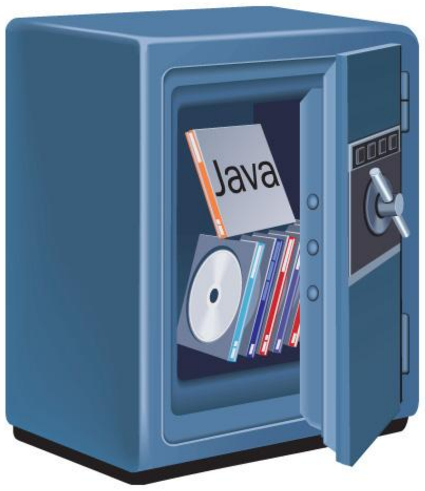
jProductivity LLC

Protect your investments with Protection!



Protection!™

Licensing Toolkit

v5.5.1

What's New

http://www.jproductivity.com

Revision

319 - 11/5/21

Contents

Chapter 1

About Protection! Licensing Toolkit

Protection! - is a powerful multi-platform Licensing Toolkit and License Manager that provides the ability to add licensing into custom applications, services or components only allowing the permitted use according to the supplied license. Protection! uses high encryption technology and provides easy integration for software developers even for cross platform products while being non-invasive for end users. Protection! License Manager offers a versatile solution for any licensing model.

Protection! allows software vendors, publishers and developers to:

- Add licensing to Web, Enterprise, Server, Cloud and Desktop applications.
- Provide users with the trial versions of their products.
- Significantly minimize or completely reduce unauthorized use of their applications and therefore dramatically increase company revenue.
- Increase revenue streams by implementing various licensing models while maintaining single code base and therefore offering higher flexibility to their customers.
- Track license usage and manage their customer base.

With the help of Protection! Licensing Toolkit and License Manager software publishers and developers are able to:

- Easily and economically increase scalability of their applications.
- Have full control over all aspects of licensing and tracking.
- Enable additional revenue generation models.
- Quickly and easily adapt Protection! to any business model.
- Implement robust licensing features into their applications while providing an easy and non-invasive environment to their end users.
- Solve today's complex licensing challenges.
- Prevent users' ability to make either unintentional or unauthorized copies.
- Free development resource to work on the core functionality that makes their application great.

Protection! 5 is immediately available for download or purchasing. Current customers, having active subscription, should contact us to receive updated licenses for this new version. Customers of previous versions of Protection! please contact us to receive upgrade details.

Chapter 2

What's New in Protection! Toolkit/Framework

2.1 Automatic License Backups

License Reader is improved to automatically back up current licenses when applying new ones. This allows to:

- Preserve valid licenses if improper or corrupted licenses were applied
- Use backed-up licenses during looking for licenses of previous versions to support "upgrade" kind of licenses

2.2 Support of License Kinds

There is a new trait added to licenses – license kind. Protection! supports the following License kinds (defined by the `License.licenseKind` property):

- `License.KIND_NEW` – licenses of that kind do not impose additional requirements to be considered valid. All the licenses generated by pre v5 versions of Protection! will be considered to be "new" ones.
- `License.KIND_UPGRADE` - licenses of that kind require existence of certain licenses of previous versions to be considered valid.

For "upgrade" licenses it is possible to specify a minimum required version of previous license and, optionally, required product editions via the `prevLicenseMajorVersion`, `prevLicenseMinorVersion` and `prevLicenseProductEditions` properties of a License.

License Host checks "upgrade" licenses according to the following process:

1. Searching for the licenses of previous version:
    - In license backups.
    - In explicitly specified locations.
2. Iterating through found old licenses (if any) and comparing their Major and Minor versions and, optionally, their Product Editions with corresponding values of a new License.
3. Stopping with OK result if a suitable license is found.
4. Calling the `LicenseHostIssueResolver.resolveLicenseCantUpgrade(…)` method if no suitable license of previous version was found. For a GUI application the License Upgrade Wizard can be invoked to let the users specify a correct location of previous version license.
5. Setting the `License.STATE_INVALID` to the license if no suitable license of previous version was found and issue resolver was unable to resolve that issue.

It is possible to specify a list of locations to search for old licenses either in the code via the `LicenseUpgradeChecker` instance like:

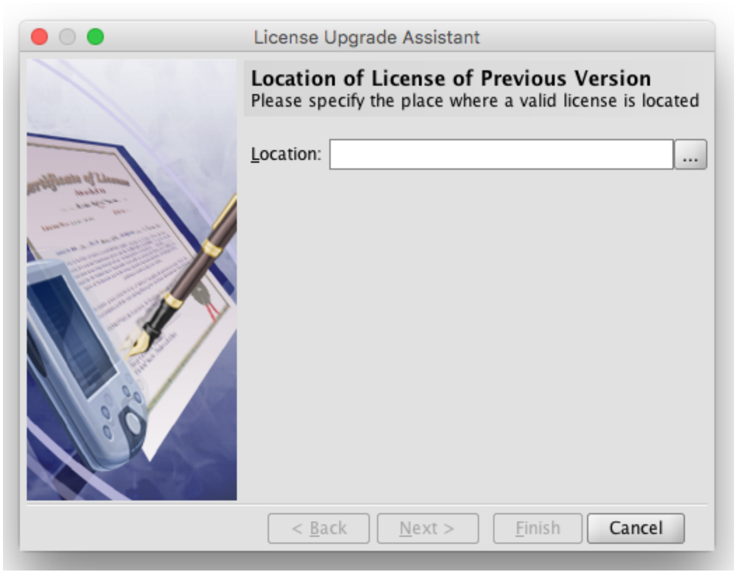
```
licenseHost.getLicenseUpgradeChecker().addPrevLicenseLocation(
        new LicenseLocation(".DemoCalc2", "DemoCalc.key", true, true));
```

or by changing the product in the Developer Control Center via the *"Edit Product | Integration | License Upgrade"* (certainly a Launcher should be rebuilt to apply changes).

License Host uses an instance of `LicenseUpgradeChecker` to delegate checking of "upgrade" licenses. It is possible to extend or change default check logic by providing a custom implementation of the `LicenseUpgradeChecker`. This can be done by sub-classing the `LicenseUpgradeChecker` class, changing the logic by overriding the `checkLicenseVersions` method and setting a new instance of custom `LicenseUpgradeChecker` to License Host via a `setLicenseUpgradeChecker` call.

If licenses of previous versions cannot be read due to encryption change then a custom implementation of `LicenseUpgradeChecker` must be used. To handle encryption change the `initLicenseReader(LicenseReader)` method should be overridden and its implementation should properly initialize passed License Reader (by specifying proper security algorithm and decrypt key bytes) to make it capable of reading licenses of previous versions.

There is a new "License Upgrade Assistant" to allow finding of licenses of previous versions, which will be invoked automatically when searching for the licenses of previous version failed.


License Upgrade Assistant

Location of License of Previous Version

Please specify the place where a valid license is located

Location:

2.3 Improved File Secret Storage

It is possible to use strong encryption for File Secret Storages. It is OFF by default to maintain compatibility with previous versions of Protection!. Strong encryption can be turned ON by using the `setUseStrongEncryption(…)` method or by creation of a File Secret Storage using appropriate constructor.

If it is possible to run several instances of the application on the same computer – it may cause File Secret Storages to become corrupt. To control and prevent concurrent modifications of the Secret Storages it is possible to let File Secret Storages to utilize lock files (`.lck`). This mode can be turned ON by using `setUseLockFile(…)` method or by creation of a File Secret Storage using appropriate constructor.

2.4 Jar File Secret Storage

This implementation of the `SecretStorage` interface stores its data as an entry in a local archive. A new instance of `JarFileSecretStorage` can be created using one of its constructors:

```
SecretStorage secretStorage = new JarFileSecretStorage(".DemoCalc",
    "DemoCalc.jar", true, "com/demo/democalc.dat")
```

Using that kind of Secret Storage allows hiding sensitive data in a `.jar` file that may look like just an application library. All other major attributes of `JarFileSecretStorage` are the same as of `FileSecretStorage`.

2.5 Briefcase Mode for Named Licenses

Sometimes it is feasible to let users to continue utilizing Floating or Named licenses offline without the need to connect to the Licensing Server (AKA briefcase mode). It can be good for temporary working outside of the corporate network e.g. traveling when there is no access to the Licensing Server.

To allow that scenario – a license should be checked out for offline use. It can be done via `ProtectionLauncher.checkoutLicense()` method. A license must be explicitly marked to allow checking it out. License's `checkoutPeriod` property specifies the number of days the license can be checked out and then used offline. Zero or negative values for this property disables ability to use a license offline. It is possible to check if a license can be checked out via the `ProtectionLauncher.canCheckoutLicense()` method.

After the license is checked out – the Licensing Server keeps license lock for that user for entire allowed checkout period. That makes a checked out license to look like a kind of temporary Named license for a limited period of time. License lock for the checked out floating license will be automatically released by the Licensing Server after the end of checkout period, making it available for other users. Though, it is possible to release the checked out license anytime earlier by having a valid connection to the Licensing Server and calling the `ProtectionLauncher.terminateLicensePass()` method.

2.6 Centralized Tracking of License Usage Limit

For licenses having the `License.USER_LICENSING_FLOATING_LS` model the usage quota will be distributed among all the users using certain license. This quota will be tracked by the Licensing Server. If that quota is reached – the license will not get the `License.STATE_EXPIRED` state. Instead the

`License.STATE_NOT_LOCKED` state will be assigned to the license and a corresponding `licenseNotLocked` event will be fired.

2.7 Tracking of CPU Cores, Memory and Usage Limits

The Licensing Server tracks all the licenses in use and calculates total values for the Number of Copies, CPU Core, Memory and Usage for each license and then compares it with corresponding limits specified in the license. If any of those limits is reached – a license lock will not be acquired therefore a user will not be able to use that license. The following are license properties which define those limits:

- `License.PROPERTY_CPU_LIMIT`
- `License.PROPERTY_MEMORY_LIMIT`
- `License.PROPERTY_USAGE_LIMIT`

Note

Applications report total physical memory to be tracked by the Licensing Server; it is neither free memory nor memory available to JVM. Obtaining total physical memory for applications requires deployment of OSHI library and its dependencies. See *deploy.html* for more information.

Utilizing such limits can be very useful for licensing of server-side, Cloud or clustered applications.

2.8 Checking the Status of Licensing Server

In some cases a Licensing Server can go offline and continue staying offline for some extended period of time; it would be either casual maintenance, hardware or software failure. In that case, the users trying to work with their applications, which use Floating or Named licenses, will see a generic “unable to connect” message. To improve the customer experience Protection! includes an ability to let the users know what is going on with the Licensing Server and when it is going to be back online.

It is done by placing a specially prepared server status XML file to some highly available location and instructing Protection! to use the data from this file in case of inability to lock licenses due to system errors or connection failure. Format of a server status file is simple and straightforward:

```
<?xml version="1.0" encoding="UTF-8"?>
<resource_status status="Down">
  <message>Server is on maintenance</message>
  <long_message>
    <![CDATA[<b>Server is down</b>. Hard drive has failed. Service is expected to be
up by 5 AM.]]>
  </long_message>
</resource_status>
```

The `status` attribute of the `resource_status` element specifies the status of the server. Any values other than “OK”, written in any case, mean that server is either offline or malfunctioning. If the server is not OK – it is possible to specify the `message` and the `long_message` attributes to better describe the issues and guide the users wait for a certain time, use some workaround or go to a Licensing Server running on a different location.

It is possible to specify location of a server status file:

1. In Developer Control Center via the *"Edit Product | Licensing Server | Server Status URL"*.

```
In the code via a LicenseHost.getLicensingServiceSupport().
setServerStatusURL() call.
```

2.9 Support of New License Lock Attributes

There are new attributes to lock licenses to:

- `LicenseHostPro.LOCK_CPU_ID` - Option specifies that license should be locked to the CPU ID.
- `LicenseHostPro.LOCK_MOTHERBOARD_ID` - Option specifies that license should be locked to the motherboard ID.
- `LicenseHostPro.LOCK_CUSTOM_ATTRIBUTE` - Option specifies that license should be locked to the custom attribute provided by a protected application.

Note

Locking licenses to CPU or Motherboard ID's requires deployment of OSHI library and its dependencies. See *deploy.html* for more information.

The *License Host* uses an instance of `LicenseHostProContext` to supply all the system information attributes. So to provide different values for those attributes, or to provide values for custom locking attribute, a custom implementation of `LicenseHostProContext` should be used. As sample code is below:

```
LicenseHostPro.setContext(new Java6LicenseHostProContext(){
  public String getCustomLockAttribute() {
    //todo: obtain custom attribute here
    return customAttribute;
  }});
```

2.10 FIPS Support

There is new support of using FIPS certified security providers. To use it:

1. Convert cipher keys for your product to a provider independent form (X509 and PKCS8) in Developer Control Center (*"Products Screen" -> "Edit | Convert Encryption Keys"*)
2. Alter your source code to apply updated decryption key bytes
3. Rebuild launcher (if used)
4. Rebuild digest (if used)
5. Use a FIPS certified provider. It is possible to use either bundled Bouncy Castle FIPS provider or use any other FIPS certified compatible provider of your choice.

Note

If you need to support older versions of your applications that make use of unconverted cipher keys – you must use at least Protection! 5.2.5. This version preserves some required information during conversion of the keys. Without that information licenses deactivation may fail for older versions of your application.

If you already have your keys converted using earlier than 5.2.5 version of Protection! – you have to use older (original) version of your Products Storage, make conversion again and use this Products Storage for all the future developments.

To use Bouncy Castle FIPS provider: use bundled `bc-fips-1.0.1.jar` library instead of `bcprov-jdk14-138.jar` (or any other newer version of the library).

Note

Make sure you have only one Bouncy Castle library (either `bc-fips-1.0.1.jar` or `bcprov-jdk14-138.jar`) in your class path. If you have both – your application will be unable to use Bouncy Castle security provider and may fail.

To use another FIPS certified security provider:

1. Check that provider of your choice supports `RSA/ECB/PKCS1PADDING` cipher
2. Consult documentation for JCE and for that provider on how to install and configure it.

2.11 Support of Native Applications *v5.2*

Though Protection! is primary intended to add protection and licensing support to Java applications, now Protection! can be easily integrated with native e.g. C/C++ applications.

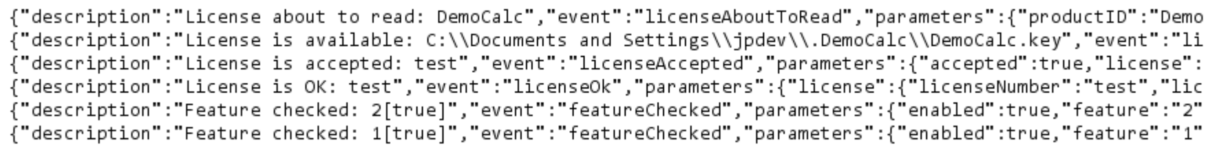
To allow licensing of native applications Protection! Developer includes a special Java application – *Protection Support*. This application is represented by a single application archive - `bin/ProtectionSupport.jar` and dependant libraries located in the `/lib` folder. When *Protection Support* application launched it:

1. Locates a *Launcher* and loads the entire licensing configuration from it.
2. Starts license reading and checking procedure.
3. Prints all the licensing events to standard output stream in JSON format.
4. Prints all the errors (exceptions) to standard error stream.
5. Either exits right after completion of license check or continue checking of license periodically.
6. Listens for commands by reading standard input stream.

Tip

You can run *Protection Support* application in console via command line and see its output to better understand how it is functioning. For example:

```
java –jar ProtectionSupport.jar DemoCalc.launcher
```



```
{"description":"License about to read: DemoCalc","event":"licenseAboutToRead","parameters":{"productID":"Demo
{"description":"License is available: C:\\Documents and Settings\\jpdev\\.DemoCalc\\DemoCalc.key","event":"li
{"description":"License is accepted: test","event":"licenseAccepted","parameters":{"accepted":true,"license":
{"description":"License is OK: test","event":"licenseOk","parameters":{"license":{"licenseNumber":"test","lic
{"description":"Feature checked: 2[true]","event":"featureChecked","parameters":{"enabled":true,"feature":"2"
{"description":"Feature checked: 1[true]","event":"featureChecked","parameters":{"enabled":true,"feature":"1"
```

To utilize *Protection Support* in your application:

1. Deploy *Protection Support* with your application.
2. Deploy JRE with your application or make sure it is present in the system.
3. Generate a *Launcher* and deploy it with your application.
4. Launch a *Protection Support* process.
5. Catch *Protection Support* error output and print or log it, for example.
6. Catch *Protection Support* standard output to be notified of all licensing events.
7. Decode licensing events from JSON format and handle all the events of interest to enable, disable or fine tune parts of your application's functionality according to a supplied license.
8. Transfer some commands to Protection Support by putting them to its input.

Please check the Chapter 3 "Adding Protection! Support to Native Applications" in the Protection! Developer Guide for more information.

2.12 Extended License Host Listener *v5.2*

There is a new `LicenseHostListenerExt` interface that provides two additional methods that allows listening about license checking start and finish events:

- `licenseCheckStarted` – called when License Host started checking license
- `licenseCheckFinished` – called when License Host stopped checking license

To utilize those two new methods you should implement and use the `LicenseHostListenerExt` instead of `LicenseHostListener` and add some code to new methods. Note: if you already have an implementation of the `LicenseListener` interface you have to add implementations for new methods as this interface extends the `LicenseHostListenerExt` interface now.

2.13 Extended License Reader Listener *v5.2*

There are two new methods in the `LicenseReaderListenerExt` interface that allows listening for license reading start and finish events:

- `licenseReadingStarted` – called when License Reader started reading license
- `licenseReadingFinished` – called when License Reader stopped reading license

2.14 Ability to Activate Applications When Needed *v5.2*

By default, Protection! will not try to activate (bring to foreground) a protected application when showing various dialogs and messages in response to Protection! events. What actually happens in those cases is platform specific so the users may miss those dialogs and messages when the application is not active and obscured by other windows. Now it is possible to instruct Protection! to automatically activate the application when showing dialogs and messages. This can be done either by calling `BasicLicenseWizard.setRequestForeground(true)` or by setting the system property `com.jp.protection.gui.dialogs.BasicLicenseWizard.requestForeground=true` if you do not want to rebuild the application. Note: Java 9 and greater is required to allow this functionality.

2.15 New Methods to Get License Information in Textual Form *v5.2*

License Host now includes several new methods to allow getting some license information in textual human-readable form:

- `LicenseHostPro.getLicenseStateInfo` – provides textual representation on the license state
- `LicenseHostPro.getLicenseTypeInfo` -- provides textual representation on the license type
- `LicenseHostPro.getLicenseInfo` -- provides textual representation on entire license

2.16 Ability to Specify Operations Allowed to Resolve Missing License Case *v5.2.1*

Now it is possible to specify which operations are allowed to a resolve missing license case. I can be done by calling the `setResolveLicenseMissingOperations(…)` method of Protection Launcher where operations is a combination of `LicenseReaderIssueResolver.OPERATION_` constants. Note: operations not supported by particular Issue Resolver implementation will be ignored.

2.17 Ability Resolve Missing License Case by Getting License from Licensing Server *v5.2.1*

Now it is possible to resolve missing license case by obtaining a license from Licensing Server automatically with no user involvement needed. It can be done by calling the `ProtectionLauncher. setResolveLicenseMissingOperations(…)` method and specifying a `LicenseReaderIssueResolver. OPERATION_GET_LICENSE_FROM_LS` operation. For example:

```
ProtectionLauncher.setResolveLicenseMissingOperations(OPERATION_GET_LICENSE_FROM_LS |
    OPERATION_USE_LICENSING_ASSISTANT)
```

To let this operation function properly either:

- Licensing Server address should be specified for the Product in Developer Control Center via the *Edit Product | Integration | Licensing Server*.
- Licensing Server should be discoverable in network.
- Licensing Server address should be explicitly specified for a License Reader Issue Resolver: `HeadlessLicenseReaderIssueResolver.setLicensingServerAddress(…)`.

2.18 Java 17 Support *v5.5.1*

Java 17 is fully supported started from v5.5.1. Though, there a tweak may be required, if you run Protection! GUI tools via command line or if you are making your own Licensing Server installers – the following option should be passed to VM:

```
--add-exports java.desktop/sun.awt=ALL-UNNAMED
```

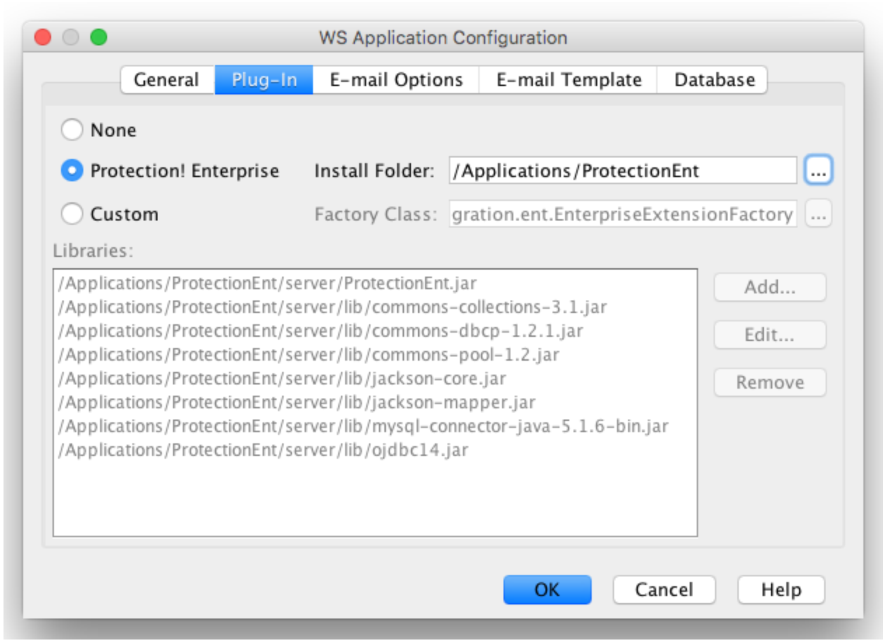
If that option is omitted – an exception will be logged once; all the apps will be functional though.

Chapter 3

What's New in Developer/Sales Control Center

3.1 Improved the "WS Application Configuration" Dialog

The dialog is improved to simplify configuration of Protection! Enterprise. Now there is no need to manually add all the required libraries and copy a proper value for the "Factory Class" – only an install folder for Protection! Enterprise should be specified; all the required attributes will be populated automatically.



Install Folder: /Applications/ProtectionEnt

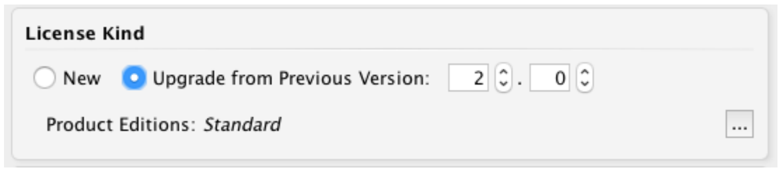
Factory Class: gration.ent.EnterpriseExtensionFactory

Libraries:

```
/Applications/ProtectionEnt/server/ProtectionEnt.jar
/Applications/ProtectionEnt/server/lib/commons-collections-3.1.jar
/Applications/ProtectionEnt/server/lib/commons-dbcp-1.2.1.jar
/Applications/ProtectionEnt/server/lib/commons-pool-1.2.jar
/Applications/ProtectionEnt/server/lib/jackson-core.jar
/Applications/ProtectionEnt/server/lib/jackson-mapper.jar
/Applications/ProtectionEnt/server/lib/mysql-connector-java-5.1.6-bin.jar
/Applications/ProtectionEnt/server/lib/ojdbc14.jar
```

3.2 Entering License Kind Attributes

The "License Screen" of Protection! Control Center now includes ability to specify license Kind and all related attributes like a version and supported Product Editions.



License Kind

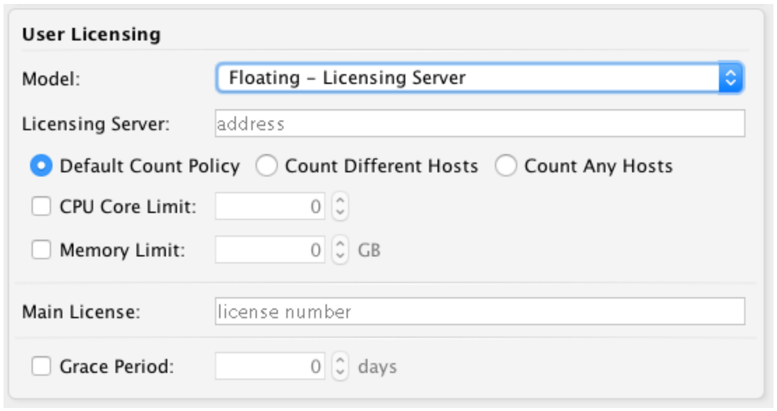
New [✓ Upgrade from Previous Version:] 2 . 0

Product Editions: *Standard*

3.3 Entering New Attributes for Floating and Named LS Models

The "License Screen" of Protection! Control Center now includes ability to specify all new licenses attributes for extended Floating and Named LS user licensing models.

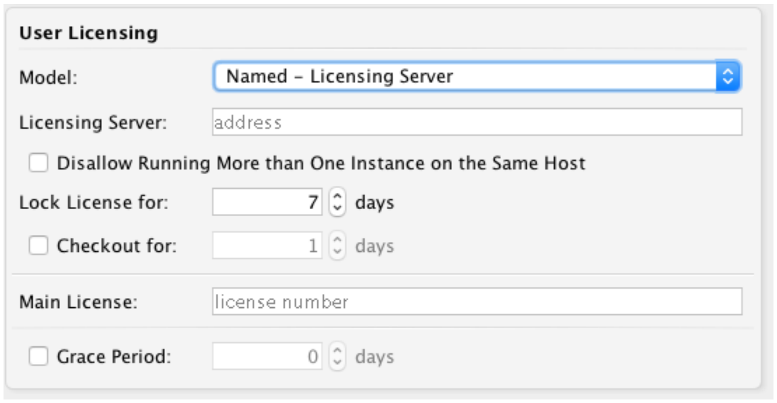
For the *"Floating – Licensing Server"* model it allows entering the *"CPU Core Limit"*, *"Memory Limit"* and *"Main License Number"* attributes.



User Licensing

Model:	Floating – Licensing Server
Licensing Server:	address
[✓ Default Count Policy] Count Different Hosts, Count Any Hosts	
☐ CPU Core Limit:	0
☐ Memory Limit:	0 GB
Main License:	license number
☐ Grace Period:	0 days

For the *"Named – Licensing Server"* model it allows entering the *"Lock License for (Period)"* and *"Checkout for (Period)"* attributes.

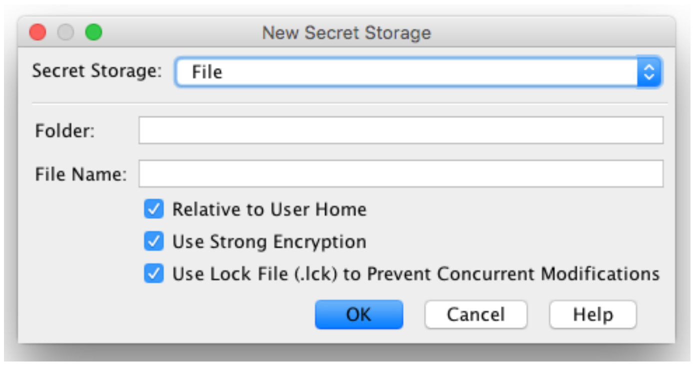


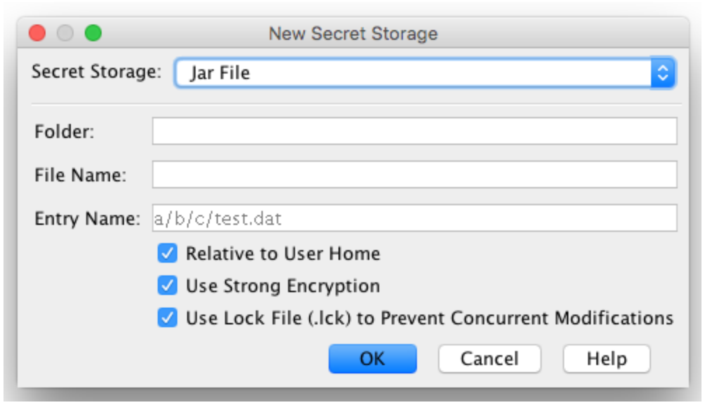
User Licensing

Model:	Named – Licensing Server
Licensing Server:	address
☐ Disallow Running More than One Instance on the Same Host	
Lock License for:	7 days
☐ Checkout for:	1 days
Main License:	license number
☐ Grace Period:	0 days

3.4 Updated the "Edit Product | Integration | Secret Storages" Page

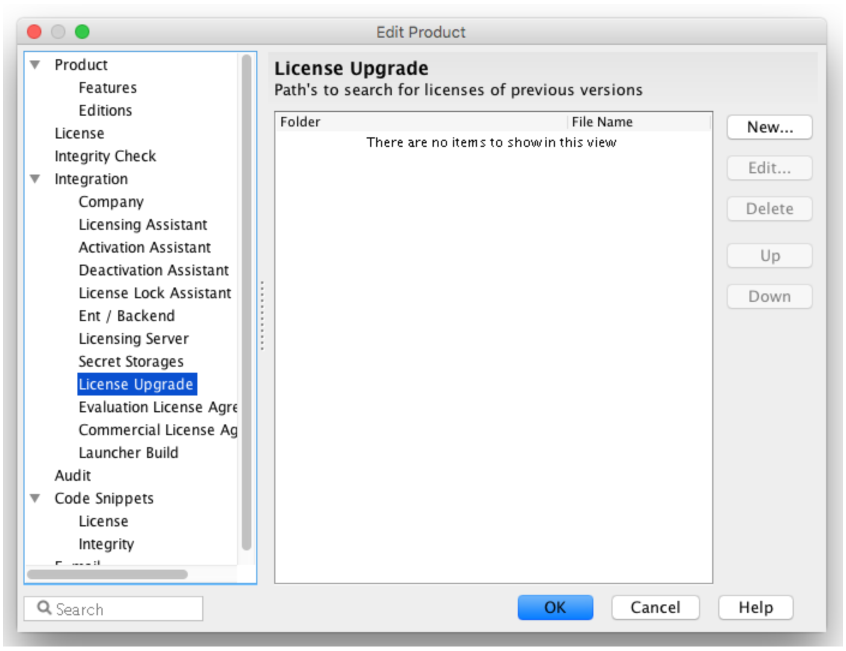
Updated the *"Edit Product | Integration | Secret Storages"* page now includes ability to specifying new types of Secret Storages and allows specifying all new attributes for particular Secret Storage types.





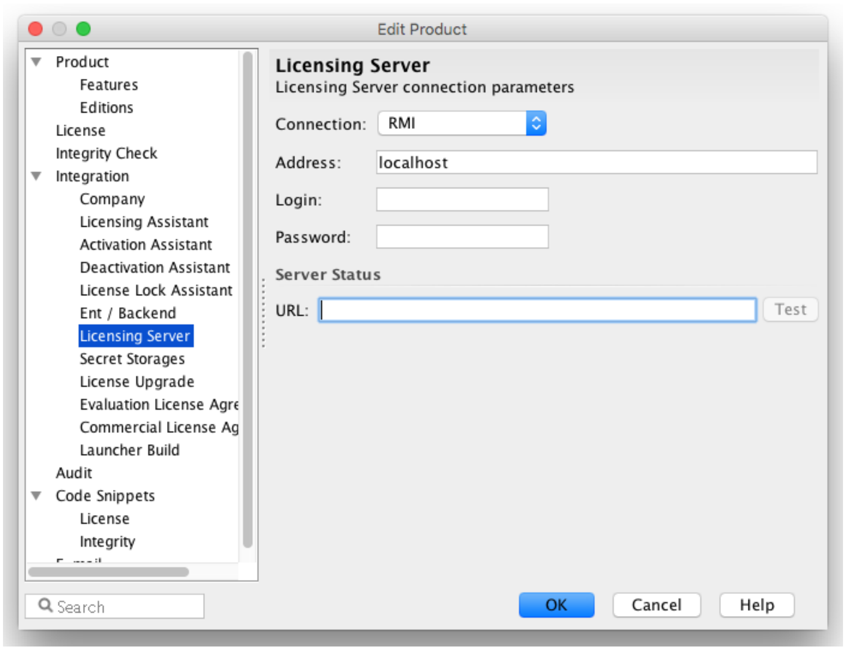
3.5 New the "Edit Product | Integration | License Upgrade" Page

There is a new *"Edit Product | Integration | License Upgrade"* page to allow specifying license upgrade options like the list of paths to look for licenses of previous versions.



3.6 Updated the “Edit Product | Integration | Licensing Server” Page

Updated the *“Edit Product | Integration | Licensing Server”* page now allows to specify the “Server Status URL” and allows to immediately test it and preview its contents.



Licensing Server
Licensing Server connection parameters

Connection: RMI
Address: localhost
Login:
Password:

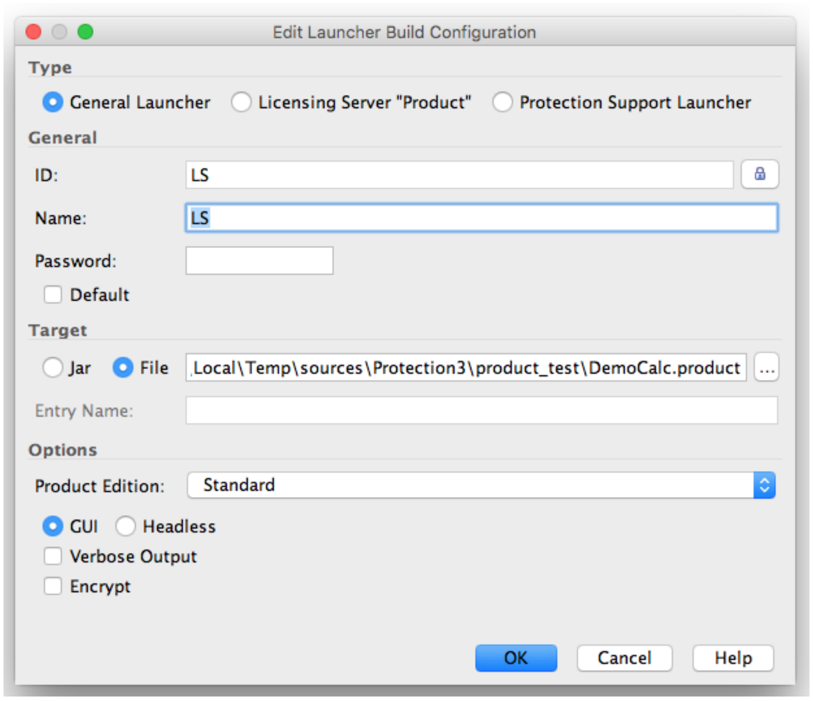
Server Status
URL:

3.7 Improved the "Edit Launcher Build Configuration"

Improved the *"Edit Launcher Build Configuration"* Dialog now allows simplifying building launchers for different kind of products:

1. General Launcher
2. Licensing Server “Product”
3. Protection Support Launcher (*since v.5.2*)

So when any option is selected – all the required attributes will be automatically propagated.


Edit Launcher Build Configuration

Type

[✓ General Launcher] Licensing Server "Product" Protection Support Launcher

General

ID: LS

Name: LS

Password:

Default

Target

Jar [✓ File] Local\Temp\sources\Protection3\product_test\DemoCalc.product

Entry Name:

Options

Product Edition: Standard

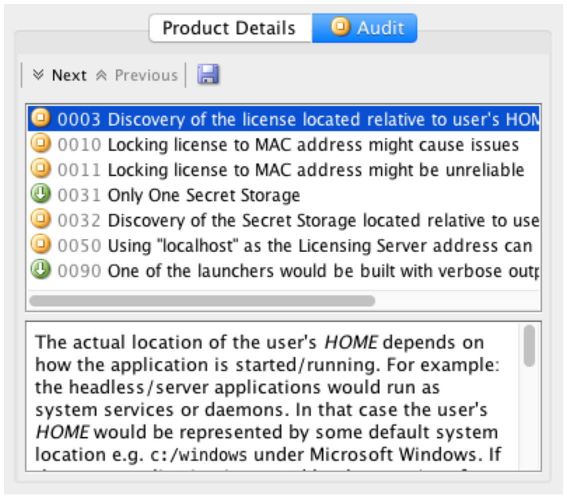
[✓ GUI] Headless

Verbose Output

Encrypt

3.8 Ability to Save Audit Results to a File

The Product Audit pane now includes an ability to save all audit results to a HTML file. This feature provides a more convenient way to share issues, discuss and resolve them. To save audit results – use the *"Save"* button on the Product Audit toolbar.

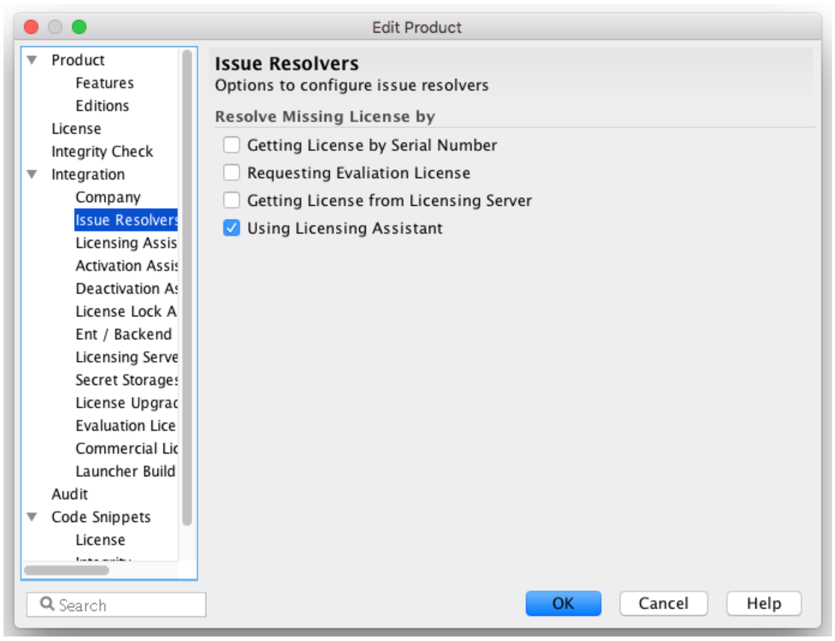


0003 Discovery of the license located relative to user's HOM
0010 Locking license to MAC address might cause issues
0011 Locking license to MAC address might be unreliable
0031 Only One Secret Storage
0032 Discovery of the Secret Storage located relative to use
0050 Using "localhost" as the Licensing Server address can
0090 One of the launchers would be built with verbose outp

The actual location of the user's *HOME* depends on how the application is started/running. For example: the headless/server applications would run as system services or daemons. In that case the user's *HOME* would be represented by some default system location e.g. `c:/windows` under Microsoft Windows. If

3.9 New the "Edit Product | Integration | Issue Resolvers" Page *v5.2.1*

There is a new *"Edit Product | Integration |Issue Resolvers"* page to allow specifying Issue Resolvers options like operations allowed to resolve license missing case.



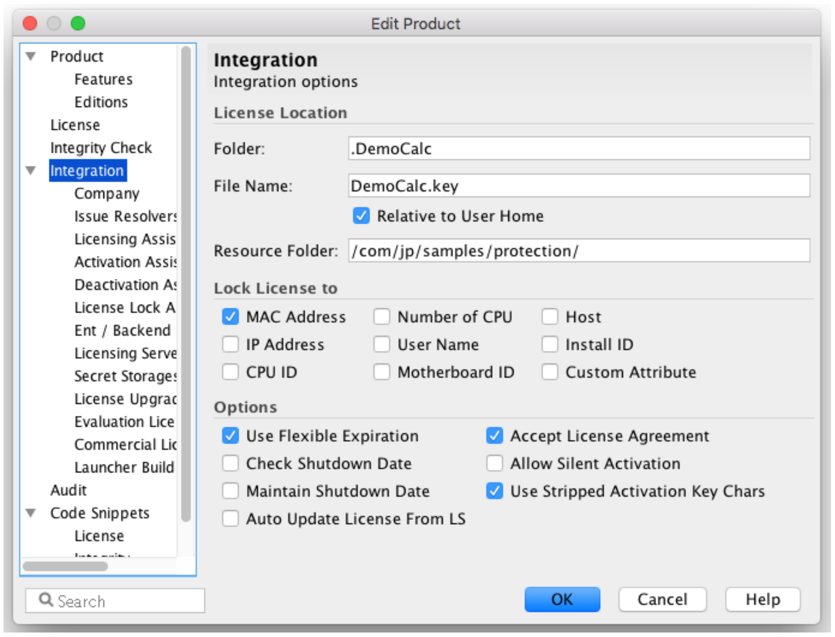
Issue Resolvers
Options to configure issue resolvers

Resolve Missing License by

- [ ] Getting License by Serial Number
- [ ] Requesting Evaliation License
- [ ] Getting License from Licensing Server
- [x] Using Licensing Assistant

3.10 Updated the "Edit Product | Integration" Page *v5.2.1*

There is a new *"Auto Update License from LS"* option that controls if automatic update of licenses from Licensing Server is allowed.



Integration
Integration options

License Location

Folder: .DemoCalc
File Name: DemoCalc.key
- [x] Relative to User Home

Resource Folder: /com/jp/samples/protection/

Lock License to

- [x] MAC Address
- [ ] Number of CPU
- [ ] Host
- [ ] IP Address
- [ ] User Name
- [ ] Install ID
- [ ] CPU ID
- [ ] Motherboard ID
- [ ] Custom Attribute

Options

- [x] Use Flexible Expiration
- [x] Accept License Agreement
- [ ] Check Shutdown Date
- [ ] Allow Silent Activation
- [ ] Maintain Shutdown Date
- [x] Use Stripped Activation Key Chars
- [ ] Auto Update License From LS

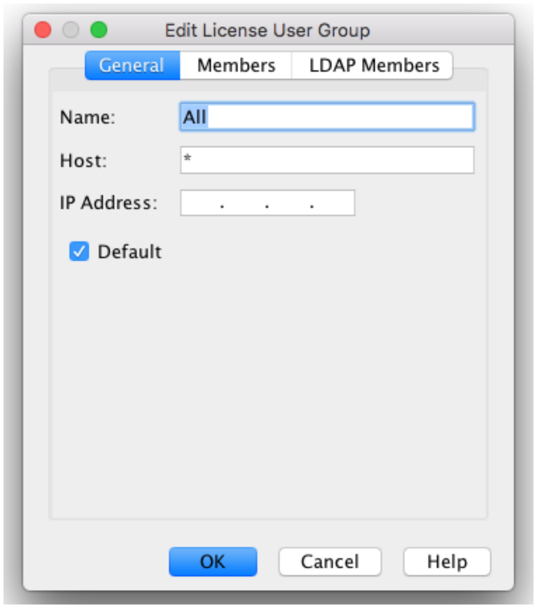
Chapter 4

What's New in Licensing Server

4.1 Server and Management Console

4.1.1 Support of Default License User Group

There is an ability to mark a License User Group to be a default one. If a default License User Group is specified – all newly imported licenses will be automatically allocated to that group.



Name: All

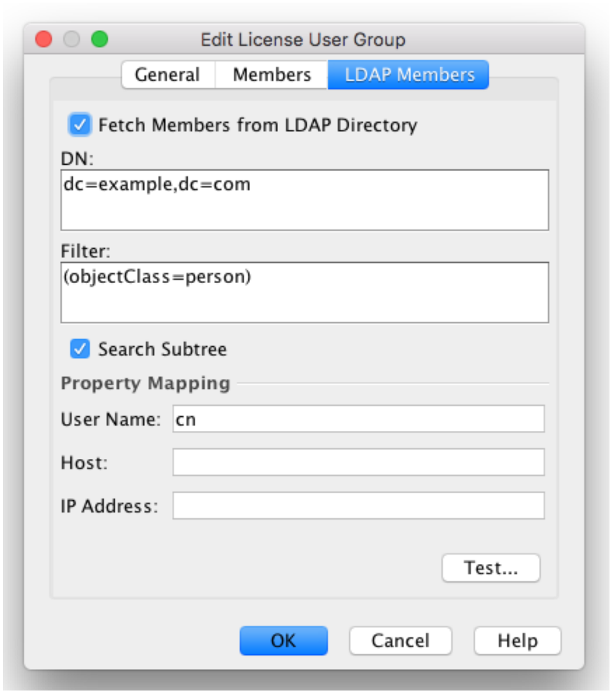
Host: *

IP Address: . . .

✓ Default

4.1.2 Control of License User Group | LDAP Members Traversal

By default LDAP members of a License User Group is searched by traversing down subtree. Now it is possible to control this behavior via the *"Search Subtree"* option in the *"Edit License User Group | LDAP Members"* dialog.



☑ Fetch Members from LDAP Directory

DN: dc=example,dc=com

Filter: (objectClass=person)

☑ Search Subtree

Property Mapping

User Name: cn

Host:

IP Address:

If the *"Search Subtree"* is OFF – members will be searched within one level only. It is recommended to keep this option OFF if possible:

- It can improve performance by avoiding going down subtree if not needed
- It can avoid bugs (NPE's) in system Java classes if LDAP directory is very complex and subtree contains some objects inaccessible for authenticated user.

4.1.3 Improved LDAP Access Testing

There is improved LDAP access testing functionality. So testing of LDAP connection and fetching of sample LDAP users are now performed by the Licensing Server while results are presented in the Management Console. This makes a difference, for example, when:

- The Licensing Server is specially configured to allow secure LDAP connections
- LDAP server is only accessible from the same computer running the Licensing Server.

4.1.4 Improved Security for Server Users *v5.3.4*

There is improved security for handling of Server Users:

1. Passwords for Server Users expire after certain period of days (90 days by default). To change the value for that period - alter the `server.xml` file and change attributes of the `authenticator` element:
   - `passwordValidityPeriod` – password validity period in days (90 days by default). To allow non-expiring passwords – state 0 for this attribute or remove it completely.

2. Empty passwords are not allowed anymore.
3. There are strict rules for new passwords composing so a password must contain:
    - At least one upper case letter
    - At least one lower case letter
    - At least one digit
    - At least one special character: **# ? ! @ $ % ^ & *-**
    - At least 8 characters
4. Passwords are not transferred between client and server and not stored on the server anymore; digests used instead.
5. Unencrypted passwords, written manually to configuration files, will be automatically encrypted and stored during startup of Licensing Server (*since v5.5.0*).

4.2 Server

4.2.1 License Inbox

There is a new *License Inbox* functionality that allows easily importing and allocation of licenses with no need to launch Management Console. To use *License Inbox* – just copy licenses to `<LS_INSTALL_FOLDER>/inbox` folder. Copied licenses will be promptly imported and allocated to the default *License User Group* (if defined). To indicate the status of the import process – licenses will be renamed after the import attempts. So successfully imported licenses will get the `.imported` sub-extension, failed licenses will get the `.failed` sub-extension.

If you need to allocate imported licenses to different *License User Groups* or to *License Users* – create sub-folders inside the inbox folder with names matching the names of desired *License User Groups* or *License Users* and copy licenses inside those sub-folders.

By default, *License Inbox* folder is `<LS_INSTALL_FOLDER>/inbox`. If you want to use a different folder – alter the `server.xml` file and specify a new path as the `inboxPath` attribute of the `<module id="licenseStorage"/>` element and restart the server.

By default, files having the `key` and `license` extensions are considered to be licenses so Licensing Server will try to import only such files. If you issue licenses having different extensions – alter the `server.xml` file and specify a comma-separated list of license extensions as the `licenseFileExtensions` attribute of the `<module id="licenseStorage"/>` element and restart the server.

4.2.2 RMI over SSL Licensing Service Interface

There is a new implementation of Licensing Server interface that utilizes RMI over SSL protocol to support secure communication between customer apps and the server.

RMI over SSL is turned OFF by default to maintain compatibility with existing apps. To turn it ON:

1. Alter the `server.xml` file and change the `secure` attribute of the `rmiInterfaceModule` element to have the "`true`" value.
2. Enable "RMI over SSL" for the customer apps:
   a. Edit your product in Developer Control Center.
   b. Open the *"Integration | Licensing Server"* page
   c. Change the *Connection* property to be *"RMI over SSL"*
   d. Rebuild launcher.

4.2.3 HTTPS Licensing Service interface

There is a new implementation of Licensing Server interface that utilizes HTTPS protocol to support secure communication between customer apps and the server.

To enable using HTTPS for the customer apps:

1. Edit your product in Developer Control Center.
2. Open the *"Integration | Licensing Server"* page
3. Change the *Connection* property to be *HTTPS*
4. Rebuild launcher.

It is recommended to use HTTPS in most of cases because:

1. It can be more efficient than RMI when working with big number of connections
2. It is easier to configure when the server is running behind firewall; there is just one port to open and that port can be already opened if the standard HTTPS port is utilized.

Note

By default, the server utilizes port 443 for HTTPS communication. You can change that port to any other value by altering the `server.xml` file and changing the `securePort` attribute of the `httpInterfaceModule` element to a desired value

Since v 5.4.0 HTTP licensing interface utilizes Vert.x.

4.2.4 Extended Main/Supplemental Licensing Model

There is extended Main/Supplemental licensing model to allow extending of Usage, CPU Cores and Memory limits on the fly.

4.2.5 Extended Named LS Model

There is updated Extended Named LS user licensing model that requires licenses to be locked to particular users for certain min amount of time (min 1 day). During that lock period the license cannot be revoked even by the administrator. Having such a requirement disallows misuse of Named licenses when they can be used like Floating licenses by letting the administrator to revoke licenses as soon as they are needed by other users.

The lock period is defined as the license `lockPeriod` property; it can be specified in the Control Center for a license by changing the *"User Licensing | Lock License for (Period)"* property.

4.2.6 Auto-creation of License Entries

Licensing Server now automatically creates license entries for not-hosted licenses, making it possible to use the "License Storage" Screen to view details and usage statistics for such licenses.

4.2.7 Ability to Specify the System Properties

There is an ability to specify the System Properties, which may be useful to fine tune some functionalities of the server. It can be done by altering the `server.xml` file, specifying `<systemProperty/>` sub-elements of the `<systemProperties/>` element, and restarting the server if it is running:

```
<server>
       <systemProperties>
              <systemProperty name="a" value="value1"/>
              <systemProperty name="b" value="value2"/>
       </systemProperties>
…
<server/>
```

4.2.8 Improvements in Licenses Distribution

1. Added basic validation for all the licenses imported to Licensing Server
2. Invalid licenses will not be distributed anymore
3. Only the newest licenses will be distributed if there are several licenses allocated to particular user or user group.

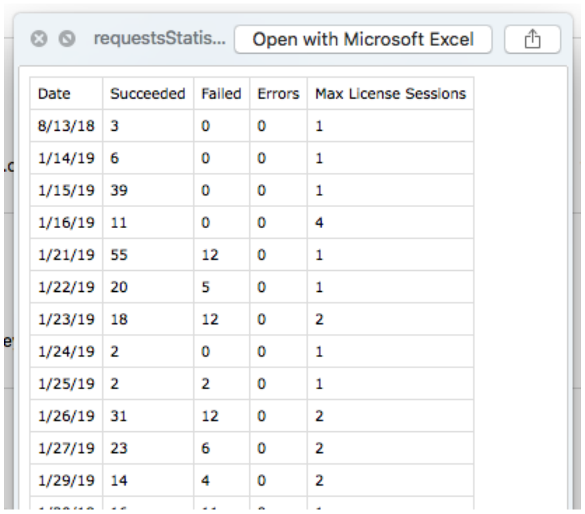
All those changes working in cooperation with the *License Inbox* functionality would allow zero administration of Licensing Server. So for example, it would be possible to import and allocate newer licenses upfront and they will be automatically distributed as soon as they became valid.

4.2.9 Improvements in LDAP Members Fetching

Procedure for LDAP members fetching was rewritten to include more detailed logging, stability and performance improvements.

4.2.10 New Requests Report

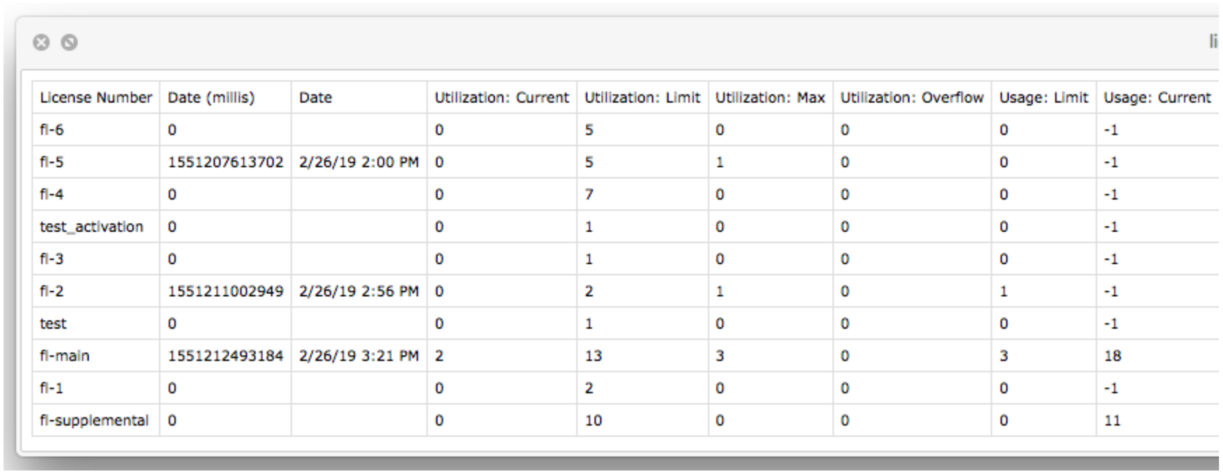
There is a new "Requests Report" that shows number of succeeded, failed requests, errors and max. sessions number per each day.



Date	Succeeded	Failed	Errors	Max License Sessions
8/13/18	3	0	0	1
1/14/19	6	0	0	1
1/15/19	39	0	0	1
1/16/19	11	0	0	4
1/21/19	55	12	0	1
1/22/19	20	5	0	1
1/23/19	18	12	0	2
1/24/19	2	0	0	1
1/25/19	2	2	0	1
1/26/19	31	12	0	2
1/27/19	23	6	0	2
1/29/19	14	4	0	2

4.2.11 New License Report

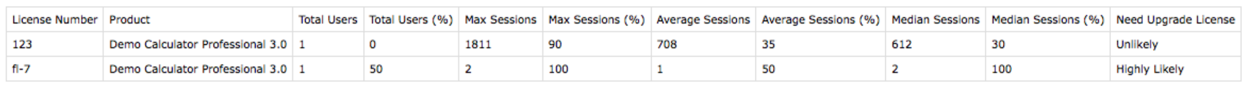
There is a new "License Report" that shows overall utilization of licenses as well as maximum and overflow values for imposed limits.



License Number	Date (millis)	Date	Utilization: Current	Utilization: Limit	Utilization: Max	Utilization: Overflow	Usage: Limit	Usage: Current
fl-6	0		0	5	0	0	0	-1
fl-5	1551207613702	2/26/19 2:00 PM	0	5	1	0	0	-1
fl-4	0		0	7	0	0	0	-1
test_activation	0		0	1	0	0	0	-1
fl-3	0		0	1	0	0	0	-1
fl-2	1551211002949	2/26/19 2:56 PM	0	2	1	0	1	-1
test	0		0	1	0	0	0	-1
fl-main	1551212493184	2/26/19 3:21 PM	2	13	3	0	3	18
fl-1	0		0	2	0	0	0	-1
fl-supplemental	0		0	10	0	0	0	11

4.2.12 New Usage per License Report *v5.2.1*

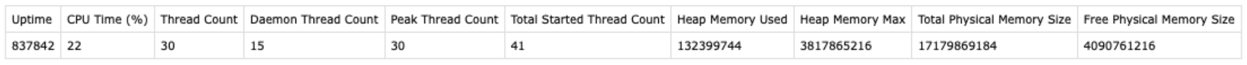
There is a new "Usage per License" report that shows total users, maximum, average and median number of sessions for a particular license. If report generated for the "overall" scope – it includes an additional "Need Upgrade License" column that shows how likely a particular license need to be upgraded.



License Number	Product	Total Users	Total Users (%)	Max Sessions	Max Sessions (%)	Average Sessions	Average Sessions (%)	Median Sessions	Median Sessions (%)	Need Upgrade License
123	Demo Calculator Professional 3.0	1	0	1811	90	708	35	612	30	Unlikely
fl-7	Demo Calculator Professional 3.0	1	50	2	100	1	50	2	100	Highly Likely

4.2.13 New Server Summary Report *v5.3.0*

There is a new “Server Summary” report that shows current server summary information - uptime, CPU load, threads and memory data.



Uptime	CPU Time (%)	Thread Count	Daemon Thread Count	Peak Thread Count	Total Started Thread Count	Heap Memory Used	Heap Memory Max	Total Physical Memory Size	Free Physical Memory Size
837842	22	30	15	30	41	132399744	3817865216	17179869184	4090761216

4.2.14 Ability to Supply Valid SSL Certificates *v5.4.0*

Licensing Server uses a self-signed certificate to allow it running out of the box with no additional configuration. If there is a need to let it use a valid SSL certificate - the following should be done:

1. Obtain a valid SSL certificate for a host the Licensing Server will be up and running.
2. Stop the server if it is running.
3. Use Java’s `keytool` utility to regenerate the `/bin/.keystore` key store so it will contain a valid key pair. Note: the key store is password protected. Please contact a vendor of protected app or directly jProductivity LLC for the password.
4. Restart the server.

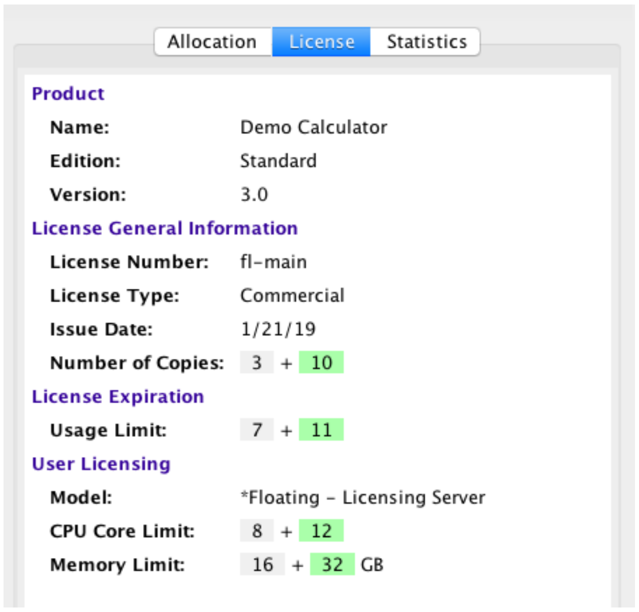
4.3 Management Console

4.3.1 Management Communication over SSL

There is a new implementation of management interface that utilizes RMI over SSL protocol to support secure communication between Management Console and Licensing Server.

4.3.2 Enhanced License Details Views

The “License Storage” Screen now provides enhanced license details views to show additional and supplemental values.



Product

Name:	Demo Calculator
Edition:	Standard
Version:	3.0

License General Information

License Number:	fl-main
License Type:	Commercial
Issue Date:	1/21/19
Number of Copies:	3 + 10

License Expiration

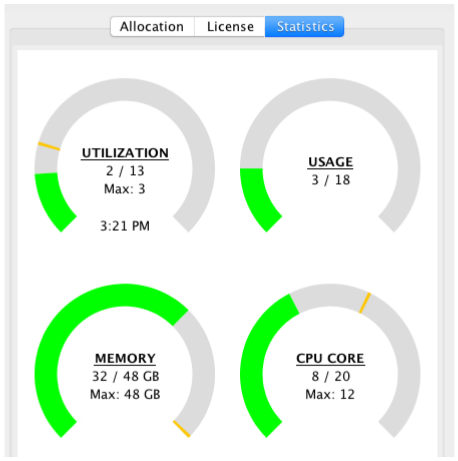
Usage Limit:	7 + 11

User Licensing

Model:	*Floating – Licensing Server
CPU Core Limit:	8 + 12
Memory Limit:	16 + 32 GB

4.3.3 License Statistics Dashboard

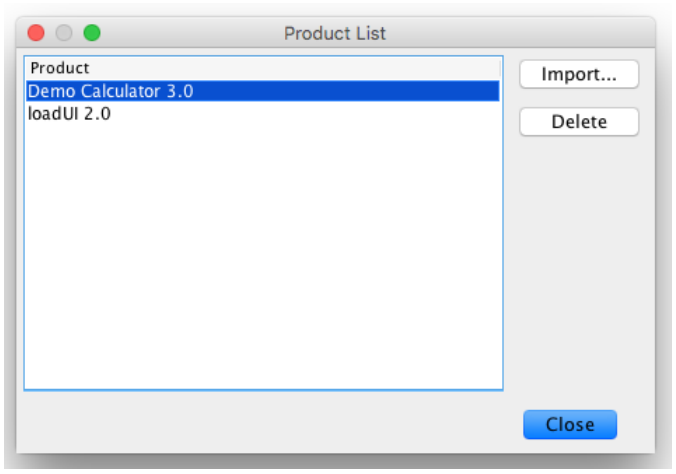
The "License Storage" Screen provides new "License Statistics Dashboard" that shows live license utilization using a set of gauges.



There is also the *"Upgrade Need"* gauge that shows the need to upgrade for particular licenses *(since v5.5.0)*. Note: the *"Upgrade Need"* gauge will be shown only if licenses need for upgrade was calculated (see 4.3.8 Checking for Licenses Need to be Upgraded v5.2.1 for more information).

4.3.4 Ability to Manage Products

Now it is possible to view all supported products and maintain them using the "Product List" dialog *("License Storage | Edit | Product List"*).



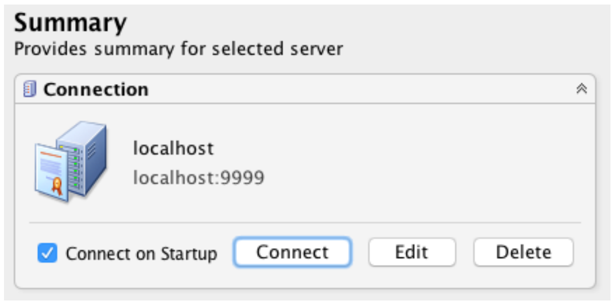
Product
Demo Calculator 3.0
loadUI 2.0

Note: If a product is in use (there are licenses for it) – it cannot be deleted.

4.3.5 Improved Handling of Connections

There is improved handing of Licensing Server connections:

- Ability to automatically select the last used connection
- Optional ability to automatically connect on startup.



4.3.6 Ability to View System Information

There is an ability to view the system information via the *"About | System Information"* menu item. That information can be useful for troubleshooting of both client and server. The following information can be showed:

- System Properties
- System Properties obtained from the Licensing Server (available for *admin* users only)
- Path and contents of the `error.log`

4.3.7 Using Management Console via Command Line *v5.1*

The Licensing Server Management Console provides limited ability to manage Licensing Servers via command line interface. Usage as follows:

```
ManagementConsole [options]
```

`-connection <connection>`	Optional connection to work with. It can be either name of an existing connection or a custom connection specified as 'user:password@host:port'. If not specified - the default connection will be utilized. To configure connections – launch Management Console in the GUI mode.
`-actions <actions>`	Actions to perform in JSON format. `<actions>` can be either: • Path to a file, which contains actions' data. • String with action's data formatted as one line. Note all the quotes must be escaped with a \ character. For example: `[{\"action\":\"enableLicensingActivities"\"}].`
`-help`	print help message

Sample calls would be:

```
        ManagementConsole -connection "localhost" -actions "actions.json"
```

or

```
        java -jar ManagementConsole.jar -actions
            "[{\"action\":\"enableLicensingActivities\"}]"
```

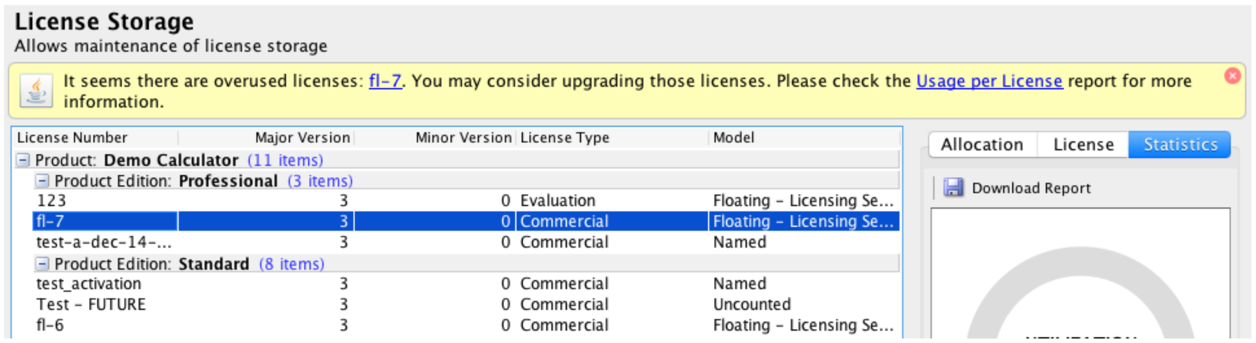
Please check the *"4. Using Management Console via Command-line"* topic of *"Protection! Licensing Server Administrator Guide"* for more detailed description of the command-line functionality.

4.3.7.1 Revoking License Sessions *v5.3.4*

There is a new `revokeLicenseSession` action that allows revoking a license session identified by the *License Number* for a user identified by the *User Name*, *Host Name* and *IP Address* attributes.

4.3.8 Checking for Licenses Need to be Upgraded *v5.2.1*

There is a new activity that runs in background and periodically checks if there are some overused licenses that might need to be upgraded. If such licenses are found – a banner highlighting them will be shown at the top of each screen.



License Storage
Allows maintenance of license storage

It seems there are overused licenses: fl-7. You may consider upgrading those licenses. Please check the Usage per License report for more information.

License Number	Major Version	Minor Version	License Type	Model
Product: **Demo Calculator** (11 items)				
Product Edition: **Professional** (3 items)				
123	3	0	Evaluation	Floating - Licensing Se...
fl-7	3	0	Commercial	Floating - Licensing Se...
test-a-dec-14-...	3	0	Commercial	Named
Product Edition: **Standard** (8 items)				
test_activation	3	0	Commercial	Named
Test - FUTURE	3	0	Commercial	Uncounted
fl-6	3	0	Commercial	Floating - Licensing Se...

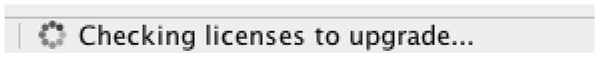
This check can be time and resources consuming so it will run occasionally:

1. On each launch of Management Console.
2. Automatically every 30 minutes. Since *v5.5.0* this ability can be controlled via the *"Options | Check Licenses Upgrade"*.
3. Immediately after adding, updating or removing licenses if the *"Options | Check Licenses Upgrade"* is ON.

Also it is possible to start checking on demand at any time via the *"File | Check Licenses Upgrade"* menu item.

4.3.9 Progress Indicator in the Status Bar *v5.2.1*

There is a new progress indicator can be shown in the Status Bar showcasing descriptions and progress of lengthy processes running in background. If there are several concurrent processes – tooltip will show progresses for all of them.



Checking licenses to upgrade...

4.3.10 Checking for Incompatible Versions *v5.5.0*

There is new activity that checks if Management Console is not compatible with a particular version of Licensing Server it is connected to. This check is done right after connections succeed and there are three possible outcomes:

1. Versions are fully compatible.
2. Versions are fully incompatible. In that case the error message will be shown and Management Console will be disconnected.
3. Versions are partially incompatible. In that case the warming will be shown, some of the features maybe missing and/or less functional.