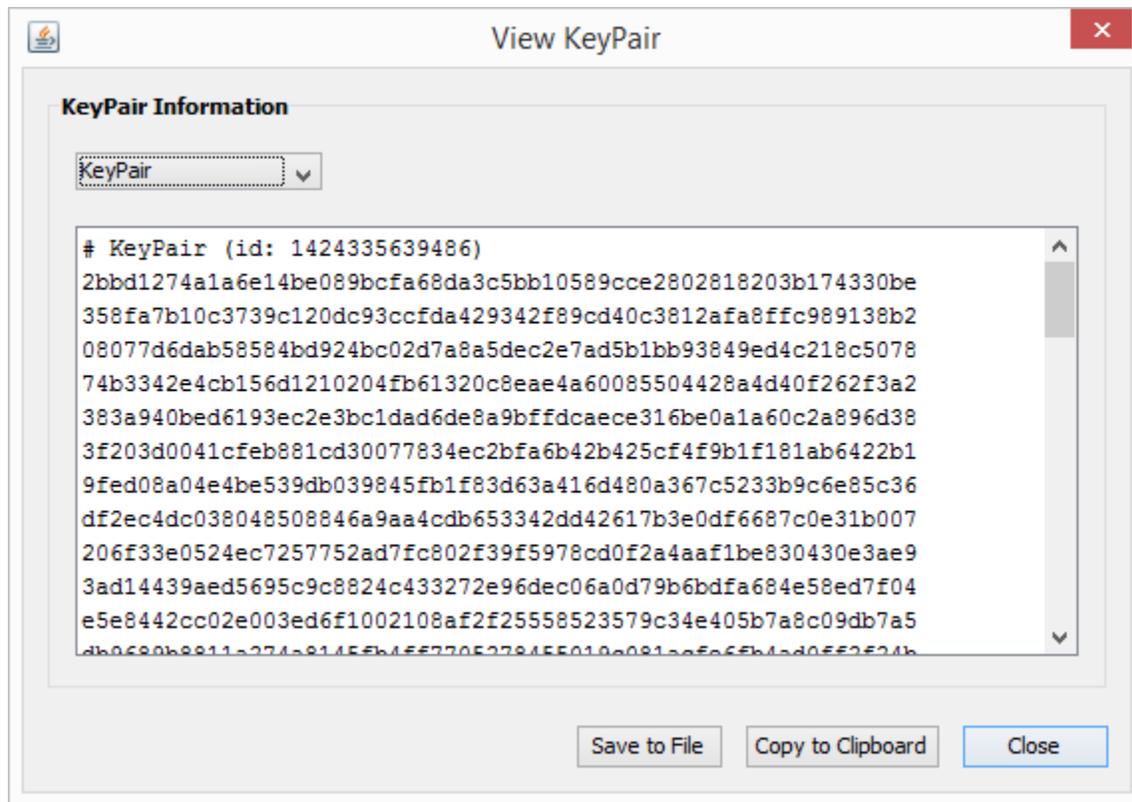


View Key Pair

Key pair, public and private key is displayed with *View KeyPair* menu item. Public key is used in license validation. Key pair and private key is displayed for informational purposes; they are not used in any part of license validation, but they are used in license generation methods.

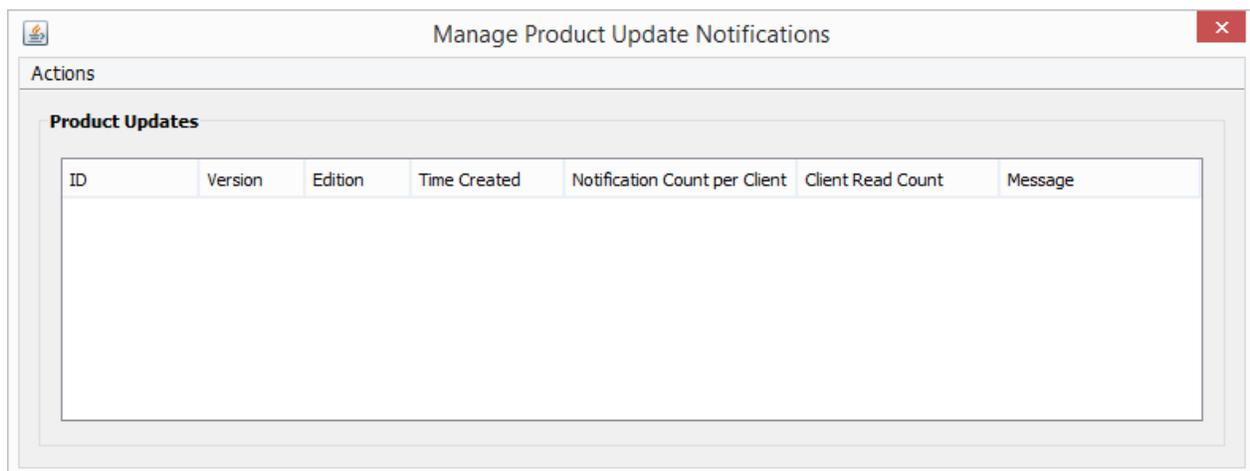


When *KeyPair* is selected from list, *Save to File* button is also displayed. Saved KeyPair file should kept in a secure place; because it includes the private key for the product. This option can be used for backup purposes or to create more than one products with same public key; so that same public key can be used to validate licenses for more than one product.

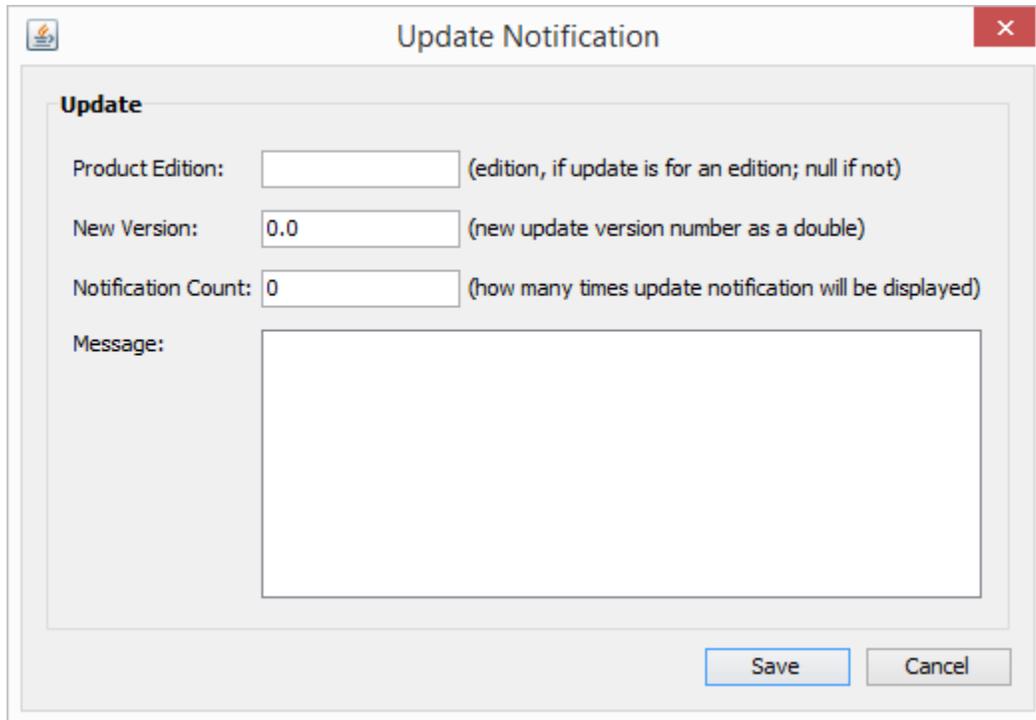


Product Update Notifications

Product update notification can be added for each product with edition, new version number and optional update message. In the update management window as in the screenshot below, update notifications are managed.



Following is the window to create a new update version. Notification count is the number of times which update notification will be displayed to user; display count is increased and stored for each different user computer.



The screenshot shows a dialog box titled "Update Notification" with a close button in the top right corner. The dialog contains a section titled "Update" with the following fields:

- Product Edition: (edition, if update is for an edition; null if not)
- New Version: (new update version number as a double)
- Notification Count: (how many times update notification will be displayed)
- Message:

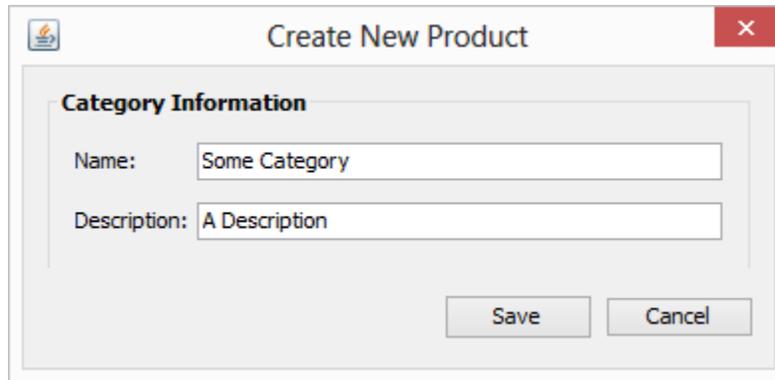
At the bottom right of the dialog are two buttons: "Save" and "Cancel".

Managing Categories

Categories is a way for grouping products on the left tree in license manager. They are just like standard folders.

Create, Edit Categories

Category name is a required field. Unlimited number of categories and subcategories can be created.



Create New Product

Category Information

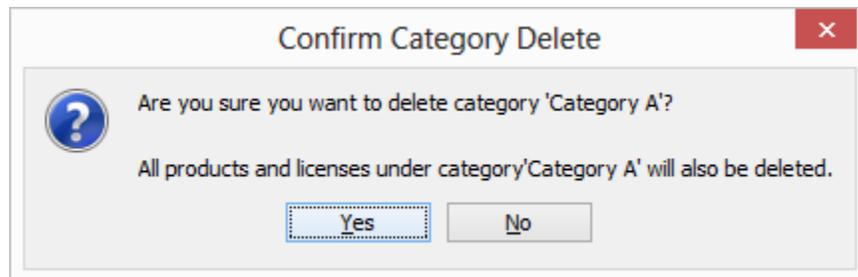
Name:

Description:

Refresh, Delete Categories

When a category is refreshed, subcategories, products and licenses for that products are all refreshed from database.

If a category is deleted, all subcategories, products and licenses also deleted recursively.



Confirm Category Delete

Are you sure you want to delete category 'Category A'?

All products and licenses under category 'Category A' will also be deleted.

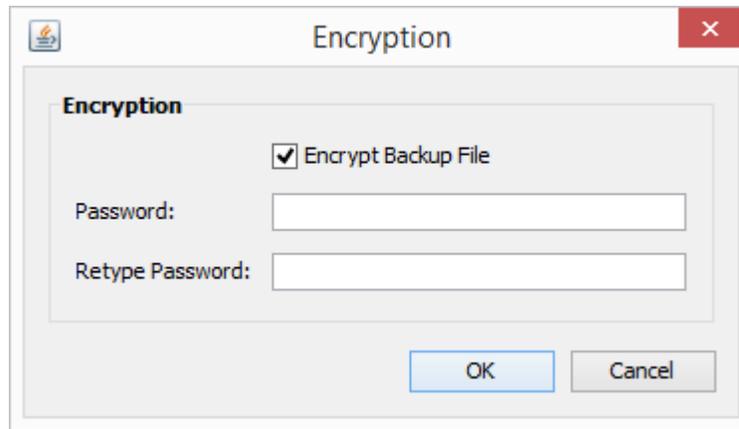
Backup and Restore Products

Products menu has *Backup Product* and *Restore Product* menu items to backup and restore all features of products.

Backup operation copies all settings of product, license templates defined, auto license generation settings, generated licenses and activations to an encrypted text file. Password for encryption is defined before performing backup.

Backup can be restored to same or any other local database storage or to Online.License4J system if same product does not already exists.

Backup file is encrypted with a password when checkbox is selected on the dialog window as in the screenshot below. Encryption may take long depending on product data size (number of licenses, activations etc.)



Backup/Restore can be also used to transfer some products with all their settings and licenses to different database storage or to/from Online.License4J system. If product includes licenses and or templates with activation feature enabled; and license generation settings; and license keys floating over internet, it cannot be restored to a local embedded derby database.

Managing Licenses

Licenses are generated for products in license manager, so a product must be selected on left tree to enable license menu items.

Five types of license are supported;

1. License Text

License text normally used as a license file, it is an encrypted text of defined properties. The first line includes product name and a unique license id for information. It can store product and license information, and also unlimited number of custom features.

Sample license text:

```
# My Product License (id: 1365675659120)
25f2ca4d8070e4a7baef4b7dd5549ba520dbb305d222dd9c18cbee490709
4cef78e28fade0e235f6011182c6764a1d5344b0605a2fd2b7769f47fb75
b7badc745e56870b04b1e658ddb6661fa53e3f53ae515af5335d724db28
cfe13f9225a4a9f893fd0493b683c7b0df9b5b547ae0364fa82059dda7af
6ef448efa121974bf1b8eb26beb795a2b77339236a9360ccb350afef372e
afaa2dc1cc60e998284f41b2cebacc295d840c3fcd1a6d0a0a698027cb91c
27248d477bcb9929878bd66c040216c1a2c12a79af3fc47eabddd279da33
ca9c58421d6de12608ef47ed5aa1af304998fc9211823b69e83d991c0d2d
5a804b5d39a2aeec8046e72c2cd93ca8fa01e643bf6ea32753d24e67d38e
60af3bd6842092fddfffd2c32383d6ed499d0aaed60c0dde2d28afb22e85
23c8296f850dad464fa40c4bc3ba3a80b57c8177f87b4c1ddd90c4f9f39e
1ec5914e68feefd5c376bceae7c98fb68d6f0f646c04d2cb12b77948b1f7
b52cdb1954eefc9ca3fb43a1f61dd4068ec426fa4bfe570cc1306f42ca4b
3174a60d1a568275c5a9f30891cadde7d968
```

Comment blocks can be added before or after the license text between `/**` and `*/` characters like in Java. Line starting with `#` sign should be either not modified or completely removed. The following is also a valid license text format.

```
/** This a comment block before license text */
/** Some text here.
    Some more text here.
*/
25f2ca4d8070e4a7baef4b7dd5549ba520dbb305d222dd9c18cbee490709
4cef78e28fade0e235f6011182c6764a1d5344b0605a2fd2b7769f47fb75
b7badc745e56870b04b1e658ddb6661fa53e3f53ae515af5335d724db28
cfe13f9225a4a9f893fd0493b683c7b0df9b5b547ae0364fa82059dda7af
6ef448efa121974bf1b8eb26beb795a2b77339236a9360ccb350afef372e
```

```
afaa2dc1cc60e998284f41b2cebac295d840c3fcd1a6d0a0a698027cb91c
27248d477bcb9929878bd66c040216c1a2c12a79af3fc47eabddd279da33
ca9c58421d6de12608ef47ed5aa1af304998fc9211823b69e83d991c0d2d
5a804b5d39a2aeec8046e72c2cd93ca8fa01e643bf6ea32753d24e67d38e
60af3bd6842092fddfffd2c32383d6ed499d0aaed60c0dde2d28afb22e85
23c8296f850dad464fa40c4bc3ba3a80b57c8177f87b4c1ddd90c4f9f39e
1ec5914e68feefd5c376bceae7c98fb68d6f0f646c04d2cb12b77948b1f7
b52cdb1954eefc9ca3fb43a1f61dd4068ec426fa4bfe570cc1306f42ca4b
3174a60d1a568275c5a9f30891cadde7d968
```

```
/** This a comment block after license text */
/** Some text here.
    Some more text here.
*/
```

2. Floating License Text

Floating licenses can only be installed on License4J Floating License Server and clients lease licenses from license server. It is same as license text with additional floating license features defined internally. It can store product and license information, and also unlimited number of custom features.

License Text file format is same with normal License Text above, comment blocks are also supported.

3. Basic License Key

Basic license key is generated with a reversible algorithm and a hidden keyword. It is not cryptographically secure. It can store hardware ID if license is node-locked, and activation required or completed information with defined activation period in days.

Sample basic license key:

```
AQDS6-UAYJU-66FXV-W9WVL-U6A4P
```

4. Cryptographically Secure License Key

This is a secure license key generated with EC algorithm. Since an encryption algorithm is used, it is a longer key. It is also used as an activation code on Online.License4J. It can store hardware ID if license is node-locked, and activation required or completed information with defined activation period in days.

Sample cryptographically secure License Key:

2UTQF-DSFT5-G9UWC-DM23D-36FA2-KJQ4A-A9FES-6C9EN-2LYLV-WD3AR-HMHSY

5. Online Basic Key Floating Over Internet

This type of license key works like floating license text. Online.License4J and Auto License Generation and Activation Server handle license lease and management. After validation a license text is obtained by Runtime Library and the license text can store many features like normal license text.

Sample basic license key:

CZ3ED-FUK99-EU9S4-9I95P-S3REF

Generate New License

A wizard is used for license generation, each step of the wizard defines various features of a license. Different wizard steps are displayed depending on selected license type.

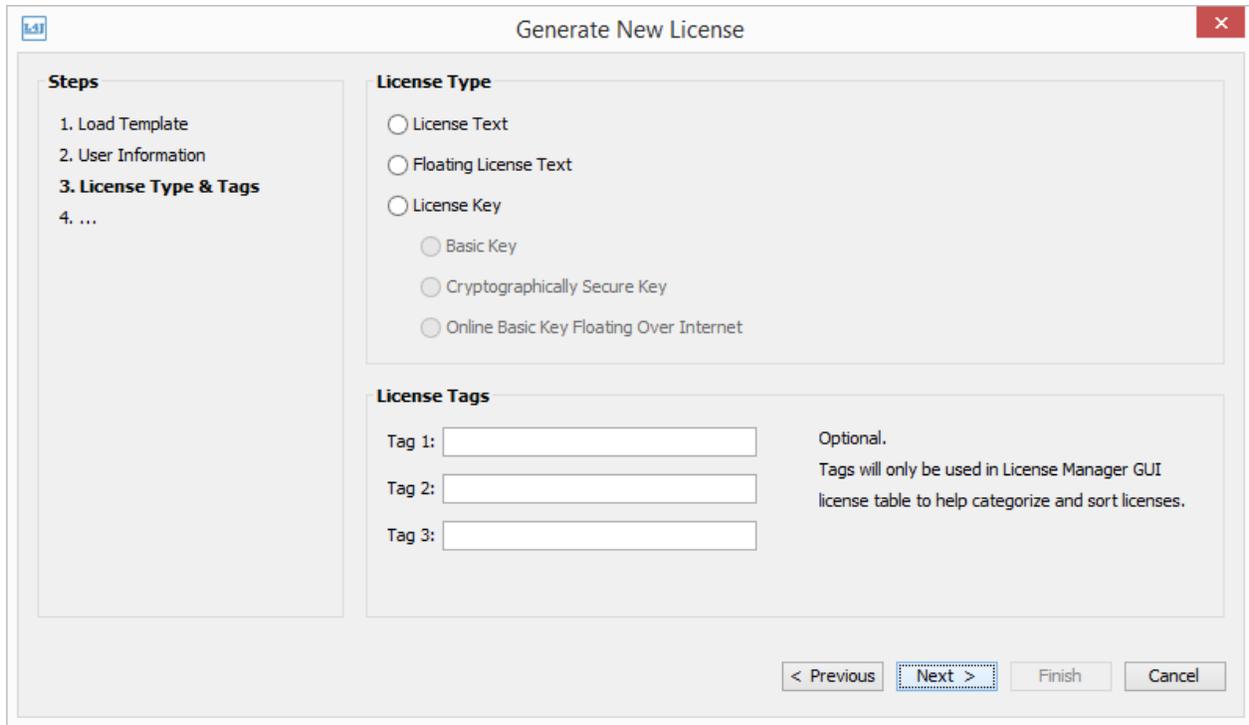
The first step is for template loading, if there is no templates defined for the product, it displays a message.

The screenshot shows a window titled "Generate New License" with a close button (X) in the top right corner. On the left, a "Steps" panel lists: 1. Load Template (bold), 2. User Information, 3. License Type, and 4. ... The main area is titled "Template" and contains a "Select Template:" label followed by a dropdown menu showing "Product does not have a template". At the bottom right, there are four buttons: "< Previous", "Next >", "Finish", and "Cancel".

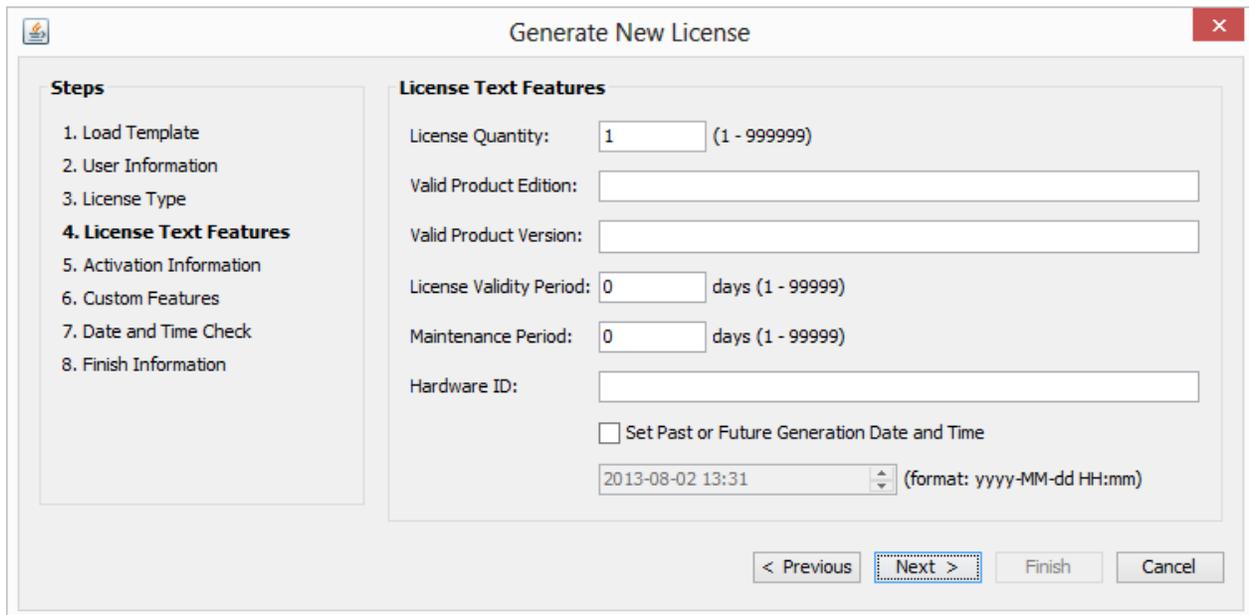
The second step includes user information. User information like name, company, e-mail is defined. This step is same for all license types to save user information.

The screenshot shows the same "Generate New License" window, now at the "User Information" step. The "Steps" panel on the left shows: 1. Load Template, 2. User Information (bold), 3. License Type, and 4. ... The main area is titled "User Information" and contains several input fields: Full Name, Register To, Company, E-Mail, Street, Telephone, City, Fax, Zip Code, and Country. At the bottom right, the "Next >" button is highlighted with a dashed border, indicating it is the active step.

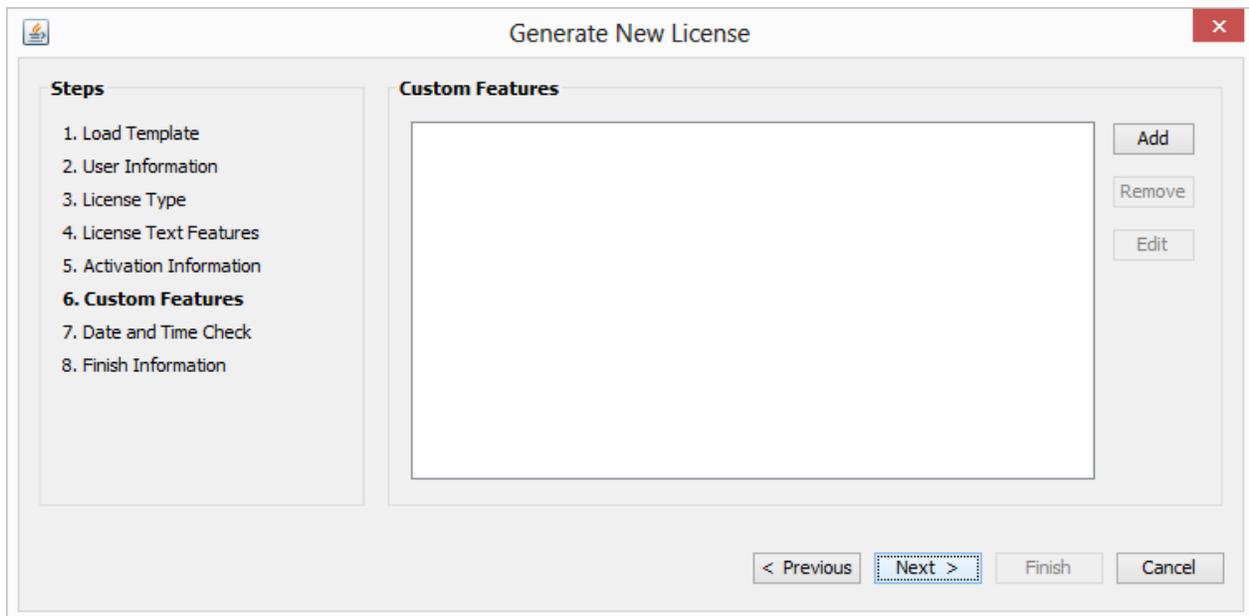
License type selection is on the third step. After selecting a license type, wizard steps changes according to selected license type.



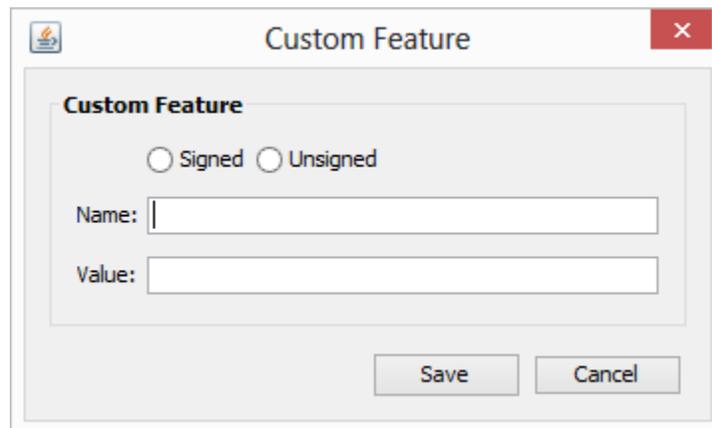
When license text is selected on the third step, following fourth step displayed for defining license text features.



The next step of license text generation is to define some custom signed/unsigned features if required.



Use the buttons on right to add, remove or edit a custom feature. Following is a screenshot of add custom feature window.

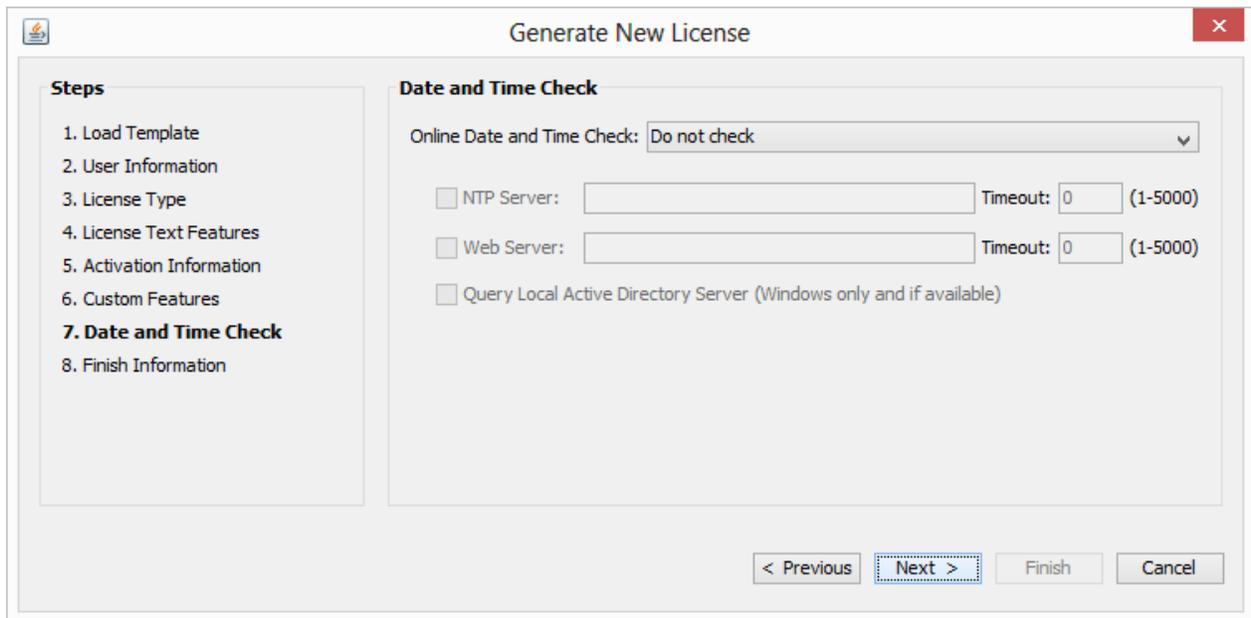


Any information can be stored in key and value pairs in license text, and can be checked/verified after license validation; depending on custom feature value some software features can be enabled or disabled. Maximum allowed number of characters for the value field is 2048.

Signed custom features are used in license validation, so they cannot be modified in license once license is generated. Unsigned custom features can be modified since they are not used in license validation.

In the next step online date/time check settings defined. If it is selected to run on each run, defined NTP server, web server, and if applicable active directory server is queried in order to get the time, so date/time changing on customers' computer can be detected. If an online date/time source is queried and the difference is more than 24 hours with customers' local computer time, license validation status is set as *INCORRECT_SYSTEM_TIME*.

Web server link must return a single line of time since epoch like 1365676045742. It should be text/plain and only contain the number.



The screenshot shows a dialog box titled "Generate New License" with a close button (X) in the top right corner. On the left, a "Steps" list contains eight items, with "7. Date and Time Check" highlighted in bold. The main area is titled "Date and Time Check" and contains the following controls:

- "Online Date and Time Check:" dropdown menu with "Do not check" selected.
- Checkbox for "NTP Server:" followed by a text input field and a "Timeout: 0 (1-5000)" label.
- Checkbox for "Web Server:" followed by a text input field and a "Timeout: 0 (1-5000)" label.
- Checkbox for "Query Local Active Directory Server (Windows only and if available)".

At the bottom right, there are four buttons: "< Previous", "Next >" (which is highlighted with a dashed border), "Finish", and "Cancel".

And the last step for license file generation displays defined features for the license, after clicking on finish button license is generated.

Steps

1. Load Template
2. User Information
3. License Type
4. License Text Features
5. Activation Information
6. Custom Features
7. Date and Time Check
- 8. Finish Information**

License Information

License Information	
ID	1374668077570
Generation Date Time	2013-07-24 15:14:37
License Type	License Text
Quantity	1
Valid Product Edition	Professional
Valid Product Version	1.0
License Validity Period	0

< Previous Next > Finish Cancel

If Floating License is selected on the third step, one additional step is added to wizard and as seen on screenshot below floating license features are defined. Floating license check performed in defined license check period times in minutes. Floating license server check performed if defined in related field. The last allow multiple instances feature defines whether it will be allowed a user to run multiple copies of software on the same host computer. This feature can be used to prevent customers using a single product license on a remote computer by running many copies with same user id. When it is allowed to run multiple instances, a limit can be defined for instances.

Superseded Floating License IDs field defines superseded floating license IDs by this new license. This feature is used to replace floating licenses on customer's servers. If it is needed to disable a floating license usage and replace with a new one due to adding or removing some new features to software product, license IDs should be defined in this field while generating new license. More than one license ID can be defined separated with space character. When a new floating license is generated and installed on floating license server with some superseded license IDs, all defined license IDs are searched in installed license list and when found they are disabled.

It is possible to supersede all older floating licenses for a specific product depending on search selection. Product ID search is always performed, then Edition and Version fields are searched if selected. License issue date values are compared and older licenses are superseded with this setting.

If the license superseding some other licenses are removed from floating license server, superseded licenses are again enabled and can be leased. Since issued and installed licenses cannot be modified on customer's floating license server, license superseding is the only method for floating license modification and/or replacement.

License supersede feature is first added to version 4.4.0. If any license with supersede definition is installed on a floating license server older than version 4.4.0, it will not supersede defined licenses because older versions does not have this feature. In order to prevent use of superseded licenses on old floating license servers, floating license server version can be queried with runtime library on product side. *LicenseValidator.getFloatingLicenseServerVersion* method sends a request to server and returns version number. It returns 0 (zero) for all floating license server versions older than 4.4.0. If server version is 4.4.0+, it returns version number as integer (e.g. 440). If new license supersede feature is used, it is recommended to check version number before floating license validation. *License.getFloatingLicenseServerVersion* method can also be used after license validation to check for server version. If version number is not 440+, leased license should be released with *License.releaseFloatingLicense* method.

Allow OverUse Percentage defines percentage of license quantity which are allowed to be used. So number of clients can exceed license quantity and it can be monitored on administration GUI, graphical reports and floating license server query runtime results. The clients which obtained overused licenses will get validation status of *FLOATING_LICENSE_OVERUSED* instead of *LICENSE_VALID*, therefore it is possible to notify user that license usage exceeds license quantity. Usually,

developer allows software product to run when *FLOATING_LICENSE_OVERUSED* returned; and only notify users to buy more licenses.

With *Allowed IP Blocks* field, a license can be limited to specific IP blocks; if requesting client IP address matches, it gets the license; all other clients are rejected. This settings is saved into license text so that it cannot be modified by users.

The screenshot shows the 'Generate New License' wizard window. The title bar reads 'Generate New License' with a close button (X) on the right. On the left, a 'Steps' sidebar lists eight steps: 1. Load Template, 2. User Information, 3. License Type & Tags, 4. License Text Features, 5. Floating License Features (highlighted), 6. Custom Features, 7. Date and Time Check, and 8. Finish Information. The main area is titled 'Floating License Features' and contains the following controls:

- License Check Period: 0 minutes (0 - 1440)
- License Server Connection Check Period: 0 minutes (0 - 1440)
- Allowed OverUse Percentage: 0 percent (0 - 10000)
- Allow Multiple Instances for Same User on Same Host
 - Maximum Number of Allowed Instances: 0 (0 - 10000, zero means unlimited)
- Superseded Floating License IDs: [text input] (separated with space)
- Supersede All Older Floating Licenses:
 - With Same Product ID
 - With Same Edition
 - With Same Version
- Allowed IP blocks (xxx.xxx.xxx.xxx/xx): [text input] (separated with space)

At the bottom right, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

If basic license key is selected on the third step, wizard steps changes as in the following screenshot. Only license key features can be defined. Internal hidden string is used in basic license key generation, if it is not defined unique product id is used, so it must be used in license validation method. Customer full name and company name can be used in license validation if related checkbox items are selected.

The screenshot shows the 'Generate New License' dialog box with the 'License Key Features' step selected. The 'Steps' list on the left includes: 1. Load Template, 2. User Information, 3. License Type, 4. License Key Features (highlighted), and 5. Finish Information. The 'License Key Features' section contains the following fields and options:

- License Quantity: (1 - 99999)
- Hardware ID:
- Internal Hidden String:
- Use Customer Name in Validation
- Use Company Name in Validation

Navigation buttons at the bottom are: < Previous, Next >, Finish, and Cancel.

There is no difference in wizard steps if cryptographically secure license key is selected in third step. The only difference is that internal hidden string is not required in license key generation since a cryptographic algorithm is used.

If "Online Basic Key Floating Over Internet" license type is selected on the third step, following step is displayed.

The screenshot shows the 'Generate New License' dialog box with the 'Online License Key Features' step selected. The 'Steps' list on the left includes: 1. Load Template, 2. User Information, 3. License Type & Tags, 4. Online License Key Features (highlighted), 5. License Text Features, 6. Custom Features, 7. Date and Time Check, and 8. Finish Information. The 'Online License Key Features' section contains the following fields and options:

- License Concurrent Quantity: (1 - 999999)
- Allowed Usage Count: (0 - 999999)
- Allowed Usage Time Limit: minutes (0 - 999999)
- Max Inactive Period: minutes (1 - 1440)
- Maximum Re-Checks Before Drop: (0 - 10)
- Don't Keep Released License Usage Records
- Allow Multiple Instances on Same Host
- Define a Special Key: (min 5, max 255 chars)

Navigation buttons at the bottom are: < Previous, Next >, Finish, and Cancel.

In this step, online basic key settings are defined. License concurrent quantity defines maximum concurrent license usage count. Allowed Usage Count defines maximum allowed usage. With allowed usage count, it is possible to limit how many times a software is allowed to run.

Max Inactive Period defines maximum inactivity time in minutes for a license. License4J Runtime Library periodically connects in defined intervals to Auto License Generation and Activation Server or Online.License4J system to acknowledge its license usage. If for some reason server does not get acknowledge for more than defined inactivate period time, it drops license usage. *Maximum Re-Checks Before Drop* setting defines maximum number of failed acknowledge checks for a license.

If Don't Keep Released License Usage Records checkbox is selected, historical online license key usage records are not stored; records are deleted when they are released. By default server keeps all released online key usage records.

Special key field allows to define the key, it should be between 5 and 255 characters long. This field is optional, if it is left blank, 25 characters long license key is generated.

Allow Multiple Instances on Same Host setting defines the behavior for running multiple instances on a computer. If unchecked, each instance will request a license from server so will increase license concurrent usage. If it is checked, then users can run any number of instances on the same host without increasing concurrent usage count.

Activation step is displayed as below for both online and database storage licenses.

The screenshot shows the 'Generate New License' dialog box with the 'Activation' step selected. The 'Steps' sidebar on the left lists: 1. Load Template, 2. User Information, 3. License Type, 4. License Key Features, 5. Activation Information (highlighted), and 6. Finish Information. The 'Activation' section includes the following options:

- Activation Required
- Activation Period: 30 days
- Allowed Activation Count: 1 (1 - 999999)
- Allowed Deactivation Count: 3 (0 - 999999)
- Activation Return: License Text
- Reject License Deactivation
- Don't Keep Deactivation Records
- Reject Modification Key Usage
- Set Activated License Text Generation Time to Activation Time
- Offset From Activation Time: 0 hours (-720 - 0)
- Hardware ID Selection:
 - Check All
 - Check Any
 - Check Custom Hardware ID
 - 1 - Hostname
 - 2 - Ethernet Address
 - 3 - System Disk Volume UUID
 - 4 - HDD Serial Number

Navigation buttons at the bottom include: < Previous, Next >, Finish, and Cancel.

If license will require activation, it can be defined in this step. As in the screenshot, allowed activation count may be defined different from license quantity thus customer buys one license but it may activate license on his/her two computers like work and home if allowed. Deactivation records are also kept if *Don't Keep Deactivation Records* check box is unselected. When maximum allowed deactivation count is defined, user can deactivate and reactivate (to move license to another computer) for defined number of times; after limit is reached activation status returns an error. *Reject Modification Key Usage* checkbox controls modification key usage for this license, if it is selected this license cannot be modified with any modification key usage.

While generating a license template, allowed activation count field is replaced with another field named as coefficient number as in the screenshot below. When a coefficient number greater than 1 is defined, licenses generated with this template will have allowed activation number set to license quantity multiplied with coefficient number.

Coefficient number may be used for some licensing methods which requires more than one activations for each generated license.

Activation period starts from license generation date, NOT license installation date on customers' computers, because license installation date can easily be changed.

Activation return types are activation code and license text. If license text is selected and license type on the third step is selected as a license key one additional license file features step is added and features which will be added to activated license can be defined. Activation code return type generates a cryptographically secure license key to be used as activation code.

Set Activated License Generation Time to Activation Time checkbox is enabled only when selected activation return is a license text. It is also available on license template generation wizard. When this checkbox is selected, activated license text which will be delivered to customer will have generation time set to same value with

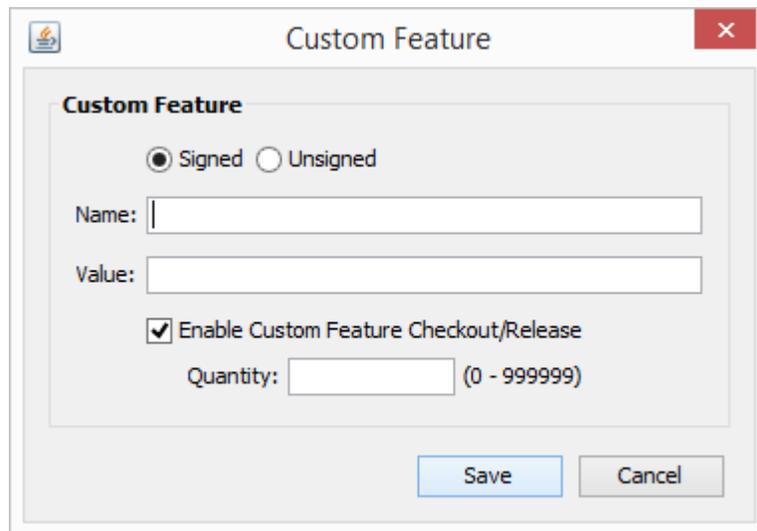
activation time. This feature is useful when a single trial license to be generated; a single trial license is generated and distributed in software or web page to all customers, and when customer activates license its validity period starts, and works for defined limited validity period. Also sometimes, it may be required to generate a license which validity period starts after activation. *Offset From Activation Time* setting defines some negative offset value from activation time. When it is defined activated license text generation time is generated with calculated offset. This is useful if users' local computer time is not synchronized with a time source on Internet; if user's local computer time is ahead real time, activated license text will be invalid until local computer time is ahead activation time.

Activated license is locked to customers' computer with a hardware id. Hardware IDs can be combined with AND and OR combination. It can be defined to lock license to a computer by using its hostname, Ethernet mac address, disk volume serial number (UUID) and disk manufacturer's serial number. Check All checkbox combines selected hardware id types with AND, Check Any checkbox combines selected hardware id types with OR.

Custom hardware ID can be used and enabled with "Check Custom Hardware ID" checkbox. When this checkbox is selected activation server will use the custom hardware ID supplied with `autoActivateWithCustomHardwareID` method.

Activation deactivation is disabled by default. If you need deactivation feature, it should be enabled in this step. If you enable deactivation and allow your customers to deactivate within your product, keep in mind that you must delete activated license text or key from customer's computer yourself. The file you kept license information should also be hidden to your customer and cannot be easily found and copied. If customer can find and copy license file, they can use that file after activation.

In version 4.5.4+ custom features step is different for floating license text type license. There is a new feature in floating license text to count custom features usage. When “Add” button is clicked to add a new custom feature, following window is displayed.



The screenshot shows a dialog box titled "Custom Feature". It contains the following elements:

- Radio buttons for "Signed" (selected) and "Unsigned".
- Text input field for "Name:".
- Text input field for "Value:".
- Checked checkbox for "Enable Custom Feature Checkout/Release".
- Text input field for "Quantity:" with a range "(0 - 999999)".
- "Save" and "Cancel" buttons at the bottom.

Concurrent use quantity can be defined for each signed custom feature, then it can be checked out by `LicenseValidator.checkoutFloatingLicenseTextCustomFeature(String featureName, License licenseObject, String floatingLicenseServer)` method in runtime library. When it is successfully checked out the value of custom feature can be verified with `license.getLicenseText().getCustomSignedFeature(featureName)` method.

It is released with `LicenseValidator.releaseFloatingLicenseTextCustomFeature((String featureName, License licenseObject, String floatingLicenseServer).`

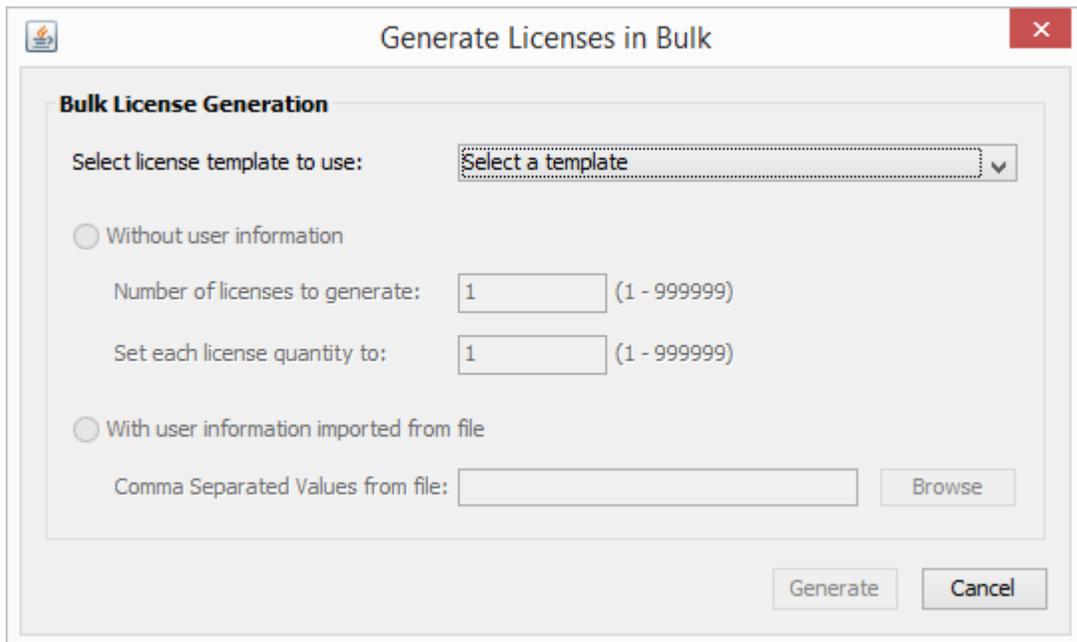
Notice: By default custom features with defined quantity is not enabled after license validation, they must be checked out specifically.

Generate New License in Bulk

Bulk license generation is possible with “Generate New Licenses in Bulk” menu item. The following screenshot is displayed to generate licenses. There are two options for bulk license generation. The first option generates defined number of licenses with empty user information. The second option is used to generate licenses for users imported from a CSV file. In both options, selected license template is used to define license type and features.

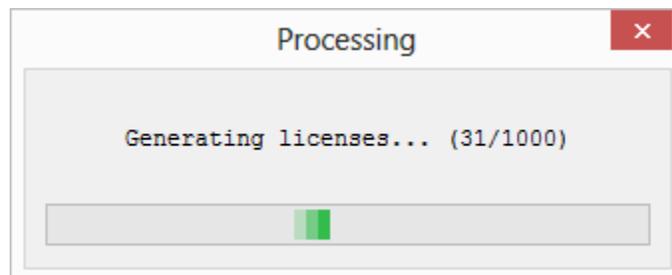
CSV file must be encoded as UTF-8 and include following columns without a header.

Full Name, Registered To, Company, E-Mail, Street, Telephone, City, Fax, Zip Code Country, Quantity



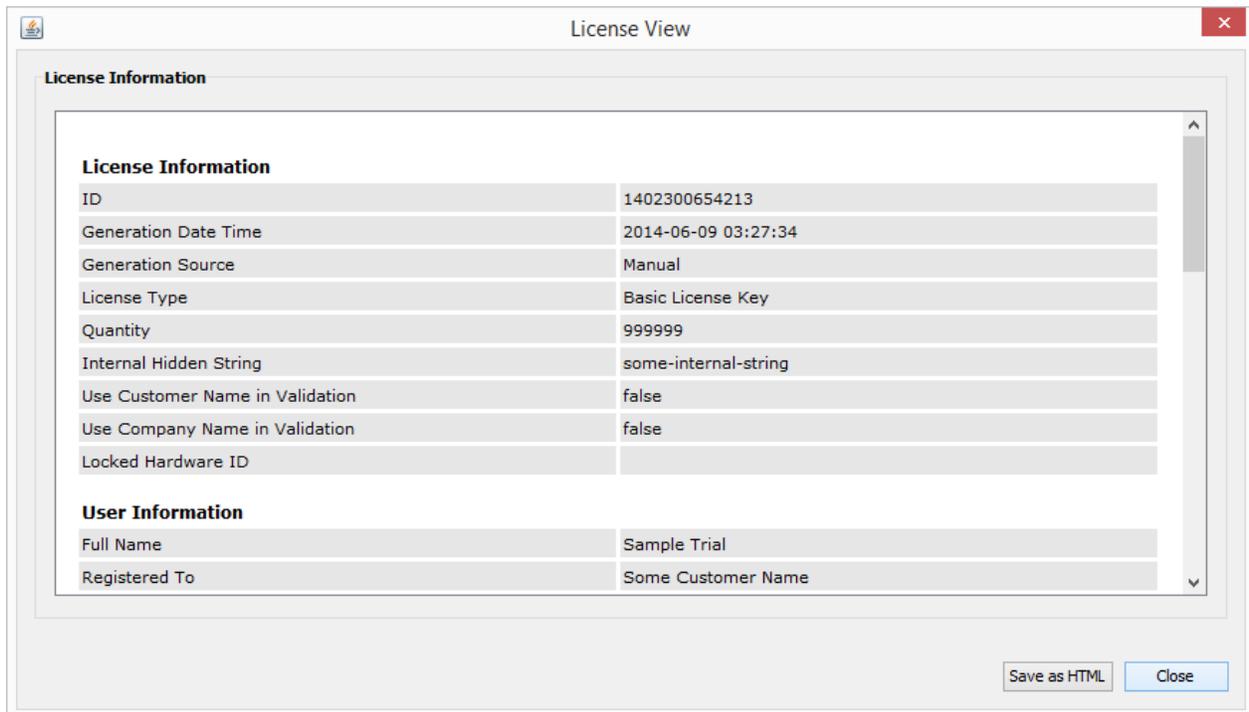
The screenshot shows a dialog box titled "Generate Licenses in Bulk". It contains the following elements:

- Bulk License Generation** (Section Header)
- Select license template to use:** A dropdown menu with "Select a template" selected.
- Without user information**
 - Number of licenses to generate:** A text input field containing "1" with "(1 - 999999)" to its right.
 - Set each license quantity to:** A text input field containing "1" with "(1 - 999999)" to its right.
- With user information imported from file**
 - Comma Separated Values from file:** A text input field followed by a "Browse" button.
- Generate** and **Cancel** buttons at the bottom right.



View License

Generated license features are displayed with *View* menu item. On the bottom, license string which will be sent to customers will also be displayed. License information as displayed on this window can be saved to an HTML file with “Save as HTML” button.



Modify License

There are two different modify menu items in *License* menu. Full modify option is used to modify all license features but this modification also changes previously generated license key or text. So that license string should be sent to users again. Minimum modify option modified license features which are not directly used in license key/text generation. The features which cannot be modified will be disabled on minimum license modification wizard. Most of the features of license keys can be modified including quantity, because there are only few features used in license key generation. Only activation features can be modified for license text. Floating license text cannot be modified.

Clone License

Cloning license is available through *Clone* menu item. The generate license wizard comes with all features defined as in the source license. They can be used as is, or can be changed. A new license with a different license ID will be generated, and the generated license string will be different.

Delete License

Licenses can be deleted with delete menu item. If deleted license has activations, all activations are also deleted. License availability on server can be checked with *LicenseValidator.checkOnlineAvailability* method. When you want to blacklist a license and stop future usage on customer computers, you can just delete or move license to recycle-bin use *checkOnlineAvailability* method in your software product.

Recycle Bin folder is only displayed when "View Recycle Bin" menu item is selected on the product menu. License menu and right click pop up menu includes a menu item as "Delete (move to recycle bin)", when clicked selected license(s) are moved to recycle bin. To permanently delete license(s), "Delete" menu item should be used.

Export License

Licenses selected on the license manager are exported to text files with *Export* menu item.

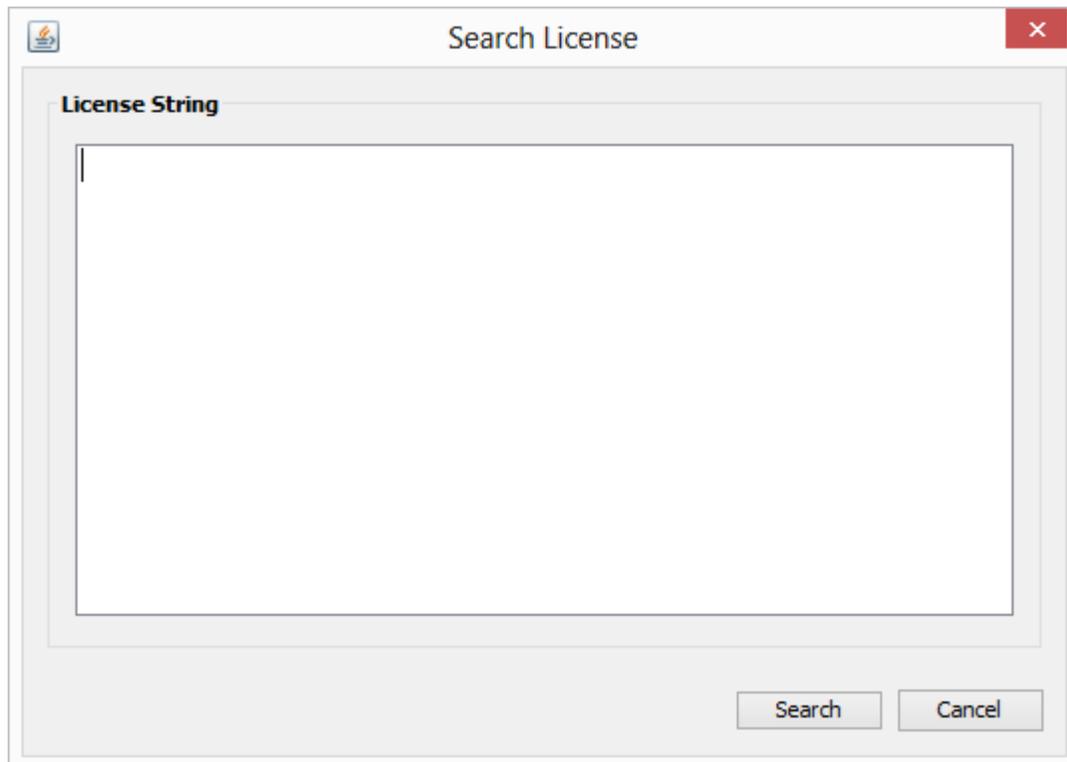
Export Licenses to CSV File

All product licenses or only selected licenses can be exported to a CSV file with *Export Licenses to CSV File* menu item. All license features in the database will be exported.

Selected licenses can also be exported to Excel (xlsx) file with current view setup. Excel file will include only the active columns on License Manager.

Search License

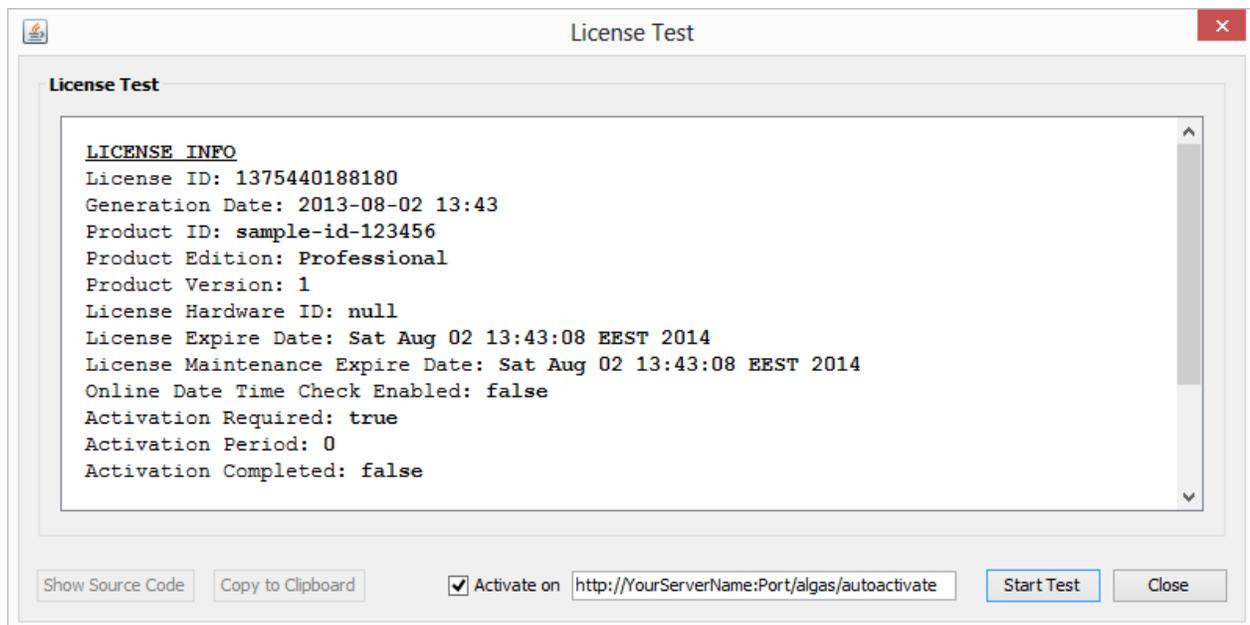
A License key or text can be searched with *Search* menu item in License menu. Single license key or any license text is searched for selected product and if found license view window displays the found license information; found license is auto selected on license table. It also searches for activation code and activated license, if given license key is an activation code or given license text is license activation it displays activation view window after finding the license and displaying activation management window. When a modification key is searched, modification key management window is opened and found modification key is selected then displayed.



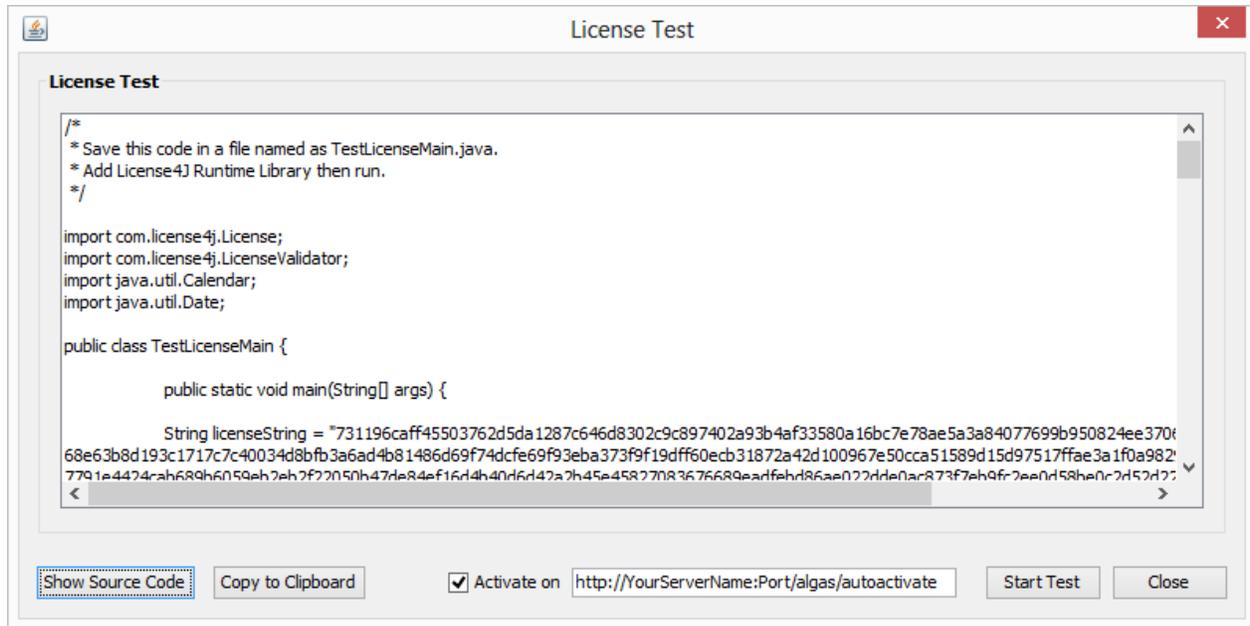
Test License

Generated license keys and license text can be checked within License Manager GUI, after test completes, source code is displayed and can be copied to run on any Java IDE tool. Test is performed with all available arguments of *LicenseValidator.validate* method. First test is performed with correct arguments to test that license is valid. Other tests are performed with incorrect variables to demonstrate and test license features. License and maintenance expiration features are tested with incorrect local time if online date/time check is not enabled.

Test method also performs activation on defined host (either Online.License4J or own Auto License Generation and Activation Server) if license activation is enabled. Activation test can be disabled with the checkbox, and also host can be defined if license is stored on your database and Auto License Generation and Activation Server is installed. After activation, returned license text features are displayed. Do not forget to delete test activation on Manage Activations window.



After test completed *Show Source Code* displays the source code for performed tests.



License Deactivation

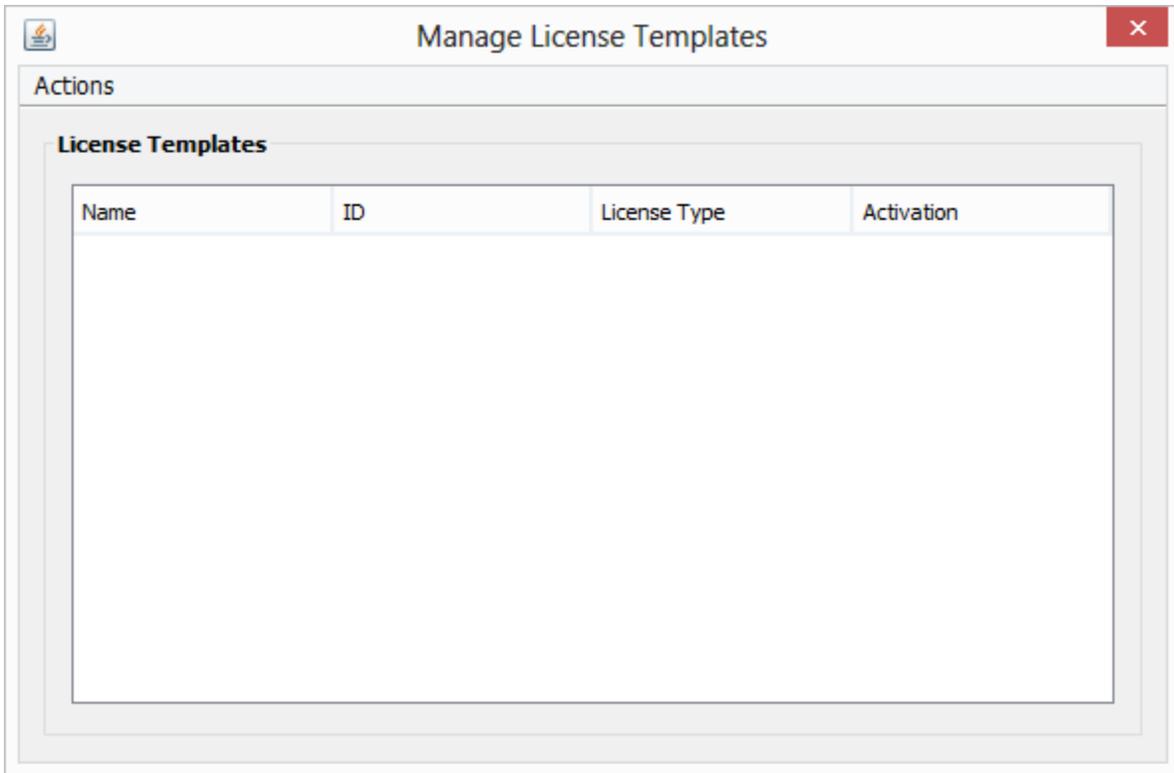
By default license deactivation is disabled. It can be enabled during license generation in activation step or later in manage activations window. Manage activations window actions menu has a menu item called “Disable Auto Activations”, a notification is displayed on the window if it is disabled.

Deactivations is still stored if it is selected in license generation wizard; and maximum allowed deactivation count is defined in license generation wizard while generating license. When there is a limit on number of deactivations, license can be deactivated and re-activated until limit is reached. Therefore license move operations can be made by users in defined limits. License activations can be deactivated manually and if required reactivated in Manage Activations window.

License deactivation can be either automatic with method *autoDeactivate* in runtime library or manual by submitting a form to license server. For manual deactivation, the string value to submit can be obtained and displayed to users with *license.getManualDeactivationRequestString()*.

Manage License Templates

Any number of license templates can be defined for each product. All license templates are displayed on template management window. Create, edit, delete, view actions are available under *Actions* menu. License template creation is done with the same wizard as license generation.



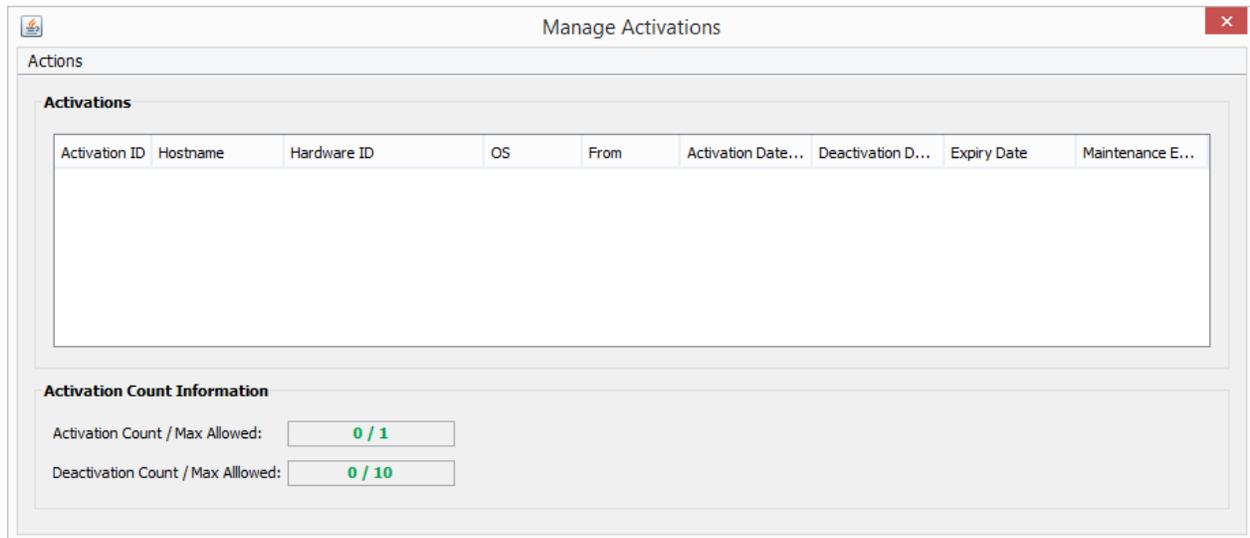
Version 4.5.4+ includes two new menu items in *Actions* menu. The first one is "*Display validate Code*"; it gives license validation code for licenses generated by using the template. The second menu item is for new *easyValidate* method. It gives encrypted string to use in *easyValidate* method.

Manage Activations

License activation is a feature which is enabled when used with Online.License4J storage system or Auto License Generation and Activation Server, so

License Manager must be connected to Online.License4J server or Auto License Generation and Activation Server should be installed and activations must be enabled for the generated license.

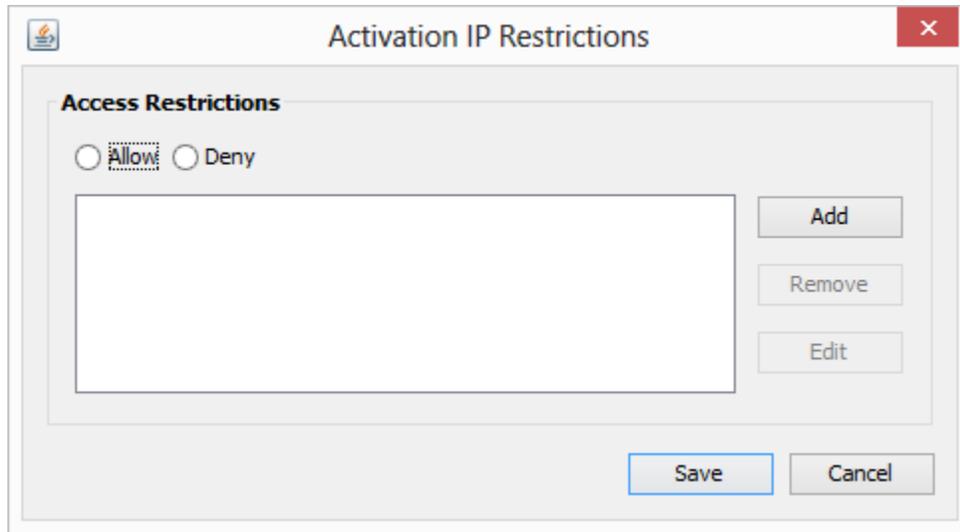
When activation management window opened, all activations for the license is displayed.



On the bottom, activation count and allowed activation count is displayed. Allowed activation count can be modified with *Modify Maximum Allowed Activation Count* menu item in *Actions* menu. Deactivation count and maximum allowed deactivation count is also displayed if defined for license in license generation wizard; maximum allowed deactivation can be modified in *Actions* menu.

Note: Value of 0 (zero) for maximum allowed deactivation count means unlimited.

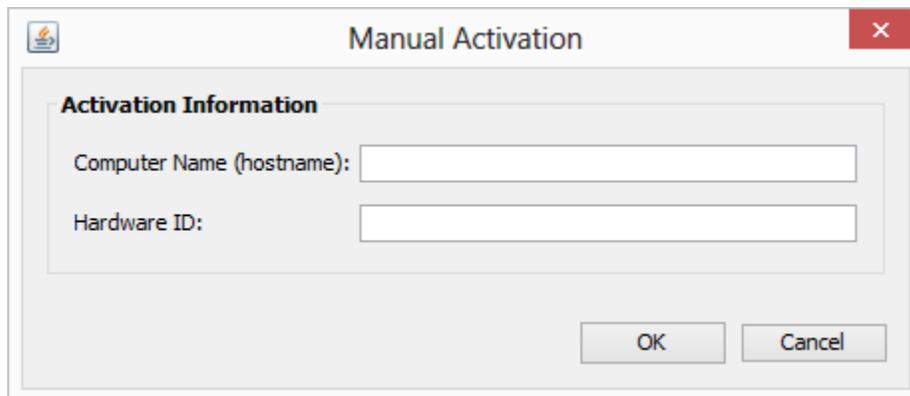
Allowed or Denied IP blocks can be defined with *Modify Allowed/Denied IP Block* menu item. Following window displayed to define IP access restrictions.



Allow or Deny action should be selected first, then desired IP blocks can be defined and modified with buttons on the right of window.

Auto and manual activations for selected license can be temporarily or permanently can be disabled with related menu items in actions menu.

Manual activation is performed with *Add Manual Activation* menu item, as on the following screenshot hostname (for informational only, not used in activation creation), and hardware id should be defined. After adding activation, it can be exported to a file to send to customer.

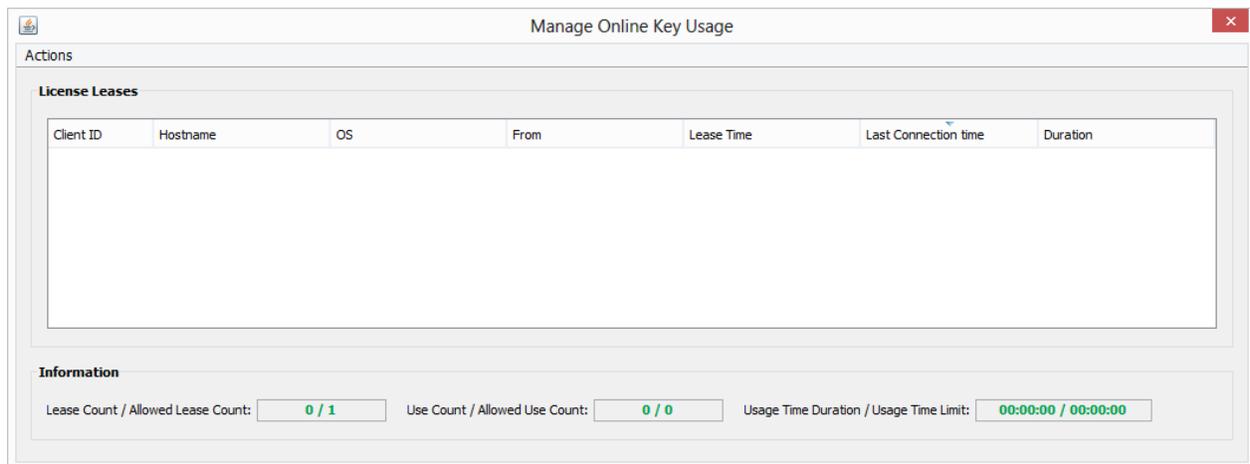


Selected activations with all database fields can be exported to CSV file with related menu item. They can also be exported to Excel (xlsx) file as seen on License Manager Table.

Manage Online Basic Key Usage

Online Basic Key Floating Over Internet is a license type which works when used with Online.License4J storage system or Auto License Generation and Activation Server.

Real time online basic key used can be monitored with *Manage Online Key Usage* menu item in main license menu. Following window is displayed and it is very similar to manage activations window.



Windows displays all usage whether it is active or inactive. Inactive usage is grayed out in table view. There are two options to remove a license lease: Remove menu item just marks usage as deleted and license use becomes available, but Permanently remove menu item completely deletes license usage record from database so license use count and usage time duration values are also updated.

License quantity can be modified with *Modify License Quantity* menu item. All changes are applied in real time. Current license usage is displayed on the bottom of window.

As in activation management, allowed or denied IP blocks can be defined with *Modify Allowed/Denied IP Block* menu item.

License usage can be disabled with “Disable New Validation and Lease” menu item in actions menu. Currently leased licenses will continue to work but new license lease requests will be denied.

If license has a limit for maximum use, current use count and maximum value is displayed at the information panel. Maximum allowed use count can be modified with related menu item in actions menu. If maximum use limit is not defined for license, window still displays current use count but does check for limits.

If maximum usage time limit is defined, usage times and maximum value will be displayed on window. When usage time limit reached, server denies license lease requests; if license is already leased defined handler timer thread (*OnlineLicenseKeyCheckTimerHandler*) is run.

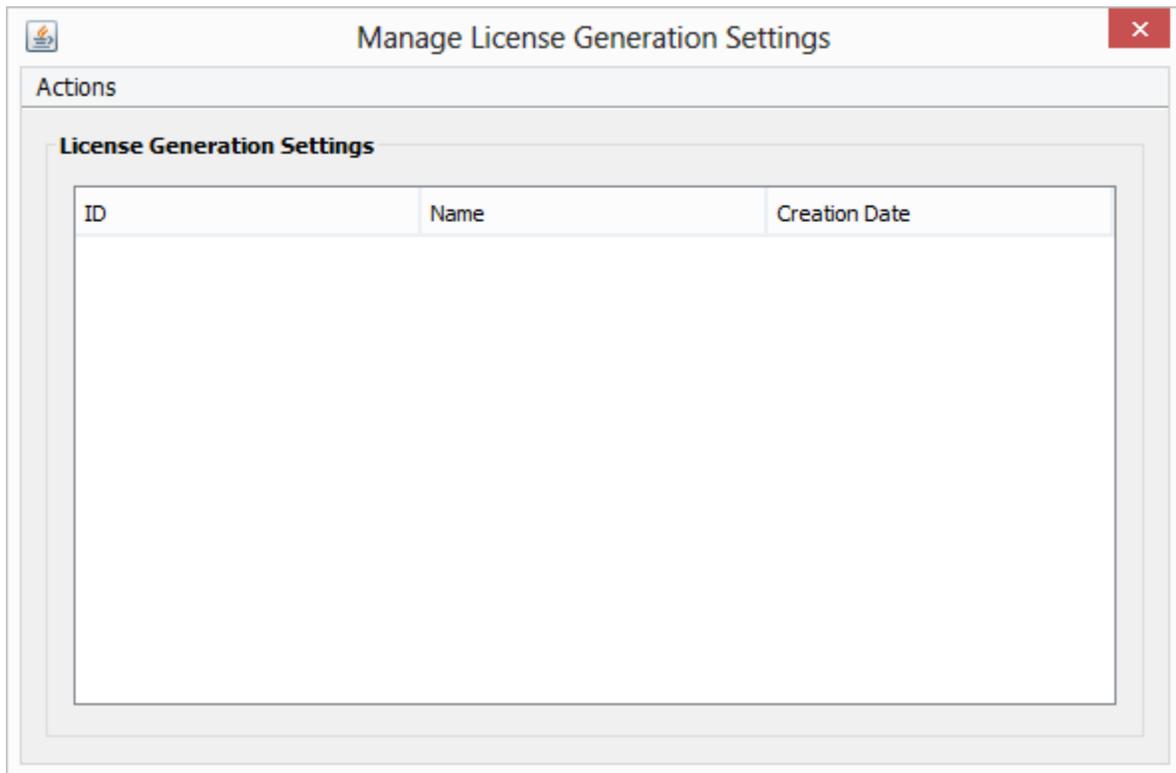
License usage table can be refreshed with related menu item or with F5 key to see current license usage.

Selected online key usages with all database fields can be exported to CSV file with related menu item. They can also be exported to Excel (xlsx) file as seen on License Manager Table.

Automatic License Generation Settings

Automatic license generation is a feature of Online.License4J system and License4J Auto License Generation and Activation Server.

Management window is similar to license template management window. Any number of different auto license generation settings can be defined.



With *Create* menu item, following window displayed.

License Generation Settings

General | Input Settings | Output Settings

Settings

Descriptive Name:

Security Code:

Access Restrictions

Allowed IPs:

Allowed Domains:

(Will be used for Access-Control-Allow-Origin header to accept AJAX requests)
(Space separated list of domains)

Predefined:

- Select
- Share-it (license key)
- Share-it (license text)
- RegNow
- SWREG (license key)
- SWREG (license text)
- FastSpring
- Avangate (key, xml output)

Window includes three tabs for general, input and output settings. Also there are predefined settings for Share-it, RegNow, SWREG, FastSpring, Avangate and PayPro which can be loaded on the bottom of window.

Since auto license generation will be performed by accepting HTTP POST requests, allowed server IPs can be defined on the first screen. Security code is for license generation URL and check on each license generation request. Allowed domains field is used to define domains which are allowed to send Ajax requests. Allowed domains are used in *Access-Control-Allow-Origin* header.

On the input settings tab, POST variable names which any payment processors or remote server is posting are defined. Custom features can also be defined, so it is possible to send many features to use in license generation.

Security code parameter and Security code value is used for some payment processors for extra security, not all payment processors support it. If a custom license generation request sent, it can be also used for security. Generated sample PHP script use this security code protect license generation.

License Validity Period and Maintenance Period is usually defined in license template; but it can also be set during license generation. Values sent while generating license overwrites values if defined in license template.

License Generation Date can be set to generate licenses with any date so license validity period and validation status is affected. Supplied date format must be **yyyy-MM-dd HH:mm**. e.g. 2017-01-30 13:00. If date is not in correct format, server will by default use current system date.

The screenshot shows the 'License Generation Settings' dialog box with the 'Output Settings' tab selected. The 'POST Variables' section contains the following fields:

Registration Name:	<input type="text"/>	E-Mail:	<input type="text"/>
First Name:	<input type="text"/>	Last Name:	<input type="text"/>
Telephone Number:	<input type="text"/>	Fax Number:	<input type="text"/>
Street 1:	<input type="text"/>	Street 2:	<input type="text"/>
City:	<input type="text"/>	State:	<input type="text"/>
Zip Code:	<input type="text"/>	Country:	<input type="text"/>
Company:	<input type="text"/>	<input type="button" value="Custom Features"/>	
<hr/>			
License Quantity:	<input type="text"/>	Hardware ID:	<input type="text"/>
License Validity Period:	<input type="text"/>	Maintenance Period:	<input type="text"/>
License Generation Date:	<input type="text"/>		
<hr/>			
Security Code Parameter:	<input type="text"/>	Security Code Value:	<input type="text"/>

At the bottom of the dialog, there is a 'Predefined:' dropdown menu with 'Select' as the current selection, and four buttons: 'Load', 'Clear', 'Save', and 'Cancel'.

On the output settings tab, HTTP response type, disposition content before and after license string can be defined to support any payment processor, or any custom license generation requests.

The screenshot shows a dialog box titled "License Generation Settings" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Input Settings", and "Output Settings", with "Output Settings" currently selected. The "Output Settings" tab contains two main sections:

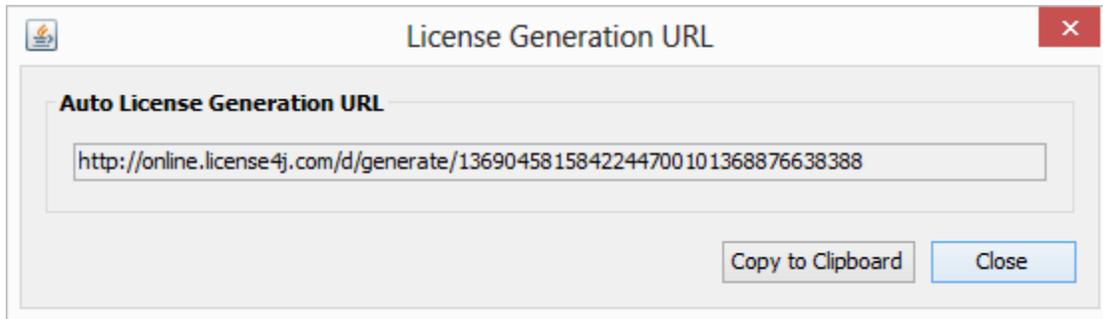
- HTTP Response Settings:** This section includes two text input fields: "Content Type:" and "Content Disposition:".
- Returned License Content Settings:** This section includes two large text area inputs: "Content before license:" and "Content after license:".

At the bottom of the dialog, there is a "Predefined:" dropdown menu with "Select" as the current value, and four buttons: "Load", "Clear", "Save", and "Cancel". The "Save" button is highlighted with a blue border.

License generation can be temporarily or permanently disabled with "Disable License Generation" menu item.

Display License Generation URL

License generation URL is only displayed with the selection of license template, a sample URL follows.



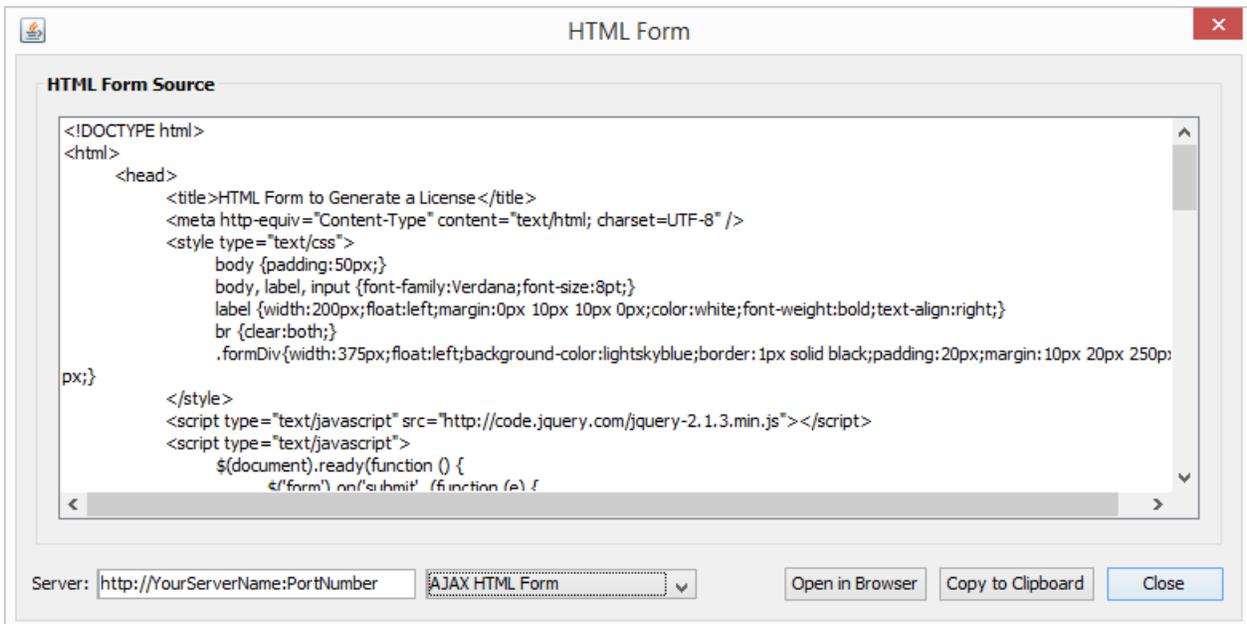
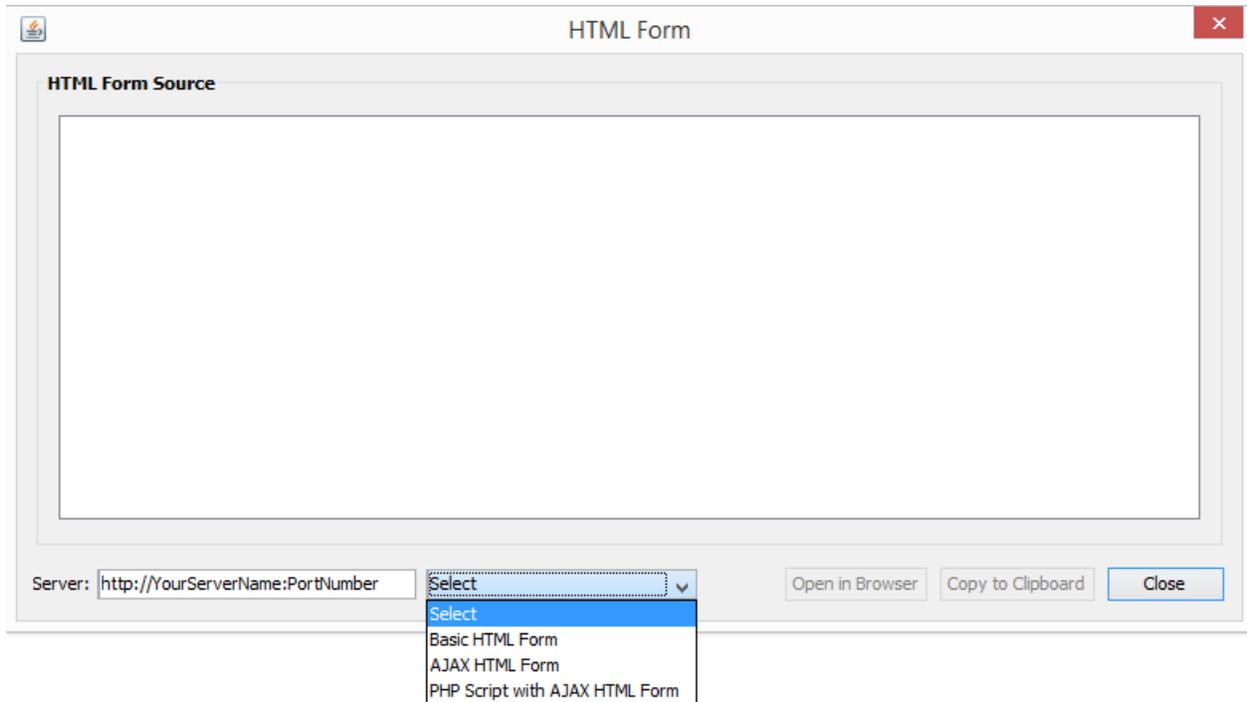
This URL is used to post license generation request with data in post variables. HTTP or HTTPS can be used; HTTPS may be unsupported or slow in some cases.

Display HTML Form Source for License Generation

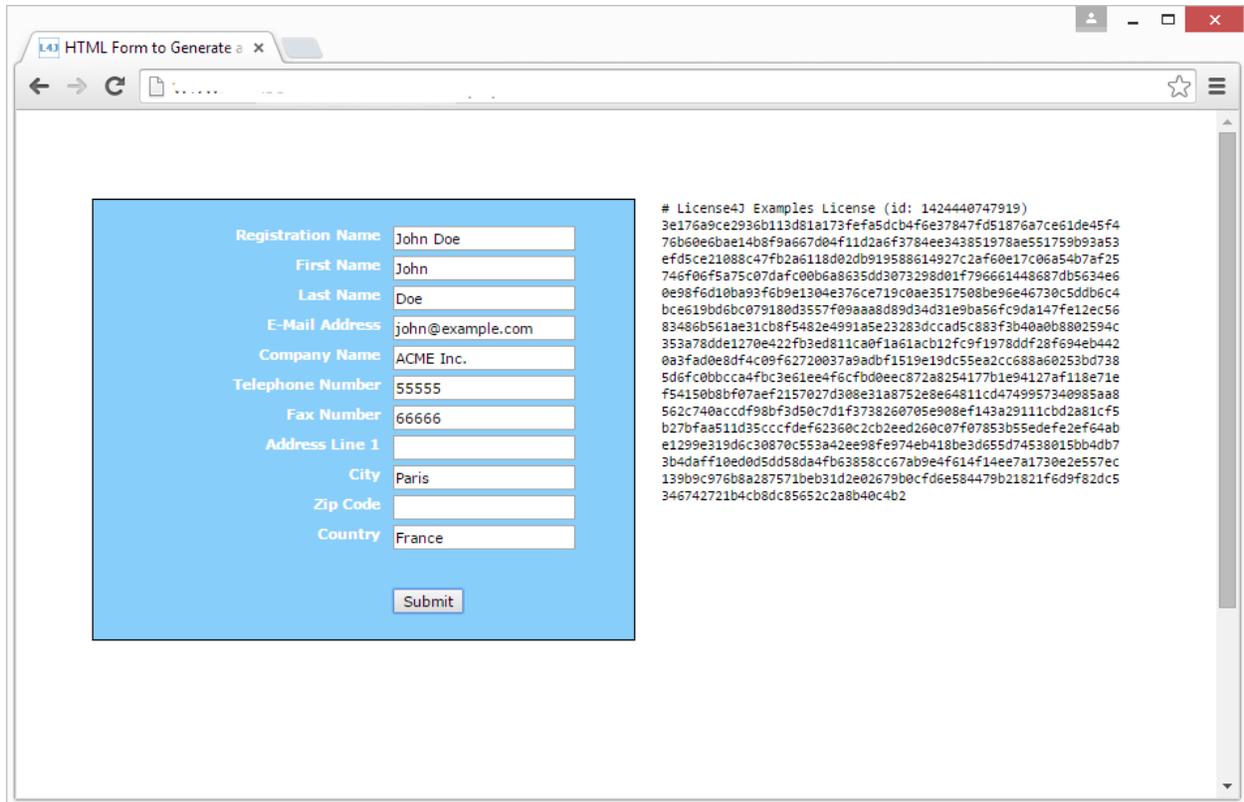
A basic sample HTML form source code is prepared from defined form variables on automatic license generation settings window; it is displayed with “Display HTML Form Code” menu item.

It can generate three sample HTML forms. Basic HTML form just submits a form to server and displays generated license. Ajax HTML form submits form with an Ajax request and displays generated license on the same page without page refresh.

Finally it also generates a PHP file for secure license generation; since PHP is a server side language, users cannot see license generation URL and security codes in PHP script. Open in Browser button saves generated HTML file to user’s home folder and open in default browser.



A sample screenshot given below to display generated Ajax form page.



Send License E-Mail

License Manager GUI can connect to any defined SMTP server and send license and activation in an attachment file to customer or any other recipient.

SMTP server settings must be defined on *Mail Server Settings* window which displayed when *Mail Server Options* menu item clicked under *Tools* menu. As seen in the screenshot below, any SMTP server with or without authentication can be used. Mail settings are should be defined as a template. From name and e-mail address are mandatory fields which will be used in sending e-mail. CC and BCC fields are optional and can be used if required. Attachment field defines attached license file name. There are two message editors available; a simple WYSIWYG editor and HTML Source editor.

SMTP Server Information

SMTP Host: Port:

Use TLS

Require Authentication

Username: Password:

Mail Settings

From Name: From E-Mail:

CC:

BCC:

Subject:

Attachment:

Message:

Dear {User-FullName},

Your license for product {Product-Name} {Product-Edition} {Product-Version} is attached.

Regards

The keywords in curly braces will be replaced by values from the license. Supported replacement keywords are: *{Product-ID}* *{Product-Name}* *{Product-Edition}* *{Product-Version}* *{License-ID}* and *{User-FullName}*.

License menu and context menu when right clicked on a license or activation includes a *Send E-mail* menu item. As seen in the screenshot below, all keywords between curly braces are replaced with the values in selected license. TO field is obtained from user e-mail setting in license. Message can be modified on this window just before sending e-mail.

Send License E-Mail

Message

From Name: From E-Mail:

TO:

CC:

BCC:

Subject:

Attachment:

Message:

Dear Sample Name,

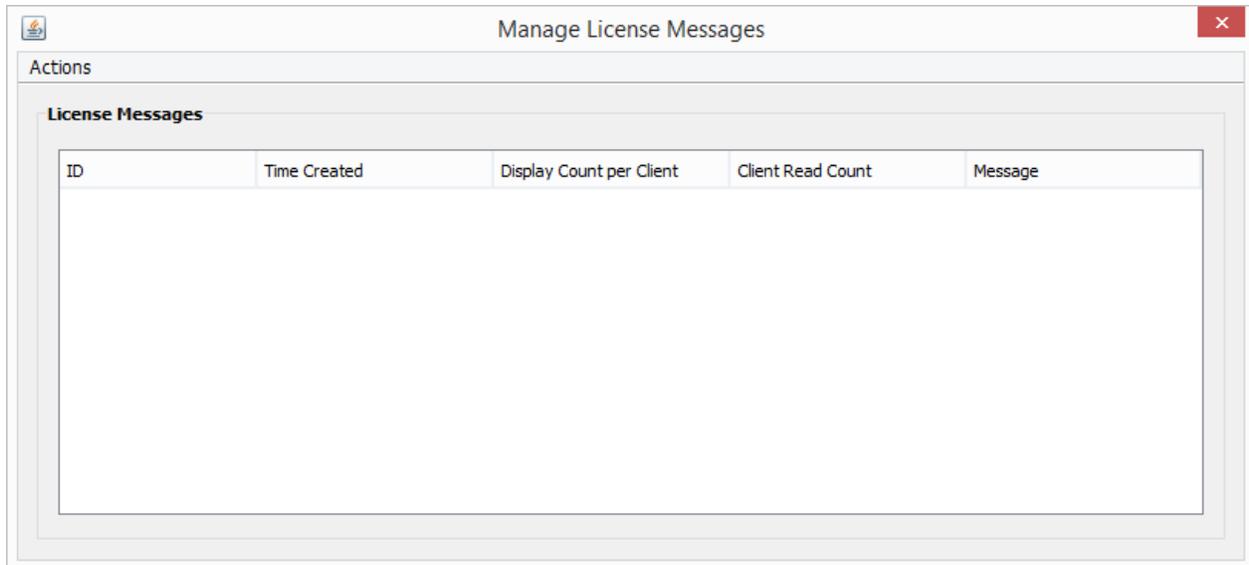
Your license for product **Example Application** is attached.

Regards

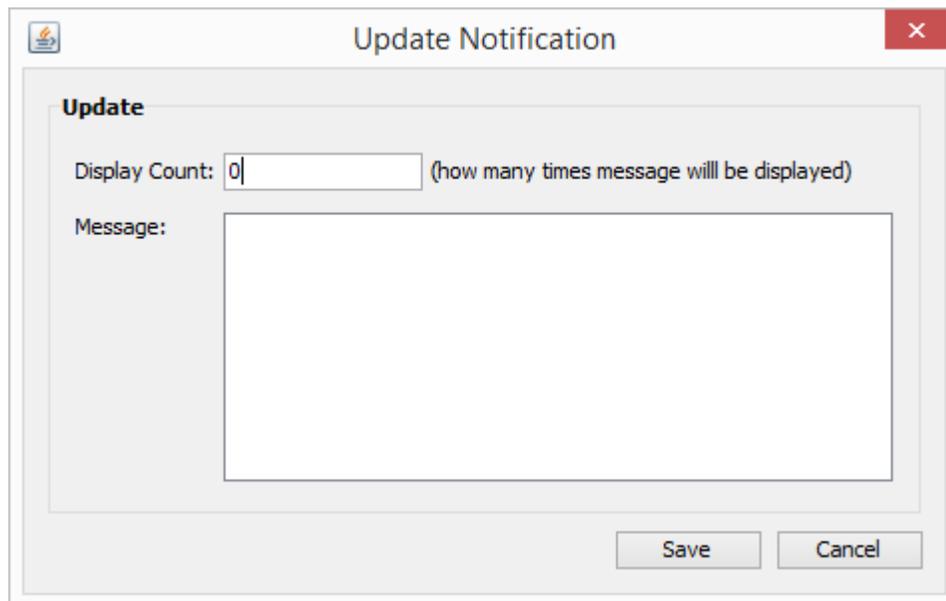
License Messages

Messages can be defined for licenses to be sent to license owner customers. Runtime library has a method to check for messages for the license. If you want to send a message to a customer after license validation, it can be added with related menu item in *License* menu.

Manage License Messages window as in the screenshot below displays all messages for the selected license. With *Actions* menu new messages can be created, edited or deleted. *Client Read Count* column displays total number of different client computers which message is displayed on.

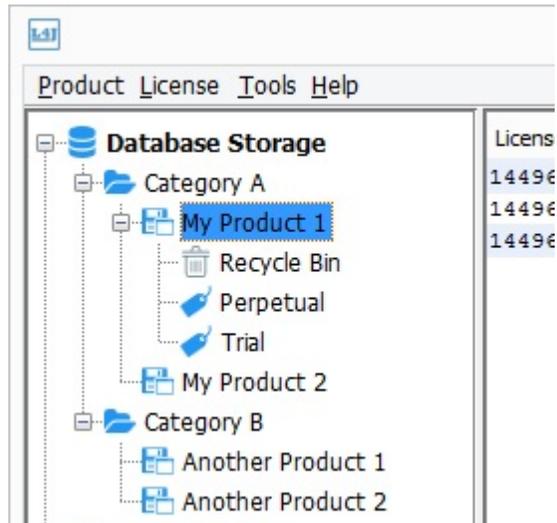


New message window has two fields. *Display Count* is the number of times which message will be displayed to user. After *checkForNewMessage* method, if new message is found it is returned to user and display count is increased for the user computer. Message is displayed until display count value is reached. If display count is zero, message is always displayed.

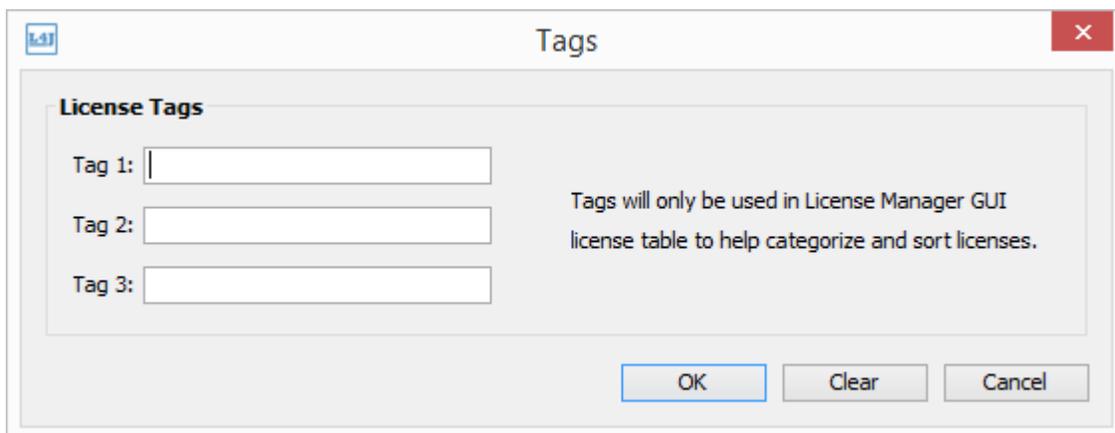


License Tags

Each license can have maximum of three tags. Tags is used to sort licenses on the table or on the left product tree. Each license tag will become a tree node when “View License Tags” menu item is enabled on Product menu or pop up menu.



On the License menu and pop up menu, there is a menu item named as “Tag”; when it is clicked a dialog is displayed to define tags.



License Use Update (last time used & total use count)

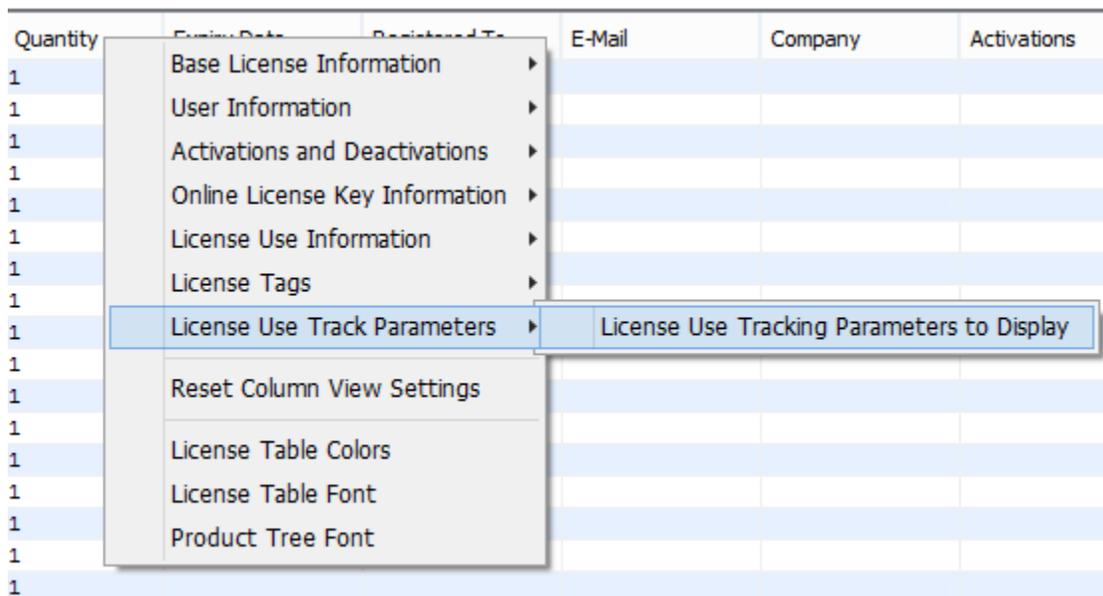
Basic license use information can be sent to server and displayed on License Manager GUI. *LicenseValidator.updateLicenseUseInfo* method in runtime library

should be called after license validation. License Manager GUI license table includes two columns named as "Total Use Count" and "Last Use Time"; when *updateLicenseUseInfo* method is called total use count is incremented and last use time is updated. This feature can be used track actively used license and software.

License Use Tracking

Runtime library has two methods to send/query key and value pairs within a hashmap to server. *LicenseValidator.updateLicenseUseTrackingInfo* method sends all key and values passed in as hashmap argument to server. *LicenseValidator.queryLicenseUseTrackingInfo* method sends key names to server to query and returns back with values.

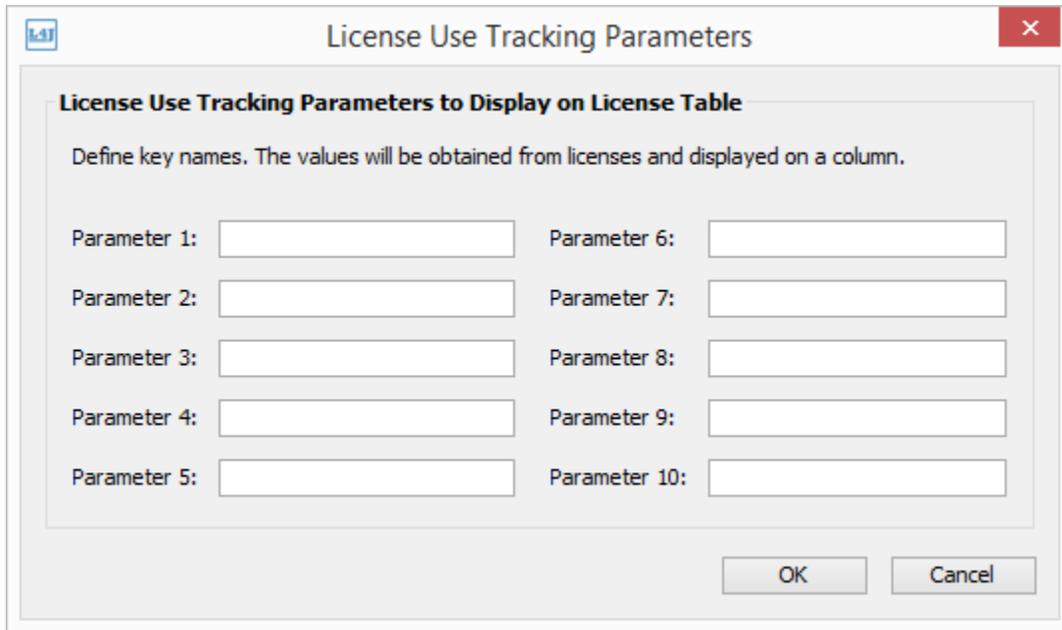
License Manager GUI license table displays up to ten license use tracking keys. The keys to be displayed is defined in the related menu item of table header pop up menu.



Quantity	Expiry Date	Registered To	E-Mail	Company	Activations
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					
1					

- Base License Information
- User Information
- Activations and Deactivations
- Online License Key Information
- License Use Information
- License Tags
- License Use Track Parameters
- Reset Column View Settings
- License Table Colors
- License Table Font
- Product Tree Font

License Use Tracking Parameters to Display



License Use Tracking Parameters

License Use Tracking Parameters to Display on License Table

Define key names. The values will be obtained from licenses and displayed on a column.

Parameter 1: Parameter 6:

Parameter 2: Parameter 7:

Parameter 3: Parameter 8:

Parameter 4: Parameter 9:

Parameter 5: Parameter 10:

OK Cancel

Both key and value should be *String* in the *HashMap* (*HashMap<String, String>*). The maximum number of characters for both key and value is 255; and the maximum allowed number of keys is 255 for each license.

Managing Modification Keys

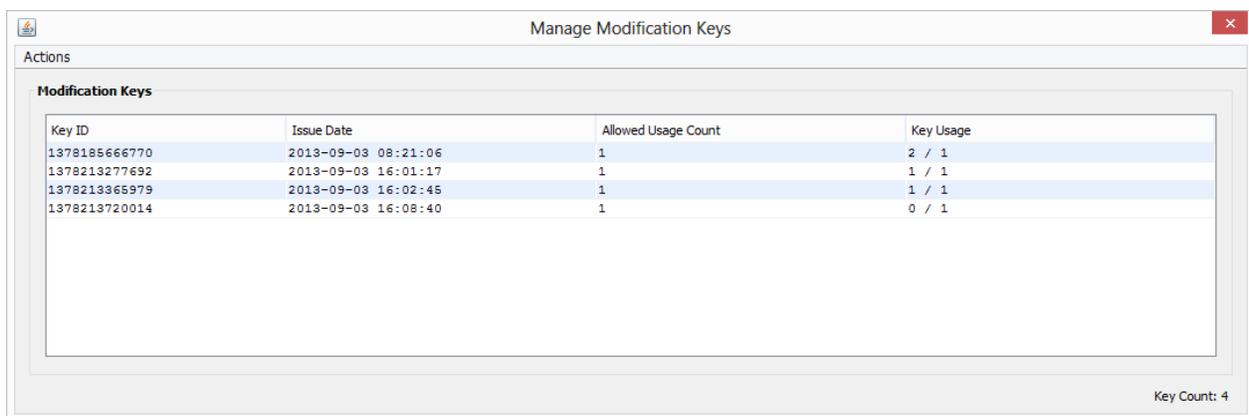
Modification keys are used to modify an already activated license both on server and client. They are similar to license keys in format, can be 25 or 55 characters long; also a special key can be defined between 5 and 255 characters long. Modification keys are used with *modifyLicense* method found in runtime library. User information, valid product edition and version, validity and maintenance validity periods, and custom features can be modified.

Modification keys are useful if you want to modify an already activated license for a customer. A custom feature may be added to add a new feature to software product or expiration date can be extended. So you do not need to generate a new license to extend validity periods, or modify features for a customer. Once a modification

key is used, license is modified and new activated license text is obtained, then it can be replaced with the old activated license text on customer computer.

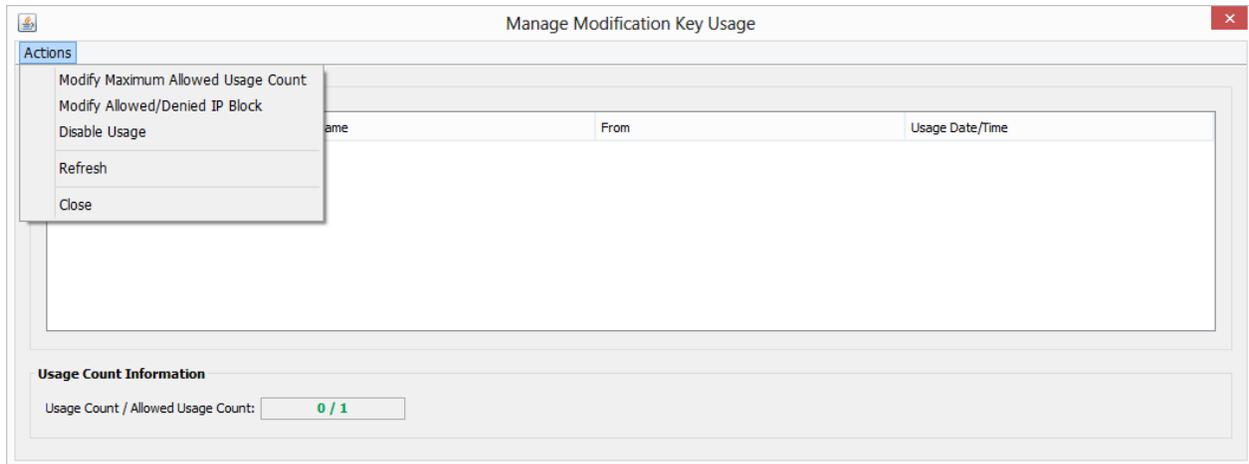
Modification keys are generated for products, so management window can be displayed by selecting a product in left tree and using License Modification Keys menu item. Modification keys also can have templates and auto generation settings, and related windows can be displayed in the same menu.

A screenshot of modification key management window is given below.



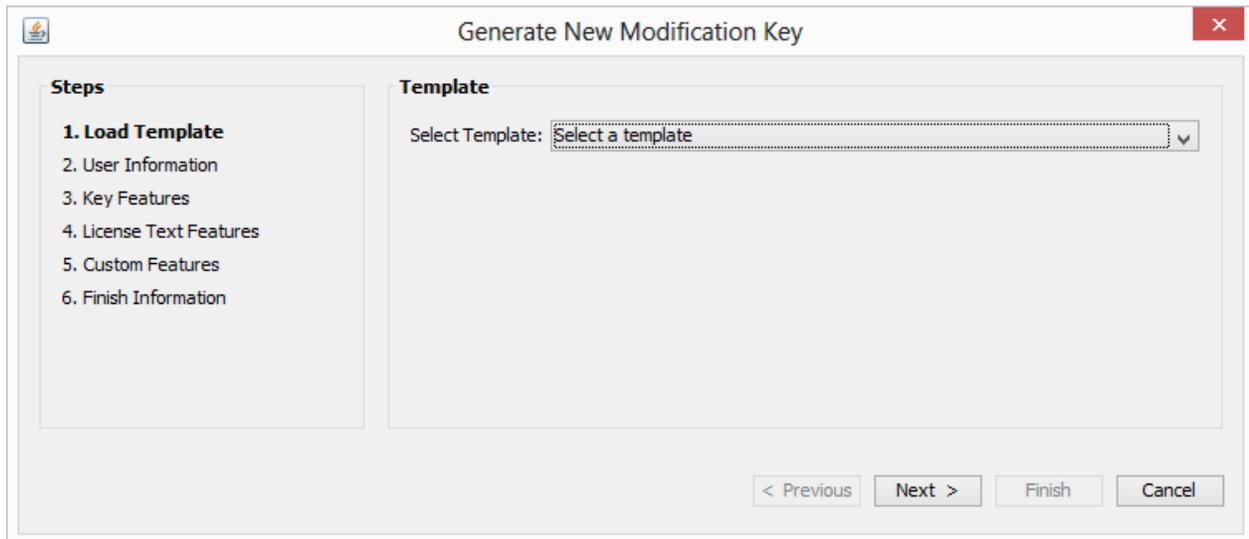
Key ID	Issue Date	Allowed Usage Count	Key Usage
1378185666770	2013-09-03 08:21:06	1	2 / 1
1378213277692	2013-09-03 16:01:17	1	1 / 1
1378213365979	2013-09-03 16:02:45	1	1 / 1
1378213720014	2013-09-03 16:08:40	1	0 / 1

Like license and activation management, all actions are available in Actions menu. Key usage window is displayed with "Manage Key Usage" menu item. In this window, key usage information is displayed, and also usage restrictions can be applied. As in the license activations, IP block restriction, maximum usage count can be modified, also key temporarily or permanently can be disabled.



Modification Key Generation

“Generate New Key” menu item is used to generate keys, and a wizard dialog is displayed to define key features.



If a modification key template is defined, it can be loaded at the first step.

If required modification keys can modify license owner information, the second step is for user information.

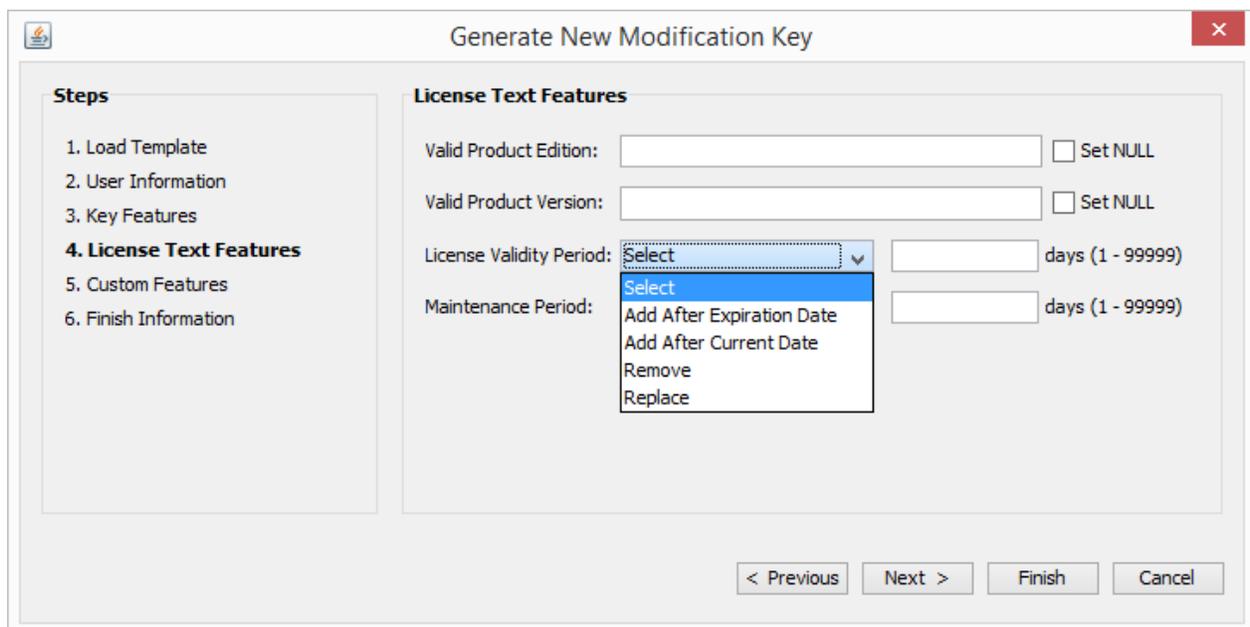
The screenshot shows the 'Generate New Modification Key' dialog box at Step 2, 'User Information'. The 'Steps' list on the left includes: 1. Load Template, 2. User Information (highlighted), 3. Key Features, 4. License Text Features, 5. Custom Features, and 6. Finish Information. The 'User Information' section contains the following fields: Full Name, Register To, Company, E-Mail, Street, Telephone, City, Fax, Zip Code, and Country. The 'Next >' button is highlighted.

The third step defines key type and quantity. Modification key can be used until defined quantity reached, and it modifies defined number of licenses. *Dedicated To License ID* field is used to dedicate this generated modification key to a specific license, and it cannot be used to modify any other license. Key validity period defines modification key expiration date. The key expires if not used with validity period. Special key field allows to define a special modification key for a customer; it can be anything from 5 to 255 characters.

The screenshot shows the 'Generate New Modification Key' dialog box at Step 3, 'Key Features'. The 'Steps' list on the left includes: 1. Load Template, 2. User Information, 3. Key Features (highlighted), 4. License Text Features, 5. Custom Features, and 6. Finish Information. The 'Key Features' section contains the following fields: Allowed Usage Count (1), Dedicated To License ID (0), Key Validity Period (0 days), and Key Format (radio buttons for Basic Key, Secure Key, and Special Key (min 5, max 255 chars)). A text box below the radio buttons contains the value 'SPECIAL-KEY-FOR-CUSTOMER-ACME'. The 'Next >' button is highlighted.

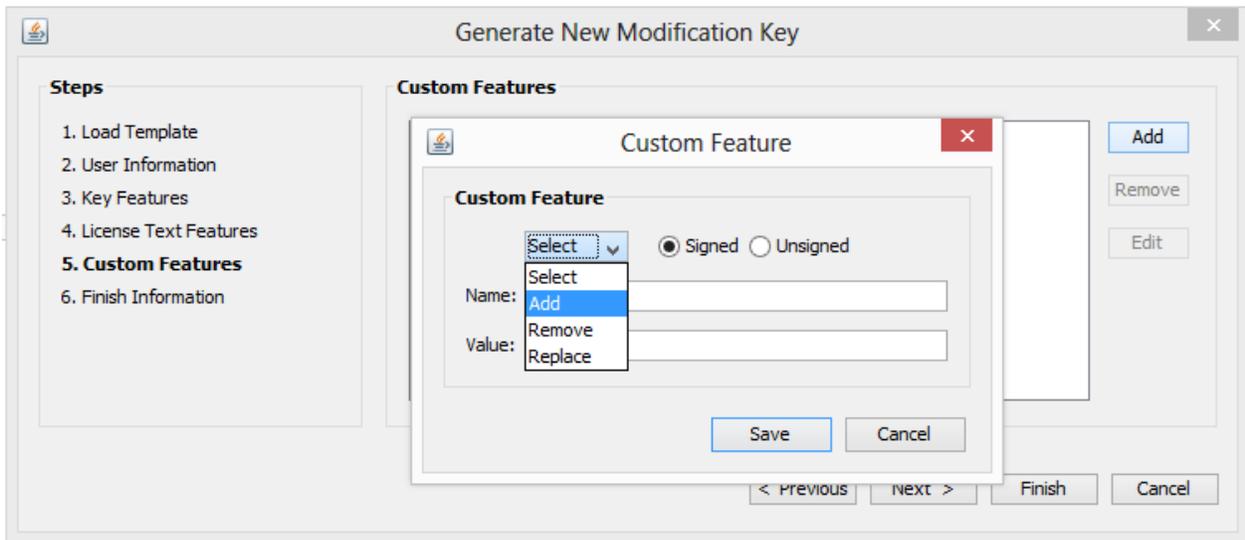
The fourth step defines license text features to be modified. If defined, valid product edition and version values replaces the same features in license. Validity period and maintenance period definition is used with an action found in the combo box. Add/Remove action adds/removes defined number of days to license validity/maintenance validity periods. Replace action changes the old value to the defined new value.

Defined number of days are added either after license expiration date or current date. E.g. If original license validity period is set as 365 days, the license expires in one year. If user buys extension and you generate modification key to extend validity period after expiration date, new value for days is directly added to original 365 days. If you generate modification key to extend validity period after current date, new value for days are added after modification key generation date (current time); so that days passed between original license expiration date and extension purchase date are also added to new validity period.



The screenshot shows the 'Generate New Modification Key' dialog box. On the left, a 'Steps' list includes: 1. Load Template, 2. User Information, 3. Key Features, 4. License Text Features (highlighted), 5. Custom Features, and 6. Finish Information. The main area is titled 'License Text Features' and contains four fields: 'Valid Product Edition' and 'Valid Product Version', each with a 'Set NULL' checkbox; 'License Validity Period' and 'Maintenance Period', each with a text input field and 'days (1 - 99999)' label. A dropdown menu is open for the 'License Validity Period' field, showing options: 'Select', 'Add After Expiration Date', 'Add After Current Date', 'Remove', and 'Replace'. At the bottom, there are buttons for '< Previous', 'Next >', 'Finish', and 'Cancel'.

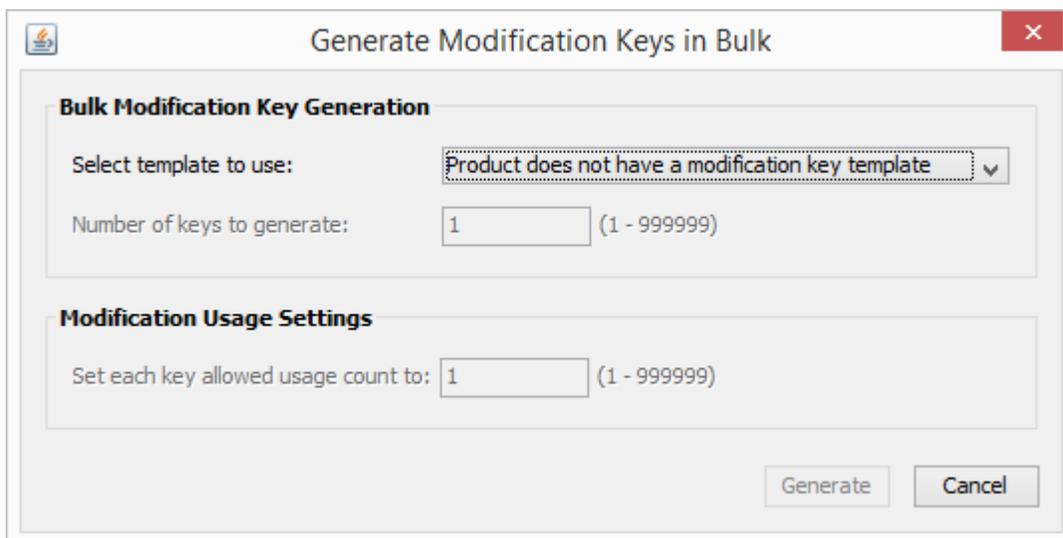
The next step defines the custom features. As in the license generation signed/unsigned custom features can be defined, but in this step also an action is defined. Add, Remove, Replace actions modifies license custom features accordingly.



Finally, defined values are displayed and modification key is generated.

Modification Key Generation in Bulk

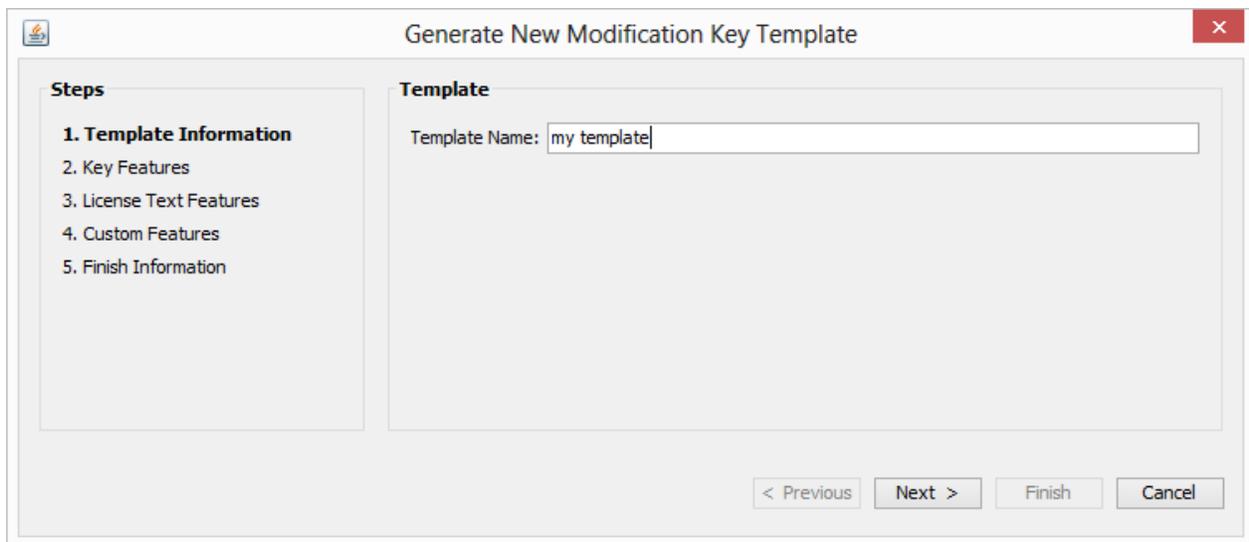
Modification keys can be generated in bulk by using a template.



Manage Modification Key Templates

Templates can be used to quickly generate modification keys; templates are also used for automatic key generation with HTTP POST requests.

Again a wizard is used in template creation, and it is similar to key generation except it does not have features applicable to key only. Template name is defined at the first step.



The screenshot shows a wizard dialog box titled "Generate New Modification Key Template". On the left, a "Steps" list includes: 1. Template Information (selected), 2. Key Features, 3. License Text Features, 4. Custom Features, and 5. Finish Information. The main area, labeled "Template", contains a text input field for "Template Name" with the value "my template". At the bottom right, there are four buttons: "< Previous", "Next >", "Finish", and "Cancel".

Modification key format is defined at the second step. Disabled fields are only for modification key usage, they cannot be used in templates.

The screenshot shows the 'Generate New Modification Key Template' dialog box. The 'Steps' panel on the left lists five steps: 1. Template Information, 2. Key Features (highlighted), 3. License Text Features, 4. Custom Features, and 5. Finish Information. The 'Key Features' section contains the following fields:

- Allowed Usage Count: (1-999999)
- Dedicated To License ID: (optional)
- Key Validity Period: days (0 - 999999)
- Key Format: Basic Key, Secure Key, Special Key (min 5, max 255 chars)

At the bottom, there are four buttons: '< Previous', 'Next >' (highlighted with a dashed border), 'Finish', and 'Cancel'.

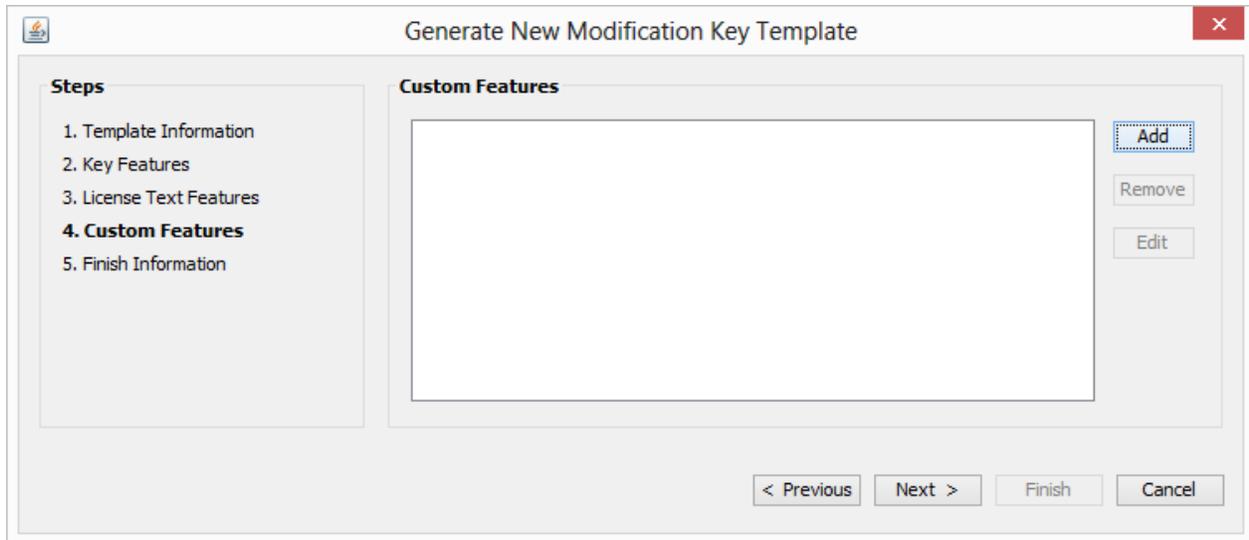
License text features are defined at the third step, and they are same as in the key generation.

The screenshot shows the 'Generate New Modification Key Template' dialog box. The 'Steps' panel on the left lists five steps: 1. Template Information, 2. Key Features, 3. License Text Features (highlighted), 4. Custom Features, and 5. Finish Information. The 'License Text Features' section contains the following fields:

- Valid Product Edition: Set NULL
- Valid Product Version: Set NULL
- License Validity Period: days (1 - 99999)
- Maintenance Period: days (1 - 99999)

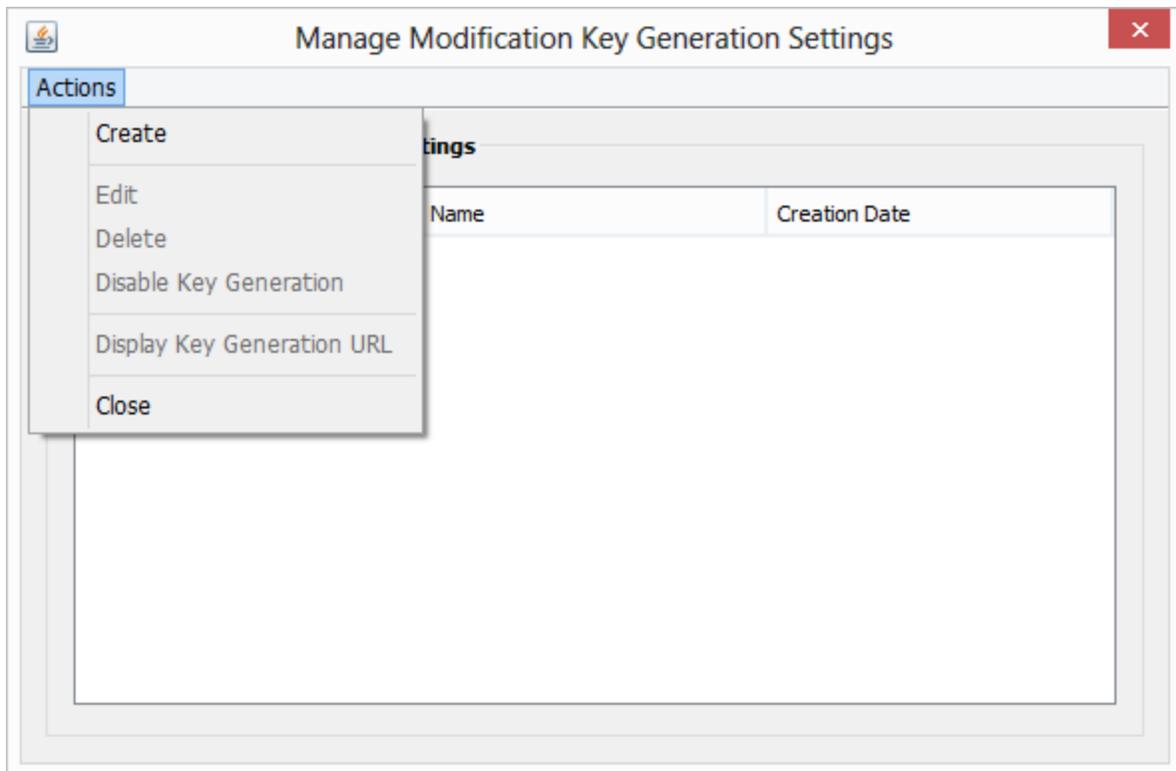
At the bottom, there are four buttons: '< Previous', 'Next >' (highlighted with a dashed border), 'Finish', and 'Cancel'.

Custom feature definition is also same as in key generation.



Manage Modification Key Auto Generation Settings

Modification keys can be generated with a received HTTP POST on either Online.License4J or Auto License Generation and Activation Server. Key generation settings window displays defined settings.



Create menu item in actions menu is used to create an auto generation setting, and it is similar to auto license generation setting. The only difference is that, output settings are removed because it always returns a single modification key.

There are predefined settings for Share-it, RegNow, SWREG, FastSpring, Avangate and PayPro which can be loaded on the bottom of window.

Modification Key Generation Settings

General **Input Settings**

Settings

Descriptive Name:

Security Code:

Access Restrictions

Allowed IPs:

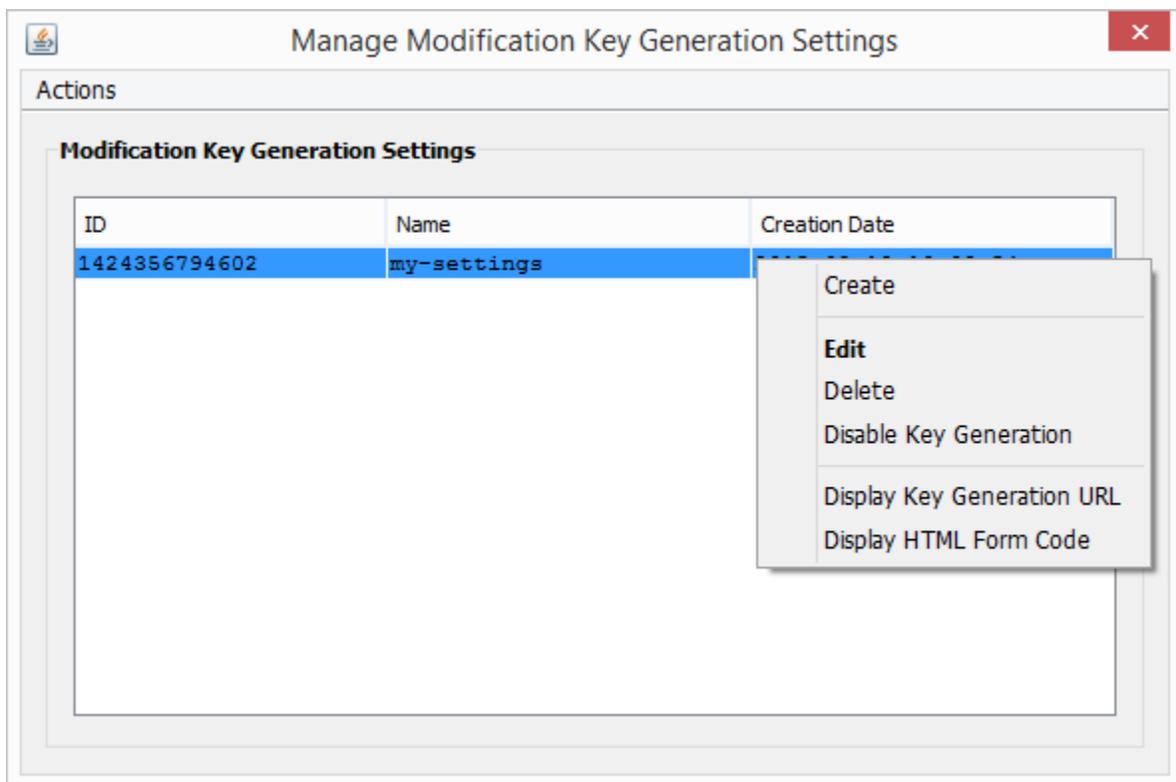
Allowed Domains:
(Will be used for Access-Control-Allow-Origin header to accept AJAX requests)
(Space separated list of domains)

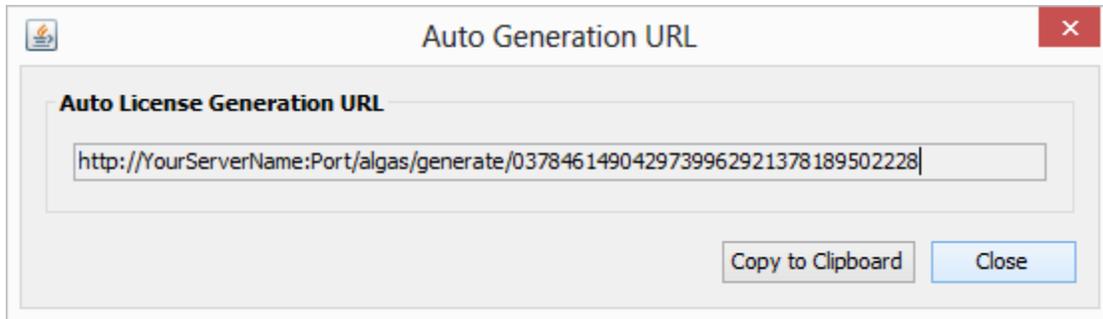
Predefined:

- Select
- Share-it
- RegNow
- SWREG
- FastSpring
- Avangate
- PayPro Global

After an auto generation setting is created, the required URL to be used to post values are displayed with "Display Key Generation URL". A template selections is made and URL is displayed in a dialog as in the screenshot below.

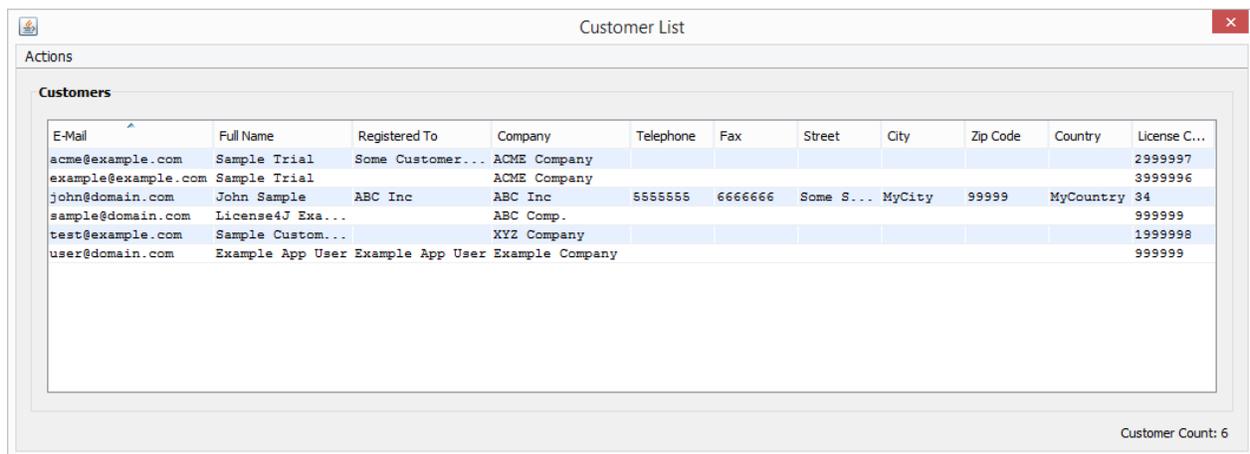
"Display HTML Form Code" menu item displays an HTML form source code for automatic modification key generation with an HTTP form post. The window is same as in license generation.





Customer List

Customer list is displayed for each product when “View Customer List” menu item is run. E-mail address is the unique identifier for customers; so a customer line is displayed on table for each different e-mail address. Following is a sample screenshot for customer list.



From the actions menu or context menu, customer and license details can be displayed; also list can be exported to a CSV or Excel file.

Tools and Options

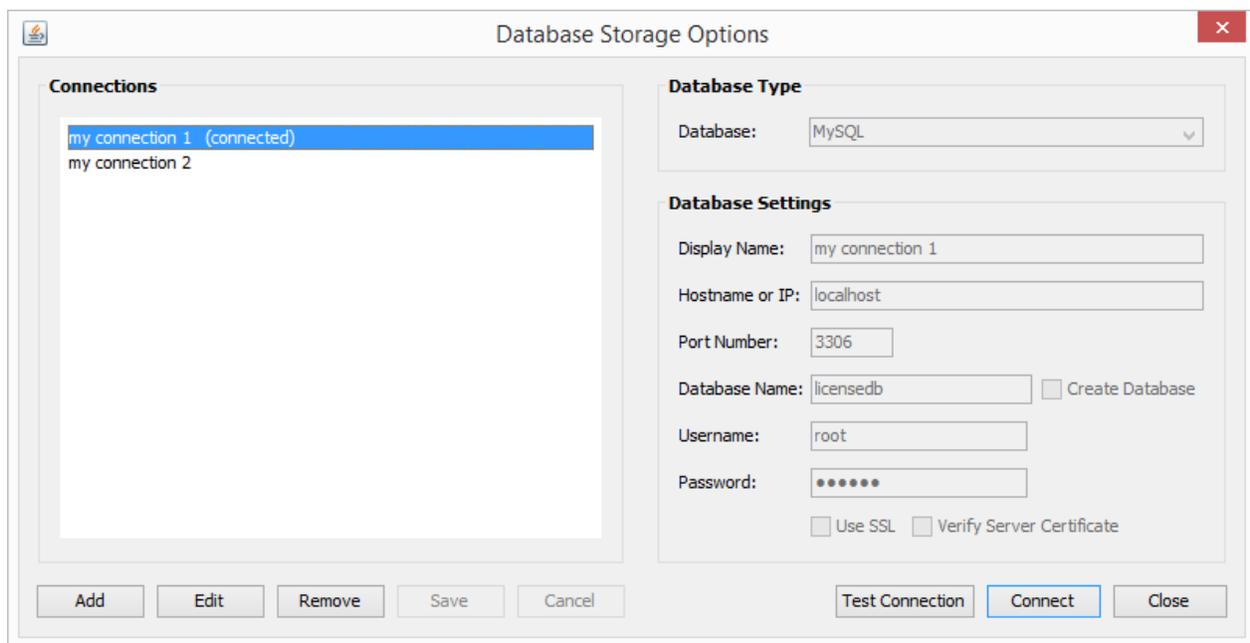
Within *Tools* menu, storage options and online connection options are managed.

Database Storage Options

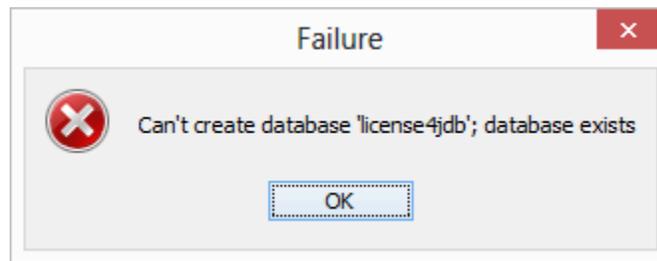
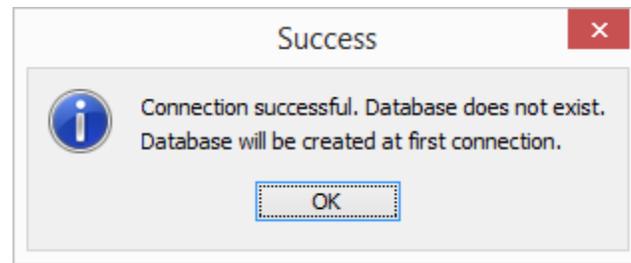
License Manager can use local embedded Java database, MySQL, PostgreSQL, and MS SQL Server for data storage. By default local embedded Java database is used and an embedded database in selected folder is created.

All database connections are displayed on list; when any connection is selected details are displayed on the right. Connection to selected database is only changed when *Connect* button is used.

If database server supports secure connection, SSL connection is possible if *Use SSL checkbox* is selected.



When *Create Database* checkbox is selected and defined database name does not exist on server then it is created. If defined database already exists, then an error message is displayed and the existing database is used.



Online.License4J Settings

Online.License4J login and logout is controlled with *Work Online* and *Work Offline* menu items.

Online Storage Connection Options

Online.License4J connection options can be changed on with *Online Storage Connection Options* menu item.

Online Storage Options

Host and Authentication

Host: Use SSL

Username:

Password:

Connect on startup

Connection via Proxy

Use proxy server for connection

Host: Port:

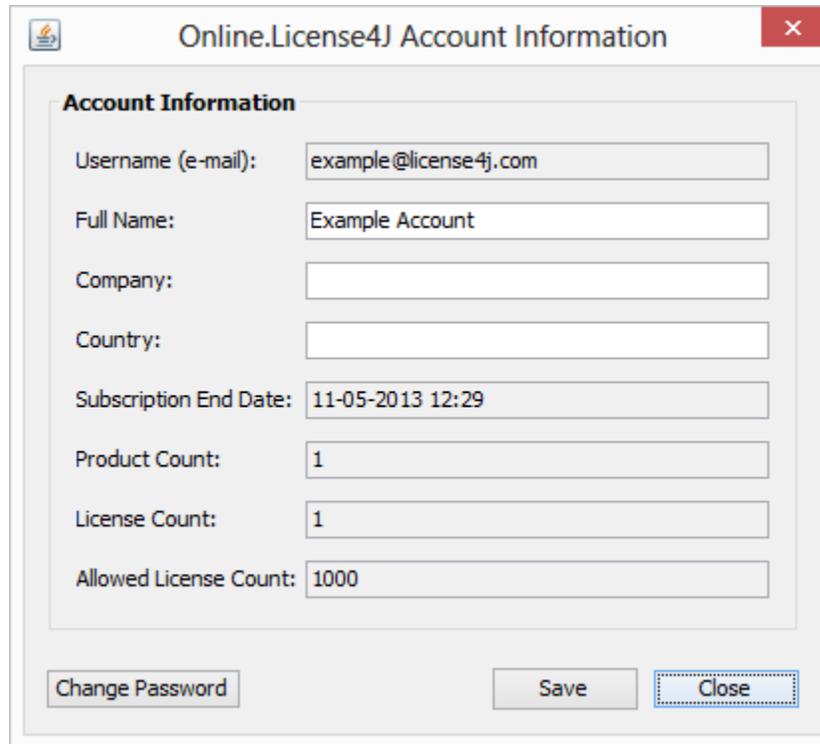
Proxy authentication required

Username:

Password:

Account Information

Account information window displays information about current connected account, and password can be changed by clicking on Change Password button.



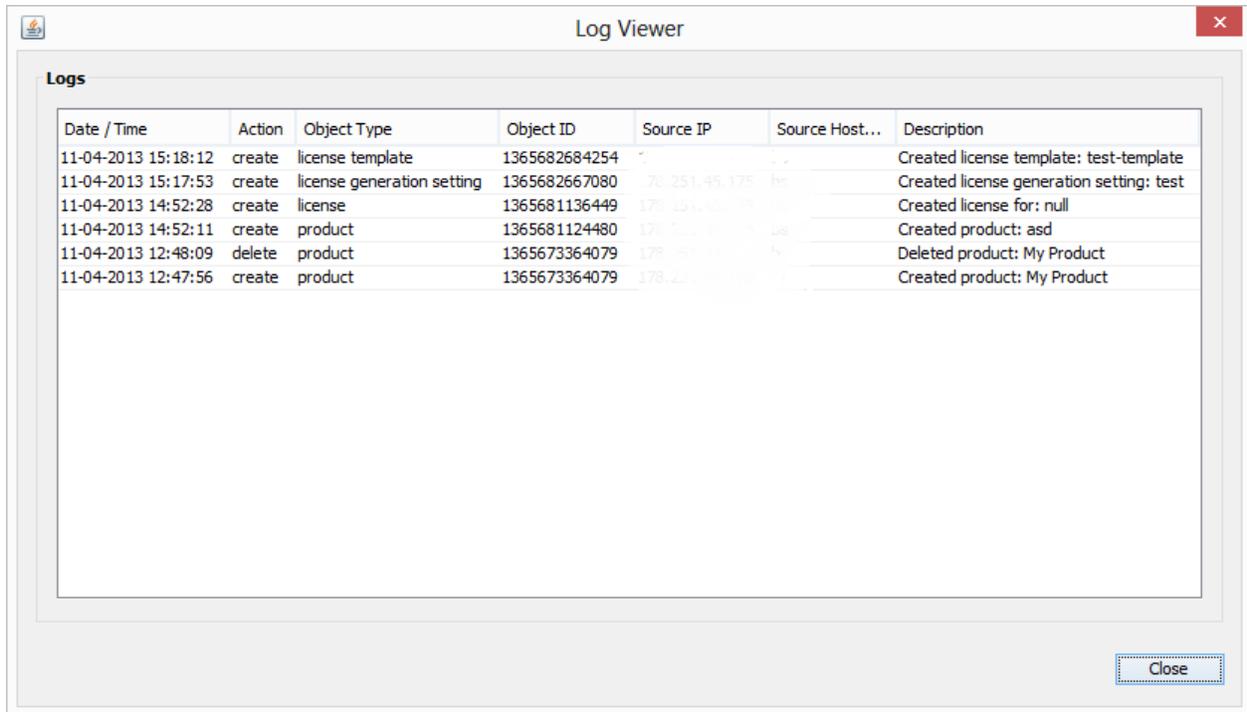
The screenshot shows a dialog box titled "Online.License4J Account Information" with a close button (X) in the top right corner. The dialog contains a section titled "Account Information" with the following fields:

Username (e-mail):	example@license4j.com
Full Name:	Example Account
Company:	
Country:	
Subscription End Date:	11-05-2013 12:29
Product Count:	1
License Count:	1
Allowed License Count:	1000

At the bottom of the dialog, there are three buttons: "Change Password", "Save", and "Close".

Log Viewer

Log viewer is available only when connected to Online.License4J, and it displays create, edit, and delete actions performed by current connected user.



Hardware ID Use

License Manager supports hardware ID generation from hostname, Ethernet MAC address, disk volume serial number and disk manufacturer serial number. Any single hardware ID or all of them can be used in hardware validation.

Version 4.4.0+ supports custom hardware IDs generated by developers. Any string can be used as a custom hardware ID, so it is possible to use practically anything as a hardware ID.

License Manager Package comes with a Hardware ID Viewer application to display all hardware IDs. You can distribute it standalone or bundled in your product free of charge to anyone.

While generating license, a single or combination of hardware IDs can be defined. AND (&&) and OR (||) combinations available. For example, to check for both hostname and disk volume serial number hardware IDs the combination should be defined as

17c9df801-238a-3e28-8ae2-675fd3166a1a&&32c43ad26-36ca-356c-8a7f-fa3118fb6e0e

both of hardware IDs should pass the check for hardware ID validation success. OR (||) combination checks succeed if any of the hardware ID validates.

License4J Auto License Generation and Activation Server

License4J Auto License Generation and Activation Server settings can be managed with Server Settings menu item in tools menu.

Server settings window screenshot is given below. Details of settings are available in License4J Auto License Generation and Activation Server User Guide.

Auto License Generation and Activation Servers

Server List:

Server Information

Name:

Version: Hostname:

Description:

Hardware ID:

Tomcat Path:

Context Main Page:

License Key:

Maintenance End:

Server Roles

<input checked="" type="checkbox"/> License Generation	<input checked="" type="checkbox"/> Online Key Validation	<input checked="" type="checkbox"/> License Availability Check
<input checked="" type="checkbox"/> Auto License Activation	<input checked="" type="checkbox"/> Auto License Deactivation	<input checked="" type="checkbox"/> Auto License Modification
<input checked="" type="checkbox"/> Manual License Activation	<input checked="" type="checkbox"/> Manual License Deactivation	<input checked="" type="checkbox"/> Manual License Modification

Notification settings allows defining various license generation and activation notifications with e-mail. Server connects to defined e-mail server and sends notification e-mails for selected cases.

The screenshot shows a window titled "Server Notification Settings" with a close button in the top right corner. It features three tabs: "SMTP Server Information" (active), "License Action Notifications", and "License Expire Notifications".

Under the "SMTP Server Information" tab, the following fields are present:

- SMTP Host: [Text Input]
- Port Number: [Text Input] (value: 25)
- From Name: [Text Input] (Display name in sent e-mails)
- From Address: [Text Input] (Single From e-mail address)
- To Address: [Text Input] (Server Admins, semicolon separated)
- Use TLS
- Require Authentication
- Username: [Text Input]
- Password: [Text Input]

At the bottom of the dialog, there are four buttons: "Test SMTP Settings", "User Notification e-mail Templates", "Save", and "Cancel".

License Action Notifications tab as in the screenshot below is used to define actions for which to send e-mails. Server admin notification e-mails are sent to defined e-mail addresses on first tab as "From Address". They are used as server administrators. User notification e-mails are sent to user e-mail address in license. If an e-mail address is not defined in license, user e-mails cannot be sent.

User notifications for some cases are also not possible when license string sent by runtime library cannot be found on license server database.

	Server Admin Notifications		User Notifications	
	Success	Fail	Success	Fail
Auto License Generation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Auto Modification Key Generation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Auto License Activation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manual License Activation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Auto License Re-Activation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manual License Re-Activation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Auto License Deactivation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manual License Deactivation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Auto License Modification	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manual License Modification	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Online Key Validation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
License Availability Check	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Only one license expire notification is sent for each license if notifications enabled. Notifications are sent either after license or maintenance expired or defined days left to expiration. More than one value can be defined for "Days to License Expire" fields; numbers should be separated with semicolon. e.g. if defined values are 1,7,14 then expire notifications are sent before 14 days, 7 days and 1 day of expiration date.

The screenshot shows a dialog box titled "Server Notification Settings" with a close button (X) in the top right corner. The dialog has three tabs: "SMTP Server Information", "License Action Notifications", and "License Expire Notifications", which is currently selected. The "License Expire Notifications" tab contains the following settings:

	Server Admin Notifications	User Notifications
Days to License Expire <input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
Days to Maintenance Expire <input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>

(Comma seperated number of days should be defined to send notification e-mails before expiration.)
(e.g. 1,7,14 to send notification e-mails 1 day, 7 days and 14 days before.)

After License Expired	<input type="checkbox"/>	<input type="checkbox"/>
After Maintenance Expired	<input type="checkbox"/>	<input type="checkbox"/>

(One notification e-mail is sent when expired.)

At the bottom of the dialog, there are three buttons: "User Notification e-mail Templates", "Save", and "Cancel".

There are default user notification e-mail templates available on database; they can be edited in e-mail template editor. Combo box lists all actions to edit. Some strings between curly braces are replaced with values from license and product.

User Notification e-mail Templates

Template Selection

Template: Successful Auto License Activation

Edit e-mail Template

CC:

BCC: {Server-Admins}

Subject: Successful Auto License Activation

Send License String Attached with File Name:

Basic Editor HTML Source

Message:

```

Product Name: {Product-Name}
License ID: {License-ID}
Activation ID: {Activation-ID}
Registered To: {User-RegisteredTo}
Company: {User-Company}
Activation count out of total allowed: {Activation-TotalCount}/{Activation-AllowedCount}

Regards

```

Reset To Default Save Close

String between curly braces below are replaced by server while sending e-mails.

{Server-Admins}	Administrator e-mails defined on SMTP information tab.
{Request-ResultMessage}	The resulting message either successful or fail.
{Request-FromIP}	IP address which request sent from.
{Request-FromHostName}	Hostname request sent from.
{Request-FromOS}	OS name request sent from.
{Product-ID}	Unique product ID.
{Product-Name}	Defined product name.
{Product-Version}	Product version if defined.
{Product-Edition}	Product edition if defined.
{License-ID}	Unique license id.
{License-LicenseString}	License string either key or long license text.
{Activation-TotalCount}	Activation number, how many activation are made.
{Activation-AllowedCount}	Allowed activation count for license.
{Deactivation-AllowedCount}	Allowed deactivation count for license.
{Deactivation-TotalCount}	Deactivation count.
{License-ValidityExpiration}	Expiration date in format yyyy-MM-dd HH:mm:ss
{License-MaintenanceExpiration}	Maintenance expiration date in format yyyy-MM-dd HH:mm:ss
{License-Quantity}	License quantity.

{License-HardwareID}	Hardware ID if license is node-locked.
{Activation-ID}	Unique activation ID.
{Activation-LicenseString}	Activation string (activated license string)
{Activation-HostName}	Hostname on which activation is completed.
{Activation-FromIP}	IP address which activation completed.
{Activation-HardwareID}	Hardware ID used in activation.
{Activation-ValidityExpiration}	Expiration date for activation (if "set activated license text generation time to activation time" is selected for license).
{Activation-MaintenanceExpiration}	Maintenance expiration date for activation (if "set activated license text generation time to activation time" is selected for license).
{ModificationKey-ID}	Unique modification key ID.
{ModificationKey-LicenseString}	Modification key itself.
{ModificationKey-DedicatedLicenseID}	License ID if modification key is dedicated to.
{ModificationKey-AllowedUsageCount}	Allowed use count for modification key.
{User-FullName}	User name defined in license.
{User-EMail}	User e-mail address in license.
{User-RegisteredTo}	Registered To field in license.
{User-Company}	Company name in license.
{User-Telephone}	Telephone number in license.
{User-City}	City name in license.

Tools menu has another Log Viewer for License4J Auto License Generation and Activation Server and it displays all successful or failed license generation/activations requests.

License4J Runtime Library Integration

License4J runtime library file name is *License4J-Library-Runtime.jar* and it is included in distribution. Library should be included in product for license validation. License validation and activation methods returns `ValidationStatus` and `ActivationStatus` enumeration objects.

ValidationStatus

Defined constants and required conditions are listed below.

- LICENSE_VALID

Applicable to all license types, and only obtained when license is valid.

- LICENSE_INVALID

Applicable to all license types, and obtained when license is invalid or license generation time is set to a future time.

- LICENSE_EXPIRED

Applicable to license text, floating license text and online basic key floating over internet. It is obtained when license validity period is over.

- LICENSE_MAINTENANCE_EXPIRED

Applicable to license text, floating license text, and online basic key floating over internet. It is obtained when license maintenance period is over and current product release date is specified in validate method.

- MISMATCH_PRODUCT_ID

Applicable to license text, floating license text, and online basic key floating over internet. It is obtained when given product id in validate method does not match the product id defined in license.

- MISMATCH_PRODUCT_EDITION

Applicable to license text, floating license text, and online basic key floating over internet. It is obtained when given product edition in validate method does not match the product edition defined in license. If a product edition is not defined in license, it will never be obtained.

- MISMATCH_PRODUCT_VERSION

Applicable to license text, floating license text, and online basic key floating over internet. It is obtained when given product version in validate method does not match the product version defined in license. If a product version is not defined in license, it will never be obtained.

- **MISMATCH_HARDWARE_ID**
Applicable to license text only, and it is obtained when computer hardware id does not match the hardware id defined in license.
- **FLOATING_LICENSE_SERVER_NOT_AVAILABLE**
Applicable to floating license text and online basic key floating over internet. It is obtained when validate method cannot connect to defined Floating License Server or Auto License Generation and Activation Server.
- **FLOATING_LICENSE_NOT_FOUND**
Applicable to floating license text and online basic key floating over internet. It is obtained when requested license is not found in Floating License Server or Auto License Generation and Activation Server. License is queried with license id, product edition and product version parameters given to validate method.
- **FLOATING_LICENSE_NOT_AVAILABLE_ALL_IN_USE**
Applicable to floating license text and online basic key floating over internet. It is obtained when maximum allowed usage count is reached for the license.
- **FLOATING_LICENSE_ALLOWED_USE_COUNT_REACHED**
Applicable to only online basic key floating over internet. It is obtained when maximum allowed use count is reached for the license.
- **FLOATING_LICENSE_ALLOWED_USE_TIME_REACHED**
Applicable to only online basic key floating over internet. It is obtained when maximum allowed usage time duration is reached for the license.
- **FLOATING_LICENSE_CLIENT_REJECTED:**
Applicable to only floating license text; it is obtained when client rejected in Floating License Server Administration GUI. See floating license server user guide for help on client reject feature.

- **FLOATING_LICENSE_OVERUSED**
The clients which obtained overused licenses will get validation status of *FLOATING_LICENSE_OVERUSED* instead of *LICENSE_VALID*, therefore it is possible to notify user that license usage exceeds license quantity. OverUse should be enabled while generating license.
- **INCORRECT_SYSTEM_TIME**
Applicable to license text, floating license text, and online basic key floating over internet. It is obtained when online date/time check feature is enabled for the license and computer time is at least 24 hours different from obtained online time.
- **VALIDATION_REJECTED_IP_BLOCK_RESTRICTION**,
Applicable to online basic key floating over internet. It is obtained when IP block restrictions are defined and license requester's IP is not in allowed IP block.
- **VALIDATION_REJECTED_FEATURE_DISABLED**
Applicable to online basic key floating over internet. It is obtained when validation is disabled in License Manager.

ActivationStatus

Defined constants and required conditions are listed below.

- **ACTIVATION_SERVER_CONNECTION_ERROR**
Obtained when client cannot connect to either Online.License4J or Auto License Generation and Activation Server.
- **LICENSE_NOT_FOUND_ON_ACTIVATION_SERVER**
Obtained when license to be activated cannot be found on either Online.License4J or Auto License Generation and Activation Server.

- **ALREADY_ACTIVATED_ON_ANOTHER_COMPUTER**
Obtained when license has single activation permission and already activated on another computer.
- **MULTIPLE_ACTIVATION_LIMIT_REACHED**
Obtained when license has multiple activation permission and all are activated.
- **MULTIPLE_DEACTIVATION_LIMIT_REACHED**
Obtained when license has defined maximum allowed deactivation count is reached.
- **ACTIVATION_COMPLETED**
Obtained when license is successfully activated.
- **ACTIVATION_REQUIRED**
Obtained before activation performed, used to test whether license requires activation or not.
- **ACTIVATION_NOT_REQUIRED**
Obtained before activation performed, used to test whether license requires activation or not.
- **ACTIVATION_HARDWAREID_ERROR**
Obtained when required hardware ID(s) cannot be generated. Required hardware ID(s) with OR or AND operands are defined during license generation; if AND operand is selected and at least one of the selected hardware IDs cannot be generated this hardware ID error returns. Obtained also if single hardware ID is required for activation and it cannot be generated.

- **ACTIVATION_REJECTED_IP_BLOCK_RESTRICTION**
Obtained when IP block restriction exists and client IP is now an allowed IP.
- **ACTIVATION_REJECTED_FEATURE_DISABLED**
Obtained when activation is temporarily or permanently disabled in License Manager GUI.
- **ACTIVATION_NOT_FOUND_ON_SERVER**
Obtained when deactivation request is sent to server but activation is not found.
- **DEACTIVATION_COMPLETED**
Obtained when the license is successfully deactivated.

ModificationStatus

Defined constants and required conditions are listed below.

- **MODIFICATION_COMPLETED**
Obtained when license is successfully modified.
- **MODIFICATION_COMPLETED_PREVIOUSLY**
Obtained when license is already modified, but calling client does not have latest license. Latest activated license returns to be saved.
- **MODIFICATION_KEY_INVALID**
Obtained when given modification key is invalid. This status is deprecated because beginning with version 4.4.0, key validation is not performed within runtime library; it is directly sent to server.
- **SERVER_CONNECTION_ERROR**
Obtained when runtime library cannot connect to defined server for license modification.

- **MODIFICATION_KEY_USED_FOR_ANOTHER_LICENSE**
Obtained when modification key quantity is 1 and it is already used to modify another license.
- **MODIFICATION_KEY_NOT_FOUND_ON_SERVER**
Obtained when given modification key is valid but it is not found on server. It was deleted by server/license administrator.
- **MODIFICATION_KEY_USAGE_LIMIT_REACHED**
Obtained when modification key quantity is more than 1, and all is used to modify other licenses.
- **MODIFICATION_REJECTED_IP_BLOCK_RESTRICTION**
Obtained when IP block restriction exists and client IP is now an allowed IP.
- **MODIFICATION_REJECTED_FEATURE_DISABLED**
Obtained when modification is temporarily or permanently disabled in License Manager GUI.
- **ACTIVATION_NOT_FOUND_ON_SERVER**
Obtained when received activated license to be modified is not found on server. It was deleted by server/license administrator.
- **MODIFICATION_STATUS_UNKNOWN**
Modification status unknown; because either license is invalid or not activated.
- **MODIFICATION_NOT_SUPPORTED**
Modification is not supported, if activation return type is an activation code.
- **MODIFICATION_REJECTED_FOR_LICENSE**

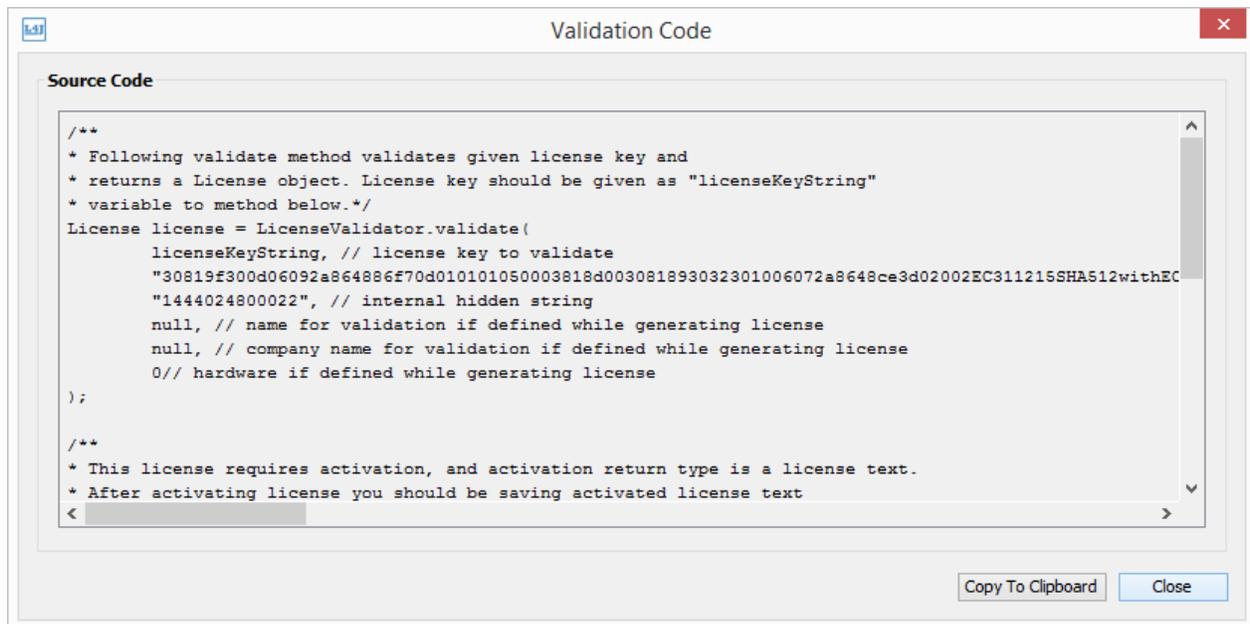
Obtained when modification is not allowed for the given license.

- MODIFICATION_KEY_DEDICATED_FOR_ANOTHER_LICENSE

If given modification key is dedicated to a single license, and given license mismatch this modification status is returned.

License Validation Code

Version 4.5.4+ includes a new feature to easily get the license validation code with all required parameters. License template management window is used to get license validation source code easily. There is a new *"Display validate Code"* menu item in license management window; it generates source code based on selected license template parameters. Sample screen shot is given below.



Easy License Validation Code

Version 4.5.4+ includes a new feature to easily validate "License Text" and "Basic License Key" and "Cryptographically Secure License Key" type licenses. There are two new methods in runtime library: *LicenseValidator.easyValidate* and *LicenseValidator.easyValidateOnStartup*

LicenseValidator.easyValidate method takes 2 arguments; the first one is the license string to validate and the second is an encrypted string which includes all parameters including license activation server if activation is enabled. This encrypted string is generated by License Manager GUI only. To get *easyValidate* method and parameters, a license template should be used. License template management window includes a menu item to get the source code.

LicenseValidator.easyValidateOnStartup method takes only 1 argument; it is an encrypted string which includes all required parameters for license validation and activation if enabled.

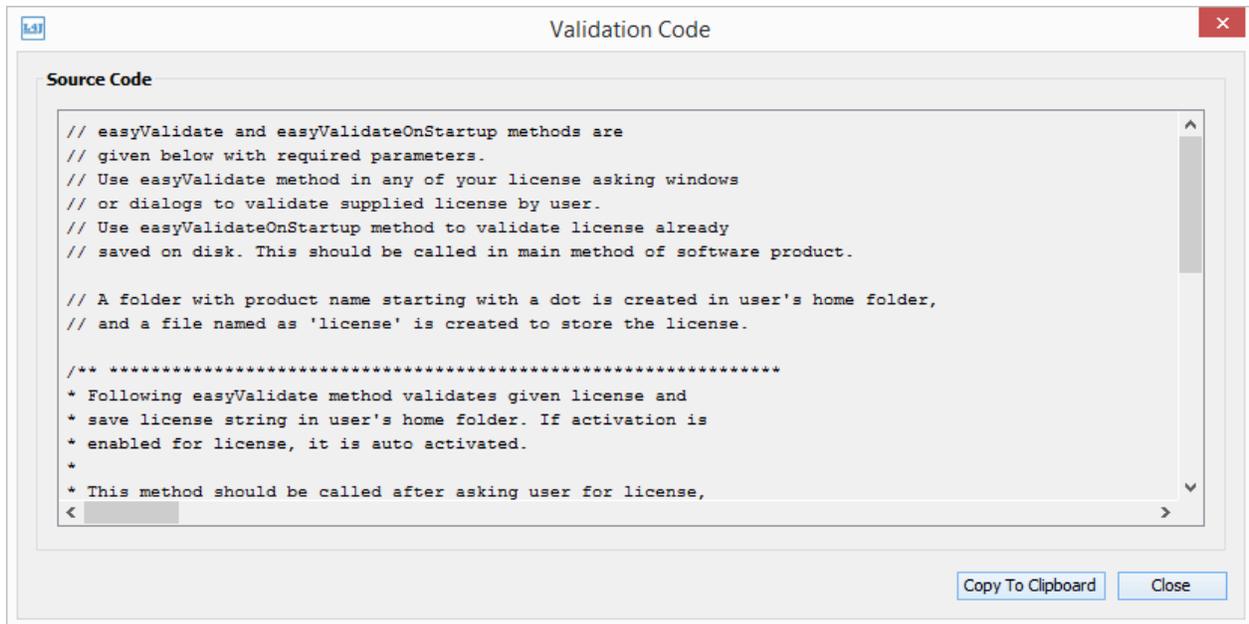
easyValidate should be called after asking user for the license. It validates given license and if enabled auto activates license. Finally it saves license in an encrypted file in user's home folder. The license file name is "license". A directory is created by using product name and license file is saved. *easyValidateOnStartup* method should be used in software product main method. It reads license file on disk and validates; it should be called on each software startup.

License file path should be carefully selected; it can be defined as a Java system property if needed. *easyValidate* method reads a Java system property when called to check for any custom license file path. The property name is **flfp** (full license file path). It can be set with the following code.

```
System.setProperty("flfp", System.getProperty("user.home") + "/" + "myLicenseFile.lic");
```

If a custom license file is defined with system property, it should be checked for access permissions. If file cannot be accessed or modified, it will print an error message. (e.g. on Windows systems, usually programs runs with standard user privileges. If you try to save license file in Program Files folder (installation path), it may not create/modify file.)

Following is a sample screen shot for *easyValidate* code window.



The screenshot shows a window titled "Validation Code" with a red close button in the top right corner. The window contains a text area with the following source code:

```
// easyValidate and easyValidateOnStartup methods are
// given below with required parameters.
// Use easyValidate method in any of your license asking windows
// or dialogs to validate supplied license by user.
// Use easyValidateOnStartup method to validate license already
// saved on disk. This should be called in main method of software product.

// A folder with product name starting with a dot is created in user's home folder,
// and a file named as 'license' is created to store the license.

/** *****
 * Following easyValidate method validates given license and
 * save license string in user's home folder. If activation is
 * enabled for license, it is auto activated.
 *
 * This method should be called after asking user for license,
 *
 * *****
 */
```

At the bottom right of the window, there are two buttons: "Copy To Clipboard" and "Close".

Basic and Cryptographically Secure License Key Validation

Basic license key validation method:

```
LicenseValidator.validate(  
    licenseString,  
    publickey,  
    internalString,  
    nameforValidation,  
    companyforValidation,  
    hardwareidValidation);
```

License string is the license key like JZENR-R6TNH-EMAXC-QYS3F-J63UL. Public key is your product's public key, it can be viewed on license manager GUI. Internal string is defined during license key generation, if not specified it is set as product's unique id by default only for basic license key. Name for validation and company for validation are the customer name and company name, which are defined during license generation, but they should be included in key validation only if specified during license generation.

If a custom hardware ID is used while generating license, `LicenseValidator.validateWithCustomHardwareID` method should be used to specify custom hardware ID.

As all validation methods, it returns a License object and it has the *ValidationStatus* enumeration. Validation status can be checked with a switch statement.

```
switch (license.getValidationStatus()) {  
    case LICENSE_VALID:  
        // TODO  
  
        break;  
  
    case LICENSE_INVALID:  
        // TODO  
  
        break;  
  
    case MISMATCH_HARDWARE_ID:  
        // TODO  
  
        break;  
  
    case INCORRECT_SYSTEM_TIME:  
        // TODO
```

```
        break;  
    }
```

License Text Validation

License text validation method:

```
LicenseValidator.validate(  
    licenseString,  
    publickey,  
    productID,  
    productEdition,  
    currentProductVersion,  
    currentDate,  
    currentProductReleaseDate);
```

Same License object returns after validation like in license key validation, and it has ValidationStatus enumeration with validation results.

If a custom hardware ID is defined in license generation or license activation features, LicenseValidator.validateWithCustomHardwareID method should be used to specify custom hardware ID.

Floating License Text Validation

Floating license text is obtained from floating license server; and validate method requests the required license.

```
LicenseValidator.validate(  
    publickey,  
    productID,  
    productEdition,  
    currentProductVersion,  
    currentDate,  
    currentProductReleaseDate,  
    serverHostname,  
    portNumber,  
    licenseValidHandler,  
    licenseInvalidHandler,  
    serverConnectionErrorHandler);
```

Product ID, edition and version are used to query for the license on floating license server. Product ID must be defined, edition and version are optional.

After a valid license is obtained from floating license server, validate method starts a thread to check license validity and inform server that leased license is still in use. Valid license handler is run when obtained license is valid. Invalid handler is run when obtained license is invalid due to e.g. expired. Server connection error handler is run when client cannot connect to floating license server. These error handlers are optional but they may be useful to force license usage policy. Basic default handlers are available in Runtime Library and also interface classes is available and can be used to implement custom handlers to manage license.

The method below is available in version 4.4.1 and later. It is simplified and a new FloatingLicenseCheckTimerHandler is added. The advantage is over the other validate

method is that, new handler has License object; to get license object getLicense() can be used. So it is possible to get updated license object and display to user.

```
LicenseValidator.validateFloatingLicenseText(  
    publickey,  
    productID,  
    productEdition,  
    productVersion,  
    serverAddress,  
    floatingLicenseCheckTimerHandler);
```

Another advantage is that more than one floating license server address can be defined as floating license server; server addresses must be separated with semicolon. e.g. *MyFirstServer:16090;MySecondServer:16090*

A handler class extending FloatingLicenseCheckTimerHandler must be created; an example is given below. run() method of handler class is called at each timer run.

NOTE: "License Server Connection Check Period" value is not used in validateFloatingLicenseText method, only license check period is used to run a timer task.

Version 4.4.3+ includes a new method with an extra argument to send some information about client to the server.

```
LicenseValidator.validateFloatingLicenseText(  
    publickey,  
    productID,  
    productEdition,  
    productVersion,
```

```
serverAddress,  
floatingLicenseCheckTimerHandler,  
additionalInfo);
```

It is additional information argument and is displayed on Floating License Manager GUI in the “additional info” column. Any string with maximum 64 characters can be sent. It can be used to display details about software on client computer like version/build number etc.

```
public class MyHandler extends FloatingLicenseCheckTimerHandler {  
    public MyHandler() {  
    }  
    @Override  
    public void run() {  
        // Call any method or update GUI here depending on license validation status  
        switch (getLicense().getValidationStatus()) {  
            case LICENSE_VALID:  
                // do something  
                break;  
            case FLOATING_LICENSE_NOT_AVAILABLE_ALL_IN_USE:  
                // do something  
                break;  
            case FLOATING_LICENSE_SERVER_NOT_AVAILABLE:  
                // do something  
                break;  
            // etc ...  
        }  
    }  
}
```

```
    }  
  }  
}
```

There is a default handler named `DefaultFloatingLicenseCheckTimerHandler`, it only prints license validation status on console.

If concurrent custom feature usage is enabled by defining quantity value for any custom feature, it must be checked out with method `LicenseValidator.checkoutFloatingLicenseTextCustomFeature(String featureName, License licenseObject, String floatingLicenseServer)` then it is verified by `license.getLicenseText().getCustomSignedFeature(featureName)` method. It is released with `LicenseValidator.releaseFloatingLicenseTextCustomFeature((String featureName, License licenseObject, String floatingLicenseServer))`.

When `validateFloatingLicenseText` method is called more than once, it always gets the same license from server because it runs over same JRE session. Therefore it may not be suitable for some situations when multiple method calls are required to be counted. To count each validate method call and assign a new floating license, there is a new Java system property defined (version 4.6.1+). When ***CountOnEachValidateCall*** property is set either in code or at startup with `-D` parameter to value "true", runtime library will get a new a new assigned license from server.

Online License Key Floating Over Internet Validation

Online license key validation parameters are given below. Returned license object includes a license text. If `serverAddress` parameter is given runtime library connects to given Auto License Generation and Activation Server to validate license. To use a

license hosted on Online.License4J system, same method without *serverAddress* parameter should be used.

```
LicenseValidator.validate(  
    key,  
    publickey,  
    productID,  
    productEdition,  
    currentProductVersion,  
    currentDate,  
    currentProductReleaseDate,  
    serverAddress,  
    onlineLicenseKeyCheckTimerHandler);
```

Validation status constants are same with any license text and floating license text. Online license key has a new feature beginning with version 4.3. Key usage limit can be defined, and users can use the key for only defined number of times so the software product. Obtained license object has two additional values for key use count if defined during license key generation. `getUseCountAllowed()` method returns maximum allowed use count value; `getUseCountCurrent()` method returns current use count. `getUseTimeCurrent()` and `getUseTimeLimitAllowed()` methods return current usage time duration and maximum allowed time duration if enabled when generating license.

Note: Both Floating License Text and Online License Key Floating Over Internet must be validated once in a single JVM run because each license validation starts timer tasks at the background to ping and check floating license servers. If more than one call to validate occurs there will be unnecessary timer tasks running at the background and usage information will be incorrect on server. If It is needed to

revalidate a floating license without shutting down JVM, first call *stopFloatingLicenseCheckTimers()* method found in *License* object to stop all background timer tasks.

Note: Beginning with version 4.4.0, there is a new method *releaseFloatingLicense()* in *License* object. It stops floating license check timers then sends a release request to server.

The method below is available in version 4.4.1 and later. It is simplified and a new *FloatingLicenseCheckTimerHandler* is added. The advantage is over the other validate method is that, new handler has *License* object; to get license object *getLicense()* can be used. So it is possible to get updated license object and display to user.

```
LicenseValidator.validateOnlineLicenseKey (  
    key,  
    publickey,  
    productID,  
    productEdition,  
    productVersion,  
    serverHostname,  
    floatingLicenseCheckTimerHandler);
```

Another advantage is that more than one license server address can be defined be separated with semicolon. e.g. *http://MyFirstServer:8080;http://MySecondServer*

Version 4.4.3+ includes a new method with an extra string argument as "additionalInfo".

```
LicenseValidator.validateOnlineLicenseKey (  
    key,  
    publickey,  
    productID,  
    productEdition,  
    productVersion,  
    serverHostname,  
    floatingLicenseCheckTimerHandler,  
    additionalInfo);
```

Additional information string can be anything maximum of 64 characters long. It can be used to send more information about client software like version numbers or build numbers. This information is displayed on “additional info” column in Manage Online Key Usage window.

A handler class extending FloatingLicenseCheckTimerHandler must be created; handler is same as in floating license text. run() method of handler class is called at each timer run. **NOTE: Schedule is determined from “maximum inactive period” value; if it is set to 1 min, timer is scheduled to run each 45 secs. If it is set to any value between 1 and 60, timer is scheduled to run each (inactive period – 1) minutes. If it is set to a value more than 60 minutes, it is scheduled to run each 60 minutes. If license validation status is not valid, timer task return old valid license object until “Maximum Re-Checks Before Drop” value is reached. In case of network/internet connection problems, timer task will wait at least for a second check to return invalid license object.**

Auto License Activation

If validated license requires activation, it can be activated with the static method `LicenseValidator.autoActivate(license)` if license is hosted on Online.License4J system or `LicenseValidator.autoActivate(license, activationServerURL)` method on your server if license is stored your own database server and you have an installed Auto License Generation and Activation Server. With `autoActivate` method, user information can be sent to activation server and license owner information can be modified. It is possible to collect user information and save in database during activation. `LicenseValidator.autoActivate(license, activationServerURL, modifyOnlyOnFirstActivation, fullname, registeredto, email, company, street, telnumber, faxnumber, city, zip, country);` method activates given license and modify given user information on server. There are `autoActivateWithCustomHardwareID` methods for each defined method parameters above; it is used to activate license with a custom hardware ID.

For the Auto License Generation and Activation Server, activation server URL is <http://yourserver.domain.com/algas/autoactivate> or same URL starting with HTTPS if your server has an SSL certificate installed.

The `autoActivate` method returns a License object and it has `ActivationStatus` enumeration with defined constants explained above.

Activation result is obtained with `getActivationStatus()` method. A simple switch statement can be used to check for activation status.

```
switch (license.getActivationStatus()) {  
  
    case ACTIVATION_COMPLETED:  
  
        // TODO  
  
        break;  
  
    case ACTIVATION_SERVER_CONNECTION_ERROR:
```

```
// TODO  
  
break;  
  
case MISMATCH_HARDWARE_ID:  
  
// TODO  
  
break;  
  
case ALREADY_ACTIVATED_ON_ANOTHER_COMPUTER:  
  
// TODO  
  
break;  
  
case MULTIPLE_ACTIVATION_LIMIT_REACHED:  
  
// TODO  
  
break;  
  
}
```

autoActivate method sets user-agent as "License4J HTTPClient" while sending HTTP(S) POST to license server. If there is a firewall or proxy software to access the Internet, and it only allows some user agent strings which are popular web browsers, then runtime library user-agent can be set to any string to access the Internet. Runtime library gets a Java system variable named with name "license4j-user-agent", if it is defined the value of this Java system variable is used before send HTTP(S) requests.

Following method sets Java system property.

```
System.setProperty("license4j-user-agent", "any -browser user agent string");
```

It can also be set on command line with a -D option.

```
java -D license4j-user-agent="any -browser user agent string" -jar YourApplication.jar
```

Manual License Activation

License4J has support for manual activation with a web form if users do not have direct internet connection for auto activation.

License object has a method named `getManualActivationRequestString`, this method returns an encrypted string including license string and computer hardware IDs. This activation request string can be displayed to users for manual activation on <https://online.license4j.com/d/manualactivation> for Online.License4J system and on <http://YourServerAddress/algas/manualactivation> for Auto License Generation and Activation Server if manual activation role is enabled.

When a user paste and submit the form, he/she will get activated license text or activation code based on selection during license generation.

If enabled, manual license activation will be allowed, but you should make a form for your customers to submit activation request. Form should be submitted to <http://YourServerName/algas/manualactivate> and should include a hidden value with name *action* and value *activate*; and license activation request string in an input area with name *linfo*. A sample form is given below.

Form variables "ct" and "fn" are optional. "ct" means Content Type, by default it is "application/l4j". "fn" means File Name and it is "license.l4j" by default. To get a text file directly display on browser set content type to text/plain as in the following form code.

```
<form action="http://YourServerName/algas/manualactivate" method="post">  
  <textarea name="linfo"></textarea>
```

```
<input type="hidden" name="ct" value="text/plain">
<input type="hidden" name="fn" value="your-license.lic">
<input type="hidden" name="action" value="activate">
<input type="submit" value="submit">
</form>
```

Runtime Library has also *getManualActivationRequestStringWithCustomHardwareID* and *getManualActivationRequestStringWithCustomHardwareIDWithUserInformation* methods. First method can be used to generate manual activation request string if a custom hardware ID is used. The second method use a custom hardware ID and user information to modify while activating license. Another method to generate manual activation request string with user information to modify while activating is *getManualActivationRequestStringWithUserInformation*.

Auto License Deactivation

Runtime library has a method named as *autoDeactivate* and it connects to either Online.License4J or any Auto License Generation and Activation Server to perform license deactivation. Activation is basically searched in database and deleted if found.

```
LicenseValidator.autoDeactivate(activatedLicenseObject, serverAddr);
```

The method returns a License object, and ActivationStatus is DEACTIVATION_COMPLETED if deactivation is successfully completed. If there is an error or license deactivation is disabled the related ActivationStatus is obtained.

Manual License Deactivation

License4J has support for manual deactivation with a web form if users do not have direct internet connection for auto activation.

License object has a method named *getManualDeactivationRequestString*, this method returns an encrypted string including license string and some hash codes. This deactivation request string can be displayed to users for manual activation on <https://online.license4j.com/d/manualdeactivation> for Online.License4J system and on

<http://YourServerAddress/algas/manualdeactivation> for Auto License Generation and Activation Server if manual activation role is enabled.

If enabled, manual license deactivation will be allowed, but you should make a form for your customers to submit deactivation request to your Auto License Generation and Activation Server. Form should be submitted to <http://YourServerName/algas/manualdeactivate> and should include a hidden value with name *action* and value *mdeactivate*; and license activation request string in an input area with name *linfo*. A sample form is given below.

```
<form action="http://YourServerName/algas/manualdeactivate"
method="post">
  <textarea name="linfo"></textarea>
  <input type="hidden" name="action" value="mdeactivate">
  <input type="submit" value="submit">
</form>
```

Manual License Modification

License4J has support for manual modification with a web form if users do not have direct internet connection for auto modification.

License object has a method named `getManualModificationRequestString`, this method returns an encrypted string including license string and some hash codes. This modification request string can be displayed to users for manual modification on <https://online.license4j.com/d/manualmodification> for Online.License4J system and on <http://YourServerAddress/algas/manualmodification> for Auto License Generation and Activation Server if manual modification role is enabled.

If enabled, manual license modification will be allowed, but you should make a form for your customers to submit modification request to your Auto License Generation and Activation Server. Form should be submitted to <http://YourServerName/algas/manualmodify> and should include a hidden value with name *action* and value *modify*; and license activation request string in an input area with name *linfo*. A sample form is given below.

Form variables "ct" and "fn" are optional. "ct" means Content Type, by default it is "application/l4j". "fn" means File Name and it is "license.l4j" by default. To get a text file directly display on browser set content type to text/plain as in the following form code.

```
<form action="http://YourServerName/algas/manualmodify" method="post">
  <textarea name="linfo"></textarea>
  <input type="hidden" name="ct" value="text/plain">
  <input type="hidden" name="fn" value="your-license.lic">
  <input type="hidden" name="action" value="modify">
  <input type="submit" value="submit">
</form>
```

License Availability (Blacklist) Check

Runtime library method `checkOnlineAvailability` connects to given server and check if given license or activation exists in server or not. License key, license text, activation code and activated license text can be checked on server.

Method returns an integer: -1 if it cannot connect to server, 0 if license or activation is not found on server, 1 if license or activation is found on server and -2 if license validation status is not valid or license is an activated license with activation status completed or license type is floating license.

```
int i = LicenseValidator.checkOnlineAvailability(  
    publicKeyString,  
    licenseObject,  
    serverAddr,  
    connectionTimeout);
```

Checking for activated license can be helpful if license deactivation feature is enabled and some users deactivated their licenses then activated on some new computers. Since deactivation removes activation from server, license availability check for deactivated licenses will return 0. Therefore, if customer computer is connected to Internet, you can be sure that deactivated license is not used anymore.

Product Update Check

```
checkForUpdate(  
    productEdition  
    currentVersion,  
    licenseServer);
```

method in License object connects to given server and checks for new update version. It returns new (the latest available) version number as double. So that you can display a notification window to user.

Product Update Message Check

```
checkForUpdateMessage (  
    productEdition  
    updateVersion,  
    licenseServer);
```

method in *License* object connects to given server and checks for update notification message for the given version number. It returns message as a string. So that you can display a notification window to user.

License Message Check

```
checkForNewMessage (  
    licenseServer);
```

method in *License* object connects to given server and checks for license messages added to the license. It returns array of string including messages.

License Modification

The method below is used to modify a license. Given license activation status must be `ACTIVATION_COMPLETED`.

```
LicenseValidator.modifyLicense(  
    license,  
    serverAddr,  
    modificationKey);
```

If license to be modified is hosted on Online.License4J, server address argument is not used. `modifyLicense` method returns a *License* object and it has a modification status set, so it can be obtained with `getModificationStatus()` and checked for modification result. The resulting license object has a method named as `getUsedModificationKeys()`, and it returns list of modification keys used to modify license separated with spaces.

License Use Information Update Code

Basic license use information is updated with the following method. It updates last use time and total number of use for the license.

```
LicenseValidator.updateLicenseUseInfo (  
    publickey,  
    license,  
    timeoutValue);
```

Updated values are displayed on License Manager GUI license table (if enabled at header right click pop up menu).

License Use Tracking Code

Following method sends a hash map to license server and store in the database.

```
LicenseValidator.updateLicenseUseTrackingInfo (  
    publickey,  
    license,  
    hashmap,  
    timeoutValue);
```

Following method query license server for values given in hash map.

```
LicenseValidator.queryLicenseUseTrackingInfo (  
    publickey,  
    license,  
    hashmap,  
    timeoutValue);
```

Hash map is updated with the values obtained from server. If some of the given keys in hash map does not exist on the server, then it returns null values for them.

Following code block sends 2 values in a hash map to server for the validated license.

```
HashMap<String, String> map = new HashMap<String, String>();  
map.put("some-key", "some information");  
map.put("another-key", "another information");  
LicenseValidator.updateLicenseUseTrackingInfo (  
    publickey,  
    license,  
    map,  
    timeoutValue);
```

Following code block queries information from server for the keys given in hash map. Previously, null can be defined for the key values; after method successfully return, values will be updated.

```
HashMap<String, String> map = new HashMap<String, String>();  
map.put("some-key", null);  
map.put("another-key", null);  
LicenseValidator.queryLicenseUseTrackingInfo (  
    publickey,  
    license,  
    map,  
    timeoutValue);  
  
System.out.println(map);
```

License4J Runtime Library Obfuscation

Runtime library can be obfuscated to make reverse engineering difficult. Various IDE software has support for obfuscation software like Proguard, and by using ant scripts the process can be automated.

License4J download package includes various examples; example3 folder is a Netbeans project with Proguard and required ant task. When defined ant task runs, it obfuscates both sample application and License4J Runtime Library; finally it also creates an exe file with Launch4j. Sample Launch4j configuration file is found in launch4j folder and can be modified as required.

LicenseValidator Debug Log

When **LicensingDebug** Java system variable is set to true, runtime library version 4.6.1+ prints all exceptions to System.err. ValidationStatus and ActivationStatus return values are already descriptive but sometimes it may be required to see detailed stack trace for the error.

It can be set at startup with `-D` option or in code like:

```
java -DLicensingDebug=true);
```

```
System.setProperty("LicensingDebug", "true");
```

SSL Verification and TrustManager

By default, runtime library will verify hostname and SSL certificate trust. If you want to bypass certificate validation for testing use the following code block just before license activation, deactivation, modification, or online validation.

```
// Create a trust manager that does not validate certificate chains
```

```
TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager() {
    @Override
    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
        return null;
    }

    @Override
    public void checkClientTrusted(X509Certificate[] certs, String authType) {
    }

    @Override
    public void checkServerTrusted(X509Certificate[] certs, String authType) {
    }
}
};

// Create all-trusting host name verifier
HostnameVerifier allHostsValid = new HostnameVerifier() {
    @Override
    public boolean verify(String hostname, SSLSession session) {
        return true;
    }
};

try {
    // Install the all-trusting trust manager
    SSLContext sc = SSLContext.getInstance("SSL");
    sc.init(null, trustAllCerts, new java.security.SecureRandom());
    HttpURLConnection.setDefaultSSLContext(sc.getSocketFactory());
    // Install the all-trusting host verifier
    HttpURLConnection.setDefaultHostnameVerifier(allHostsValid);
} catch (Exception ex) {
    ex.printStackTrace();
}
```

License4J Development Library Integration

Required methods are available in License4J development library to generate licenses programmatically. The process starts with generating your keypair, then LicenseText or LicenseKey objects are initialized, features set and license string is generated. Sample code are given below. JavaDoc documentation is provided within License4J download package.

Generate KeyPair:

```
LicenseKeyPair keypair = LicenseManager.createKeyPair(  
    "key pair descriptive name",  
    "some description",  
    LicenseKeyPair.KEYPAIR_ALGORITHM_RSA,  
    LicenseKeyPair.SIGNATURE_ALGORITHM_SHA512withRSA,  
    LicenseKeyPair.KEYSIZE_1024);
```

Key pair and signature algorithms are defined as static variables in LicenseKeyPair.

Create LicenseText or LicenseKey object and set required features:

```
LicenseText licenseText = new LicenseText();  
licenseText.setUserFullName("some user");  
licenseText.setUserEMail("email@domain.com");
```

Generate License:

```
String licenseString = LicenseManager.generateLicenseText(keypair, licenseText);
```

After this three step, you will have a license string, copy it in a file and send to your customer. License4J package also includes a small utility for reading and writing to text files, you can write your license string to a file as below.

```
FileUtils.writeFile("c:\\license.txt", licenseString);
```

License4J Development library requires a valid license and it must set as java system properties in source code. If License Manager GUI is run same computer, this code can be obtained with "Display Activated License" button in License4J license window in Help menu item.

If License Manager GUI is not used on computer, development library can be used to activate the license. To install and activate the license you have to use `com.license4j.License4JLicense` class in command line. To install 30 days valid trial license and activate use the command below.

```
java -classpath LICENSE4J-Development-Librar.jar com.license4j.License4JLicense getTrialLicense
```

This command connects to Online.License4J and activates trial license for running computer, and it returns license key and activated license text.

To install your purchased license, use the command below.

```
java -classpath LICENSE4J-Development-Librar.jar com.license4j.License4JLicense getActivationResult  
type-your-license-key
```

After getting your license text you have to set Java system variables which development library checks for. The license key below is the trial license.

```
System.setProperty("license4j.license.key", "<license key> ");
```

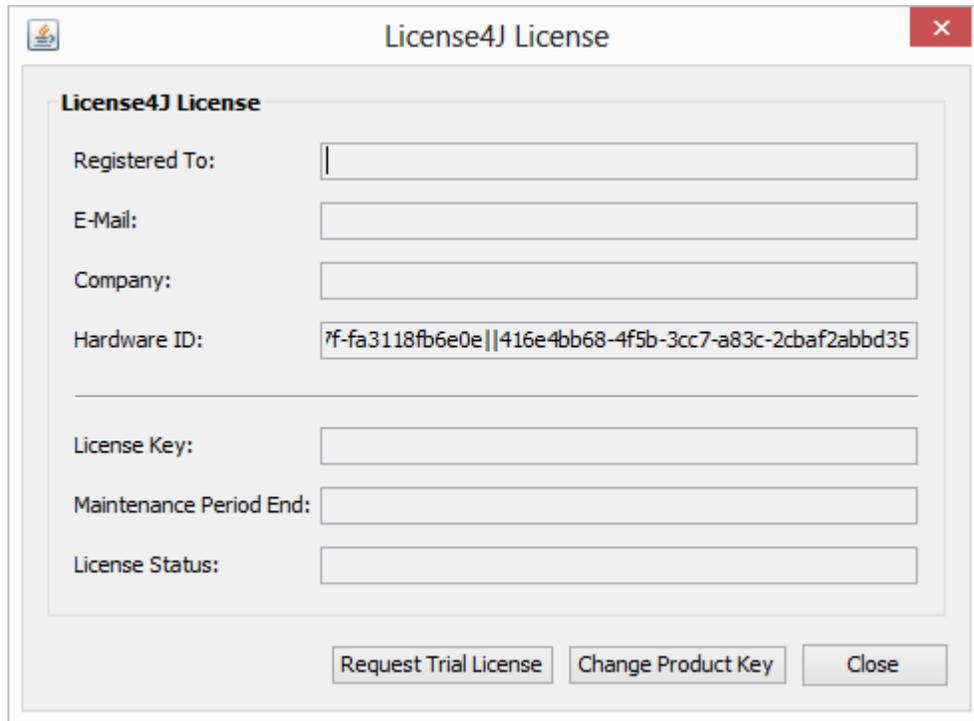
```
System.setProperty("license4j.license.text", "<license text obtained with getActivationResult  
or getTrialLicense">
```

These properties should be set before generating license string. Finally, a complete running example is given below.

```
System.setProperty("license4j.license.key", "<license key>");  
System.setProperty("license4j.license.text", "<license text>");  
LicenseKeyPair keypair = LicenseManager.createKeyPair(  
    "key pair descriptive name",  
    "some description",  
    LicenseKeyPair.KEYPAIR_ALGORITHM_RSA,  
LicenseText licenseText = new LicenseText();  
licenseText.setUserFullName("some user");  
licenseText.setUserEMail("email@domain.com");  
String licenseString = LicenseManager.generateLicenseText(keypair, licenseText);  
FileUtils.writeFile("c:\\license.txt", licenseString);
```

License4J License Manager License

When License Manager first run licensing window appears as below.



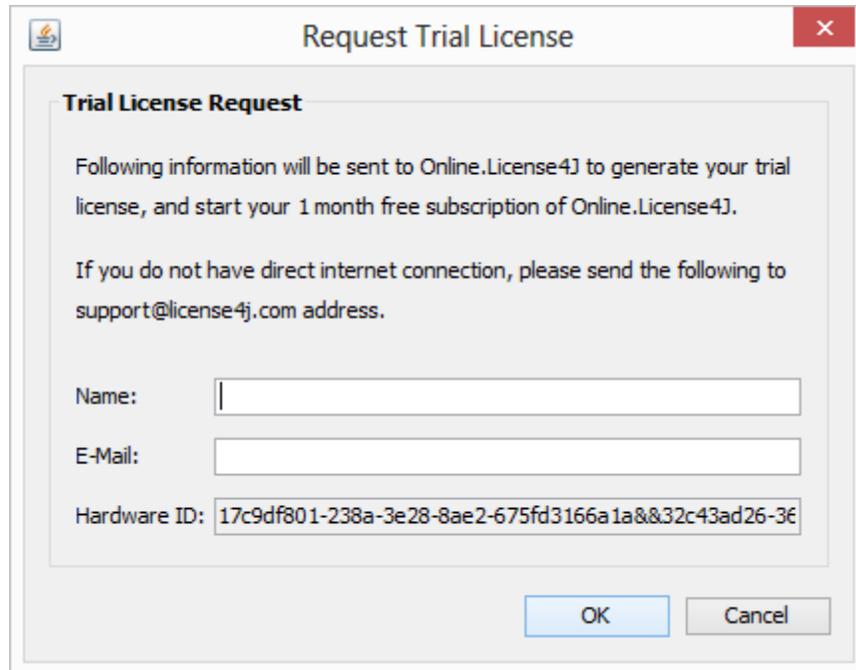
The screenshot shows a window titled "License4J License" with a close button in the top right corner. The window contains the following fields and buttons:

- Registered To:** An empty text input field.
- E-Mail:** An empty text input field.
- Company:** An empty text input field.
- Hardware ID:** A text input field containing the value "7f-fa3118fb6e0e||416e4bb68-4f5b-3cc7-a83c-2cbaf2abbd35".
- License Key:** An empty text input field.
- Maintenance Period End:** An empty text input field.
- License Status:** An empty text input field.
- Buttons:** Three buttons are located at the bottom: "Request Trial License", "Change Product Key", and "Close".

If you want to use trial period for testing, click on Request Trial License button to get your trial license and Online.License4J account online. If you do not have internet connection contact us.

Trial period has one restriction. All licenses are created for name "LICENSE4J Trial.

Trial request window asks for name and e-mail address for Online.License4J account creation, your e-mail will be used as username. Trial license will be activated with your hardware id.



Request Trial License

Trial License Request

Following information will be sent to Online.License4J to generate your trial license, and start your 1 month free subscription of Online.License4J.

If you do not have direct internet connection, please send the following to support@license4j.com address.

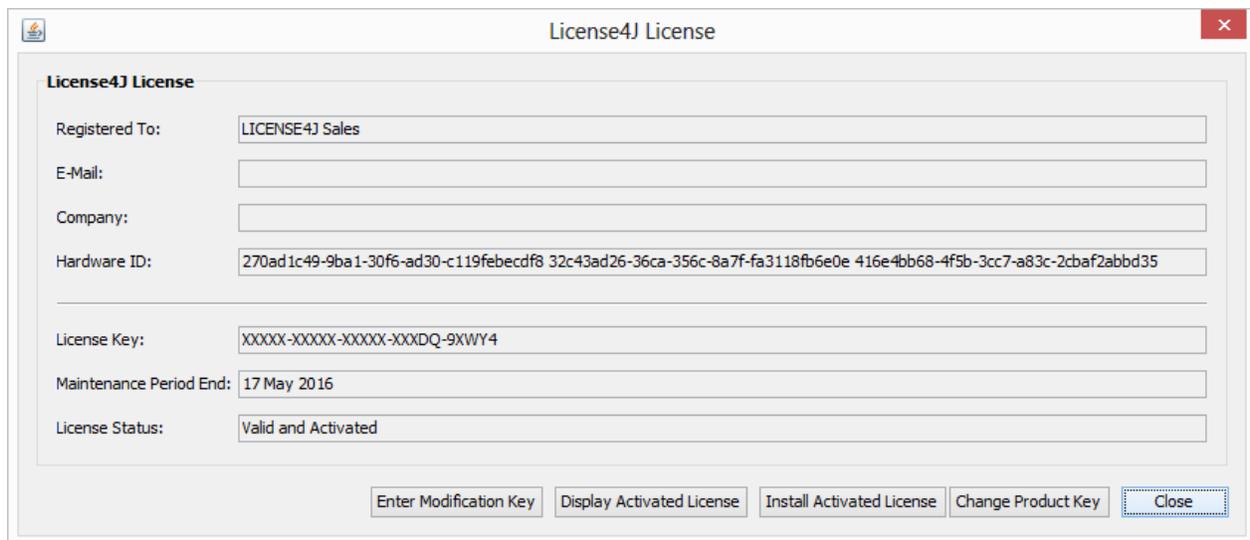
Name:

E-Mail:

Hardware ID: 17c9df801-238a-3e28-8ae2-675fd3166a1a&&32c43ad26-3e

OK Cancel

After trial license is activated, licensing window will display license details.



License4J License

Registered To: LICENSE4J Sales

E-Mail:

Company:

Hardware ID: 270ad1c49-9ba1-30f6-ad30-c119febecdf8 32c43ad26-36ca-356c-8a7f-fa3118fb6e0e 416e4bb68-4f5b-3cc7-a83c-2cbaf2abbd35

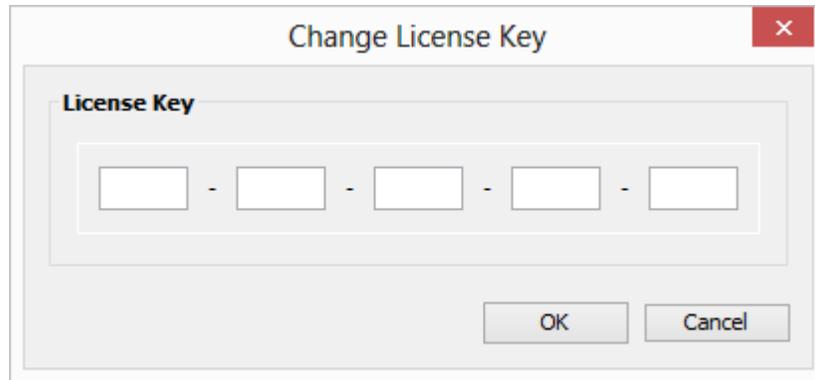
License Key: XXXXX-XXXXX-XXXXX-XXXDQ-9XWY4

Maintenance Period End: 17 May 2016

License Status: Valid and Activated

Enter Modification Key Display Activated License Install Activated License Change Product Key Close

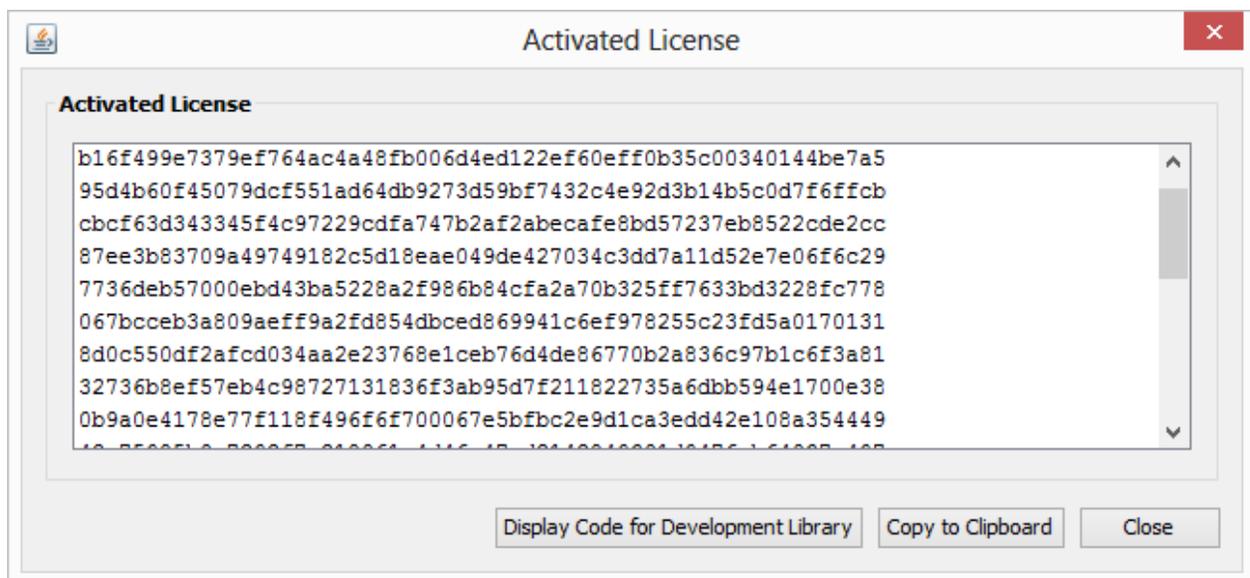
When you purchase License4J License Manager, you will have a license key sent by payment processor. Click on Change Product Key button to enter your license key.



After license key validation you have to activate your copy of License Manager within 30 days.

When a maintenance license is purchased, you will get a modification key which you should enter with "Enter Modification Key" button. It will extend maintenance period for 1 year.

Activated license text and code to use in Development Library is displayed with Display Activated License button.



License Manager GUI Tool Error Logging

License Manager GUI logs errors and exceptions to a log file on disk. Log file is named as *gui-log.txt* and can be found in current user's application data folder in windows.

(e.g. C:\Users\\AppData\Local\License4J License Manager\logs\gui-log.txt)

On Linux and Mac OS log file is saved in the current path which License Manager GUI is running.

End of document.