jProductivity LLC

Protect your investments with Protection!

Protection!™

Licensing Framework for Java

Frequently Asked Questions

Contents

Frequently Asked Questions

General Questions

What is Protection!?

Protection! Licensing Framework for Java is an intuitive, feature rich and platform independent tool for developers interested in protecting their custom applications and implementing a powerful licensing solution. The new version includes:

- A variety of Licensing Models: Named User License, Floating User License and Grace Period Support.
- License Upgrade, Activation and Deactivation support.
- Ability to specify exactly which product features are enabled for a given license type and license state combination.
- Ability to specify a resolver that can be used to intercept and fix any issue that occurs during license reading and/or validation.
- Powerful Licensing Assistant and License Activation Wizards.
- Powerful Protection! Control Center: designed to facilitate product maintenance with the ability to create an unlimited number of products and feature sets for each of the products.
- Web Services Support: provides a ready-to-deploy Web Services application with built-in support for remote license generation and activation.
- Back-End Development Support: includes rich API for building powerful back-ends. Back-End Support offers a default implementation as a foundation for effortlessly building custom back-ends that are exported through Web Services, RMI or other remote invocation technology.
- Plug-ins Support: offers the ability to easily extend default built-in implementations to allow for additional functionalities such as license generation and activation process tracking.

Protection! delivers a powerful licensing solution while providing low TCO for any level of development company, from “lone wolf” developers to an enterprise corporation.
http://www.jproductivity.com/about/press/2004/11-09.htm

Protection! Licensing Framework for Java solves complex licensing challenges. Developers using Protection! are able to build and distribute their applications with the peace of mind that unauthorized use of their applications is minimized, avoiding potentially lost revenues.

Protection! Licensing Framework home:
http://www.jproductivity.com/products/protection/protection.htm

You can use the following URL to purchase Protection! licenses.
http://www.jproductivity.com/products/protection/buy.htm

Please check the following links that would help you with your understanding of Protection!:

User Guide:
http://www.jproductivity.com/products/protection/doc/pdf/Protection_User_Guide.pdf

Developers Guide:
http://www.jproductivity.com/products/protection/doc/pdf/Protection_Developers_Guide.pdf

Feature Matrix:

http://www.jproductivity.com/products/protection/doc/pdf/Protection_Feature_Matrix.pdf

Protection! API:
http://www.jproductivity.com/products/protection/doc/html/index.html

To see Protection! in action please visit:
http://www.jproductivity.com/products/protection/demos.htm

Animated presentations will demonstrate some of the major functionality offered by Protection! Licensing Framework, from demonstration of basic concepts of Protection! Control Center and its usage to illustration of the user experience when running an Application protected with the Protection! Licensing Framework for Java.

Contact sales@jproductivity.com for commercial inquiries or protection@jproductivity.com for technical questions.

What are the software and hardware requirements for Protection!?
256 MB RAM minimum (512 MB RAM recommended)
14 to 20 MB hard disk space minimum (depending on edition, plus additional space for development files and temporary space during installation)

Windows®
- Intel® Pentium® III/500 MHz or higher (or compatible)
- Microsoft® Windows® 2000 (SP4), Windows XP, or Windows 2003

Solaris™
- UltraSPARC™ II or higher
- Solaris™ 9 (2.9)

Linux®
- Intel Pentium III/500 MHz or higher (or compatible)
- Sun™ Java™ Desktop System Release 2 or Red Hat® Enterprise Linux® 3.0

Mac OS® X (English Only)
- 800 MHz G4 or higher, 256 MB RAM minimum, 512 MB recommended
- Mac OS® X 10.3

How can I contact your customer service representative? *E-mail* To send us comments, questions or feedback concerning our products and services; please use the appropriate e-mail address below:

Products

Productivity!	productivity@jproductivity.com
Protection!	protection@jproductivity.com
Components!	components@jproductivity.com

General Addresses

General Information	info@jproductivity.com
Sales	sales@jproductivity.com
Support	support@jproductivity.com
Career	career@jproductivity.com
Press	press@jproductivity.com
Webmaster	webmaster@jproductivity.com

Mailing Address

jProductivity, L.L.C.
275 Madison Avenue
New York, NY 10016

I have some suggestions and feedback. How can I submit these?
As part of our continuing efforts to improve our product, we welcome your comments, suggestions, and general feedback on the project. If you have questions about Protection!, please feel free to contact us at mailto:protection@jproductivity.com.
If you discover any issues or defects in Protection!, please send a description to mailto:support@jproductivity.com. We would appreciate it if you could provide us with additional information. This will assist us in finding a quick resolution of the problem. Please let us know your:
1. Exceptions stack trace if any.
2. JVM vendor and version.
3. Operating system version.
4. Running threads dump.

How many versions of Protection! are available?
By offering Standard and Professional editions, Protection! delivers a powerful licensing solution, while providing low total cost of ownership (TCO) for any level of development company, from "lone wolf" developers to an enterprise corporations. Protection! Developer and Protection! Sales are targeted to developers and sales stuff respectively.

What is the support policy of Protection!?
jProductivity offers a range of technical support services, from free self-service and pre-sales support to 15-day post-sales installation and configuration support, per-incident support, and premium support.

Premium Support
Get Premium Support and an Upgrades subscription for Protection! Licensing Framework, and add even more power to your application protection and licensing solutions. With Premium Support you will receive a full year of phone and email support. Your subscription also includes free product upgrades, help with custom code extensions, along with fast answers and how-to assistance.

Per-Incident Support
If you are a current customer, but do not have a Premium Support membership, you can get technical assistance on a per-incident basis for US $135.00 per incident. This fee includes telephone and email support for a single issue until the reported problem is resolved. An incident is defined as a single support issue with Protection! Licensing Framework, and the reasonable effort needed to resolve it. A single support issue is a problem that cannot be broken down into subordinate parts.

Free Evaluation (Pre Sales) Support
jProductivity offers complimentary email pre-sales support while you are evaluating Protection! Licensing Framework.

Self-Service Support
jProductivity offers several English-only on-line self-service technical support options, available 24 hours a day. These include:

User Guide (PDF):
http://www.jproductivity.com/products/protection/doc/pdf/Protection_User_Guide.pdf

Developers Guide (PDF):

http://www.jproductivity.com/products/protection/doc/pdf/Protection_Developers_Guide.pdf

Protection! API (HTML):
http://www.jproductivity.com/products/protection/doc/html/index.html

Frequently Asked Questions
http://www.jproductivity.com/products/protection/doc/pdf/Protection_FAQ.pdf

For more information on Support options please see the following link: http://www.jproductivity.com/products/protection/support.htm

What is Premium support?
Get Premium Support and an Upgrades subscription for Protection! Licensing Framework and add even more power to your application protection and licensing solutions. With Premium Support you will receive a full year of phone and email support. Your subscription also includes free product upgrades, help with custom code extensions, along with fast answers and how-to assistance.
Fro more information on Support options please see the following link: http://www.jproductivity.com/products/protection/support.htm

What is the price of Premium support?
- Protection! Std Premium Support: $407 per developer
- Protection! Pro Premium Support: $1,487 per developer

Please note: Premium Support Includes 1 (one) year of technical support and product upgrades.

For more information on Support options please see the following URL: http://www.jproductivity.com/products/protection/support.htm

What are the benefits of Protection! to its users?
Protection! Licensing Framework for Java solves complex licensing challenges. Developers using Protection! are able to build and distribute their applications with the piece of mind that the unauthorized use of their applications is minimized, allowing them to recover potentially lost revenues.

How Protection! is licensed?
Protection! Licensing Framework for Java is royalty free and priced per developer. "Royalty free" means that you can deploy as many applications (built with Protection! Licensing Framework) to as many of your customers as you want. There's no hidden extra cost. "Priced per developer" means that all developers working with the Protection! Licensing Framework at any given time and/or any developer that uses Protection! Public API (com.jp.protection.pub.*) needs to have one license of Protection! each.
The Protection! Backend Deployment license is priced per Server/CPU. This means that any application that uses the Protection! Web Services application and/or the Protection! Private API (com.jp.protection.priv.*) to generate or activate licenses requires a Protection! Backend Deployment License.

"all developers working with a Protection! Licensing Framework" means every developer who is using:
1. the Protection! Control Center GUI and its Tools.
2. the Protection! API or the Runtime - "Directly or Indirectly" to utilize an object which directly or indirectly accesses Protection! Run-Time during initial or post-deployment development in order to perform a function during the process of developing new logic.
3. Working with project(s) that has Protection! library in its CLASSPATH

If any of the above is true for a developer, then that developer requires a separate individual developer's license.

Example: Two developers sharing a multi-user system would require two licenses; two developers each working on their individual system plus a separate nightly build system would require three licenses.

Note, this is not a complete definition. Please refer to Protection! License (<*install_dir*>*license.htm*) section: "2.3 Named User Grant of License" for the full definition.

Please note: Protection! Licensing Framework has:
- No Runtime Fees
- No CPU or Server Fees
- No Deployment Fees

How to buy

Please see the following ling for more details and purchasing options available to you: http://www.jproductivity.com/products/protection/buy.htm

Do you provide consulting services?
jProductivity will be happy to assist companies with any protection schema design and implementation. jProductivity are also offering consulting services for companies that would need any help and assistance with implementation of Protection! Licensing Framework into their products.

For inquiries please contact sales@jproductivity.com

Using Protection!

What database is used as a backend with Protection!?
There is no default database that comes with Protection! Backend, and no dependencies to any other database. Protection! backend does not represent any kind of CRM/SFA/ERP application. If such functionality is required, the developer is responsible to link Protection! Backend to an existing/new database and/or existing CRM/SFA/ERP applications via Protection! Backend API.

Will the Private Key Bytes ever change once the product has been created?
No. Once the product has been created in Protection! Control Center the Key Bytes will not change for a given product under any circumstances.

Is it possible to generate a commercial license with an expiration period?
You can generate a commercial license for you customers with the specified expiration period. At a time of license reading and validation, you would check for an expiration period and would (or would not) check for the application version.
This allows some software publishers to implement "rental" licensing mechanism. In this case, your client is renting your application/library for a mutually agreed renting/leasing period (this would indicate license expiration period span).

Why is the Commercial radio button on the License Screen disabled?

Generation of the Commercial license is disabled in the Evaluation version of Protection!. You would need the full licensed version in order to generate Commercial licenses for your application.
Because after you generate a license, it has life of its own - we are not able to provide commercial license functionality in the evaluation version of Protection!

Why does the Option to "Provide User Information" have no effect?
These settings would only have effect when a license is set with "Activation" and during activation of the license the user chooses on-line activation. If the user chooses off-line activation, then he/she theoretically would provide his/her user information via the phone/e-mail to the developer/publisher/sales staff.

Does Protection! prevents the end-user from decompiling the Java byte code?
It is not the responsibility of Protection! to prevent de-compilation. However, Protection! does prevent potential malicious application patching via Protection! CRC support module.

Do you consider Protection! to be un-crackable?
One of the major things that differentiates Protection! from a small number of competitive products is that Protection! is a framework. Therefore, you, as a developer/publisher not only have *full* control of design and implementation of your licensing solution, but it is also totally up to you on how well and/or strongly you will protect your application. Please keep in mind that there is always a fine-gray-line between strong protection and user-friendliness of your application. On a different note - I do not think anyone can honestly say (and if anyone would - it would be a lie) that something is un-crackable. However, this is a more theoretical and philosophical question for debate :)

I need to give our resellers ability to manage licenses for our products
Q. *Or reseller will be managing the licenses for our application and will distribute all those licenses to our customers. How can this be implemented? For example: We will provide the reseller with the ability to use 400 featureA and 200 featureB licenses of our product, and the reseller will be dealing with the management and distribution of those licenses.*
A. You can handle this several ways:

a) Protection! Control Center allows you to generate Serial Numbers in bulk for a given license via the Edit | Generate Serial Numbers... menu. You would generate 400 Serial Numbers for featureA and 200 Serial Numbers for featureB. You can then send all 600 Serial Numbers to your reseller, and he/she would distribute them to your customers as needed. When customers launch your application, they would have the ability to get a license by providing a valid Serial Number. In this case, your Protection! Web Service back-end would generate a valid license for a given Serial Number and distribute it (via Direct Internet Connection or e-mail) to the customer.
b) You can also generate 600 (400/200) actual license key files and supply them to your reseller. Then, the reseller would be responsible for providing each license key file to your customers.
c) Depending on your business and logistical relationship with your reseller, you can also purchase an additional Protection! Backend Deployment License for your reseller, sell it or give it to him as you see it fit, and deploy Protection! Web Service on the reseller's site. This would give your reseller the ability to generate Serial Numbers and Licenses without limitations. Certainly some form of accountability and control needs to be implemented in this scenario, so you can know at least how many and what kind of licenses your reseller generated and to whom these licenses were distributed.

What is the Custom License Agreement option?
Please note that this section is reserved for *TRULY* custom License agreements. In normal usage, you would specify the License agreement in your code through the following methods (this is also generated by Protection! Code snippets):

```
licenseHost.setLicenseTextCommercial("Commercial license agreement text");
licenseHost.setLicenseTextEvaluation("Evaluation license agreement text");
```

You should use custom license agreement in situations when you need to create a special agreement for one customer in some special case only!

Please note: Custom License agreement is encoded into the license key. Therefore if your custom license agreement is large - your generated license key would also be increased in its size.

Why am I getting an expired license message when trying to run the Demo application?
Possibly the license expired at some point in the past. If this is the case, then the expiration flag was written into the secret storage - this prevents your users from downloading evaluation licenses and using them after the expiration date. Either generate an extended evaluation license, or delete your secret storage and run your application again with a regular evaluation license.

Development Questions

General

How do I implement Protection! with applications running on a GUI-less, server environment?
It is quite easy to embed Protection! into any application. The developer can use the following steps:
1. Create a new product in Control Center using the Edit Product Dialog, and specify product's editions, features and other attributes.
2. Specify the resource folder attribute for your product using the Edit Product Dialog | Locations | Resource Folder edit box to let Protection! know where to find the license in your application archive (e.g. /com/acme/app)
3. Use "Headless" code snippet to get a template of the class that should be used to embed Protection! in your server application.
4. Modify the generated code to execute actions appropriate for certain events. Here is a simple example:

```
private LicenseAdapter licenseListener = new LicenseAdapter()
{
  public void licenseOk(LicenseHost aSource, License aLicense);
  {
    enableMyFunctionality(true);
  }
  public void licenseExpired(LicenseHost aSource, License aLicense)
  {
    enableMyFunctionality(false);
  }
};
```

In this example, you have some functionality disabled by default and you enable it only when the license is valid and is not expired.

5. Add a call of checkLicense() method somewhere in your code during application startup, and schedule a call of it each new day (hour, minute...) to ensure that the license is still OK.
6. Generate a new evaluation license, make an application archive (e.g. .jar, .war, etc...) and include application classes and license file into it.

That's it! Now you can deploy your application on your Server, and it will work until the date stated in the bundled license. When your customer purchases a commercial license he/she should simply copy it to the HOME folder to get the application licensed!

Is activation with lock limited to the platforms that have native libraries provided?

If you use the default implementation provided by Protection! - then Yes. However, you are free to provide your own implementation. Protection! Professional edition does provide developers with the ability to lock a license to a specific system. The default implementation allows you to lock the license to a network card MAC address. However, the developer/publisher is free to lock the license to any other user-definable attribute.
For example: You can lock the license to the user's host address by making a subclass of LicenseHostPro and overriding the getActivationLockKey() method to employ any other methods/lock mechanisms/etc...
The following code snippet illustrates how to employ a node's IP address to be used as activation and lock key:

```
public class LicenseHostProEx extends LicenseHostPro
{
  protected String getActivationLockKey(LicenseImpl aLicense)
  {
    return getActivationKey(InetAddress.getLocalHost().getHostAddress().
      hashCode());
  }
}
```

How do floating licenses work?
Protection! sends a UDP broadcast (Firewall friendly) message on a specific port, and with
a specific signature. At the same time, Protection! listens and collects responses. Response will contain the license number and other important license attributes. As responses are collected, Protection! counts how many licenses are allowed and how many licenses are in use. If the number of licenses in use is greater than allowed by the floating user license, then Protection! fires an appropriate event.

You should override the numberCopiesViolation(…) method in your implementation of the LicenseListener interface to listen and properly handle such situations.

Please note: Protection! is a framework, and it will provide you with all necessary mechanisms and sometimes their default implementations. As a developer, however, you need to tell in you implementation code what should happen with your application when Protection! tells you that certain events/conditions have occurred.

What if on-line activation is not possible?

Please see the following simple code snippet that allows generation of the activation key. This snipped can be embedded or used to generate the activation key. The generated activation key can then be emailed back to the original developer/publisher, along with the original license for further generation of the license with the "activation and lock" option.

```
public class ActivationKeyGenerator
{
  // simply prints activation key
  public static void main(String[] args)
  {
    LicenseHostPro licenseHost = new LicenseHostPro();
    System.out.println(licenseHost.getActivationKey());
  }
}
```

Please note: Above code snippet is suitable for "headless" environment. For GUI based applications, the Protection! License Activation Wizard should be used to resolve license activation issues.

How do I implement Protection! with a product that has SDK and Runtime libraries?

Q. *We are deploying our products as:*

1. SDK - The customer develops applications which are linking with our libraries
2. Runtime - The customer purchases a runtime license to run the applications they created with our SDK
3. Applications - The applications that we created with our software base
4. How we going to differentiate Runtime and SDK distributions?

A. The schema could work as follows:

- For your SDK - you could create a product in the Protection! Control Center with 2 (two) editions - API and Runtime. You would then generate a license for the API edition and would implement license reading and validation somewhere in either/or both - initialization method, a specific API call, etc. In your API license, you would set some limitations like expiration date, number of copies, etc. In your Runtime/Deployment license, such limitations would go away.
- For your Applications - if they are using your SDK, then it is probably a good idea to make your application see the SDK license. This way you would not need to generate two or more licenses for an application. However, you can also have two or more licenses if desired - one for the SDK and one for the application.

Can Protection! be integrated with online automated systems?

Protection! would allow you very smooth and seamless integration with almost any backend system by either using Protection! Web Services (default implementation) or by writing your own backend implementation (RMI, EJB, etc). One of the major points that separates Protection! from its competitors is that Protection! is a framework and therefore gives you as a developer/publisher all the necessary tools and methodologies, while not imposing any limitations or having any "underwater stones". You can use and embed Protection! into your application in as little as 5-10 minutes (via automatically generated for your product and ready to use java implementation files) or extend Protection! to fit your specific needs.

There are many security models that could be implemented with the aid of Protection! These models could be either manual (require manual license generation) and/or automated where a license can be generated by the Protection! backend automatically. With automatic license generation, the license can either be deployed to your customer immediately or can be stored in some queue for further review and approval by

sales/management before distribution). You can also tie Protection! backend to your sales-force application that processes your orders, so that the backend will query if a valid purchase was recorded before the commercial license is activated. As you can see, these models could be as complex or as simple as you want them to be. Protection! gives you 100% flexibility to implement any of them by being a Framework, and by not imposing any of the models on you as developer/publisher.

In general, for a GUI application you could follow these steps:

1. Vendor generates a license that requires activation and lock
2. Vendor provides this license to a customer
3. Customer copies the license file to the right place manually, or starts the application and uses the Licensing Wizard to specify the license file location
4. Customer gets License Activation Wizard prompting him/her to activate the license If offline activation is chosen, then he/she should provide the Activation Key shown in the Wizard back to the vendor by e-mail or phone
5. Vendor re-generates the license, turning on activation and lock and places supplied by customer Activation Key in the "Activation Key" box in the Protection! Control Center
6. Vendor provides the activated license to a customer.

If Protection! Backend is used, then steps 5-6 will be handled by Protection! Backend automatically.

How can I implement an efficient "Secret Storage"?

Protection! provides a file-based implementation of secret storage that has no native code dependencies, and therefore will work on any platform. You can use several instances placed in different locations e.g. one based in the user's HOME folder, another one in some unique system folder, etc. Having several secret storage files can be enough in most of the cases. As an alternative, you can implement your secret storage using standard Preferences API to hold data in a more-or-less secret location (e.g. in Registry under Windows).

Actually, the only time you should be seriously concerned about efficient secret storage design/implementation is when you provide a bundled evaluation license, and therefore allow a flexible expiration date (the flexible expiration date is set based on the first usage of the application). In this case, if customer locates and removes all secret storages of your application, he can continue to use the evaluation indefinitely.

If no bundled license is used, and therefore you will provide evaluation licenses personally to each of your customers, the only way to trick your application is by playing with the system time. Certainly it is possible in some rare situations, but it is completely inappropriate for server and enterprise based applications as it can cause a series of other server applications to behave incorrectly.

How does Protection! recognizes the expiration date if the application is already running?

If your protected application is currently running and/or in constant running state (i.e., server based application) you should perform license checking procedures at some predetermined/scheduled time interval.

How does Protection! handle various events fired during the license reading/validation processes?

During license checking and validation, Protection! Licensing Framework provides detailed feedback about the process status and its results back to the application. This feedback is provided by firing an appropriate set of events. It is the responsibility of the application developer to decide which action should be taken in response to a particular Protection! event. Default implementation assumes no actions for any such events

provided by Protection! Licensing Framework. For example: when it is determined by Protection! that the license file is invalid, the application can either exit or the application could disable all or part of its functionality. Such a decision is the responsibility of the application developer (there is no default behavior provided by Protection! Licensing Framework). In addition to the event based analysis/action, it is always possible to get the current status of the license at anytime during an application's run/lifecycle, to see whether the license is valid.
While the default implementation of the license checking mechanism provides functionality that would be enough for the majority of applications, developers can easily override/extend any license reading/validating procedures to get the desired results.

Would Protection! detect a "number of copies violation" if the application is running on two different networks?
Probably not! It depends on the network settings/configuration. If firewall(s) or router(s) are instructed to filter broadcasting, then this approach will not work. While the current default implementation of the "number of copies violation" detection mechanism has some well known disadvantages, it requires zero-administration and eliminates additional costs of buying and hosting a licensing server.

Can Protection! be integrated into an app server environment?
Yes, Protection! can be easily integrated into an application server environment. Protection! Pro also provides:

- Powerful default back-end implementation that gives you a foundation for building custom back-office system integration
- Plug-in support to allow easy extension of default back-end implementation
- Ready-to-deploy Protection! Web Services application

Please look at the "Licensing Facade Configuration Dialog" section in the Protection! User Guide for more information.

Does the secret storage file get deleted during the uninstall of the protected application?
This depends on how you build your installation. If you deploy your secret storage during the installation and you use Protection! file-based default implementation, then probably every installer package on the market today would allow you to exclude certain files from the uninstaller script, so the file would remain on the user's system. This said, you can generate secret storage dynamically after the installation and during the first usage of the application (see sample DemoCalc that comes with Protection! for an example of such an implementation).

Is Protection! secret storage only file or Java Preferences API based?
No. The secret storage is a mechanism which used to provide a way to persistently store various information about the application and any other information (if necessary). The default secret storage implementation is file or Java Preferences API based. However it is up to you as a developer to implement a desired secret storage mechanism of your own, if so desired (e.g. Windows Registry based).

Can someone else who bought your product generate licenses for our products?
Without having product storage there is no such possibility for anyone who also has Protection! to generate licenses for someone else's products. Similarly, you would not be able to generate license for the Protection! Control Center yourself.

Is Protection! compatible with obfuscation products?
Protection! co-exists very happily side-by-side with code obfuscation.

In general, you should embed all Protection! redistributable classes inside your application archive, which in turn should be obfuscated using your favorite Java byte code obfuscator.
Also, please note that if you will use Protection! provided CRC support, then with CRC and obfuscation you should use the following schema:

- Create your classes, build your jar, obfuscate it
- Generate CRC on already obfuscated classes, update CRC checking code and recompile it.
- Rebuild your archive
- Obfuscate your archive *excluding* classes that you used in CRC calculation

Is it possible to define the product editions, features, etc. in a relational database?

In theory, it is possible to define the product editions, features, etc. in the relational database, along with the key used to generate the licenses, and use the API to pass that information in to generate the license.
You can obtain products from the products storage and save their properties to a some database. Later, when you'd like to read or generate licenses you can create a product instance, fill its properties from the database, and use it. Please check classes in the `com.jp.protection.priv` and `com.jp.protection.priv.products` packages to get full access to the required low-level API.
However, the most simple way to do this is by using BLOB's in your database (if supported). In this case, you can simply save and load products storage to and from DB using a BLOB field. `ProductsStorage` class provides the following methods suitable for such a task:

```
public boolean save(OutputStream anOutputStream) throws IOException
public boolean load(InputStream anInputStream) throws IOException
```

Where can I find Javadocs for Protection! API?

You can find Javadocs in *<install_dir>/doc/ProtectionDoc.jar*

Why doesn't Protection! provide any error messages?

Suppressing error messages is by design, as having any output during license reading and validation can effectively reduce efforts required to break the protection. That's why there are no exceptions shown during the process. But you can change this behavior for debugging purposes by calling `setVerbose(true)` methods for license reader and host.

Can Protection! be integrated with the installation tools

You can take the following approach:

1. Use GUI code snippet that is generated by Protection! Control Center.
2. Build it and add it to your Installation tool of choice to be executed as custom code (see your installation tool documentation on how to include/execute custom code).
3. Call it (ProtectionSupport code) from within your Installation tool where you think it is appropriate.
4. Call to ProtectionSupport class would initiate standard Protection! Licensing Wizards and will aid your users in obtaining the license.
5. When your user would launch your application, the license would already be obtained (previous step).

What is the encryption size used for a Protection! license?

Protection! uses 128-bit encryption. This size was chosen for several reasons:

- this is enough security for the license key file
- this allows us to keep the size of license key files realistic and manageable, as increasing the key size will dramatically increase the file size for the license keys

Are you using RSA key pair?
Yes

Do you generate a key pair for each product?
Yes

Does Protection! support a license with a flexible expiration date?
Yes, Protection! will allow you to generate a license key having a flexible expiration date - meaning that the actual expiration date would be set when the user launches your application for the first time. The expiration date would be set for a number of days (specified by you) from the first usage of your application.

Protection! allows you to bundle a license with your application. You should specify the license location options using the License Location group in the Location tab of the Edit Product dialog. You should then specify license file name, folder to find the license in the local file system and resource folder (e.g. */com/jp/samples/protection/*) to find the license within your application archive. By default, Protection! tries to find license in the local file system. If the license is not found, then Protection! tries to find the license in the application archive. This allows for bundling evaluation licenses with your application (e.g. for CD distribution). When a customer buys a commercial license, he/she should place it into the specified folder or use the help of the Protection! Licensing Assistant to do so.

Please use the following method `setAllowFlexibleExpirationDate()`to specifiy whether the expiration date of a license should be adjusted according to the current date and duration of the evaluation period.

How much time would it take to implement Protection! in my application?
Protection! has an extremely fast learning curve. With the help of the Protection! tutorials and provided examples, a developer would be able to write protected application in a matter of hours.

Can a License be created from the remote location?
Yes - this could be done with the aid of Protection! Backend via WebServces (default implementation) and/or EJB, RMI, etc.

Why are activation, serial number, and other things missing from the License Activation Wizard dialog?
You probably did not create and register `LicensingFacadeProvider`. Therefore, Protection! does not know how to get a license from a remote host, etc...

Please look at the DemoCalculator (that comes with Protection!) source code

<install_dir>\samples\src\com\jp\samples\protection\DemoCalcProtectionSupport.java

for `initLicensingFacade()` method.

How can I enhance the license checking procedure?
Please look in your *ProtectionSupport.java* file (generated code snippet):

```
private DefaultLicenseAdapter
  licenseListener = new DefaultLicenseAdapter(owner) {…}
```

has a lot of methods such as `licenseCorrupted(…)`, `licenseExpired(…)` etc. You can override any of these methods to handle your situations. For example if license is expired you can invoke `System.exit()`, etc.

How does the extended evaluation license work?
Extended evaluation ignores any previously set expiration time, and therefore it allows you to extend the evaluation period for eligible customers. If and how many times you and/or your backend will issue extended evaluation license is all up to you as a developer/publisher. A user could request extended evaluation license ad infinitum, but you (if you issue licenses manually) and/or your backend implementation would need to make a decision based on your business practices if you allow extended evaluation to be requested at all/more then once/how many times/ etc...

What is the file size overhead for embedding Protection! libraries into our application?
This is depends on the Protection! Licensing Framework functionality that you are planning to utilize in you application. For a description of the deployment libraries please see *<install_dir>\lib\deploy.html.* In case you are only deploying Protection! Library Redistributables, you would need to deploy Protection.jar only (~640 KB).

Are native calls always used even when I do not check the MAC address?
No. You need to provide native libraries only if you are planning to utilize the "named-user" licensing model with the ability to lock a license to a specific computer system.

Is it possible to provide default user details with the application?
You can do this using the following scenario:
On Publisher Side: put customer details into the license properties.

On Client Side:
1. Read the license.
2. Create an instance of the Customer class and specify its properties based on the properties you put into the license.
3. Write customer data using

```
customer.toPreferences(LicenseUtils.getCustomerPreferences(license.
  getProduct()));
```

Now, customer data will be shown in the Protection! About Dialog.

Why does the license checking process not fully complete sometimes?
It seems the issue is caused by missing native libraries. Such libraries are required to obtain the network card address used to lock a license to particular computer. To see any exceptions, you should turn on verbose mode of `LicenseHost` by calling `setVerbose(true)` method. Please consult the *<install_dir>lib/deploy.htm* for more information.

What part does the FileSecretStorage.setDirty() play?
It allows you to set Secret Storage mode when changes were made since the last load or creation. If secret storage is not modified (not dirty) the `save()` method simply does nothing.

Can multiple Secret Storage files be kept in-sync?
Yes, it is possible by using the `LicenseHost.saveSecretStorageProperty(…)` method.

When should I call the SecretStorage.save() method?

Q. *Should I manually call `SecretStorage.save()` after a call to `SecretStorage.setProperty()` or is it called automatically after a property has changed?*
A. You should manually save Secret Storage by call `SecretStorage.save()` when you think it is appropriate.

Should I implement the code for checking the number of copies myself?

You should override `numberCopiesViolation(…)` method in your `LicenseListener` implementation and write code that will make appropriate actions to handle this case.

Please note: Protection! is a framework - it will provide you with all necessary mechanisms and sometimes their default implementations. However, you as a developer need to tell in your implementation code what should happen with your application when Protection! tells you that certain events/conditions have occurred.

Does Protection! provides ability for a product to be locked to a specific system?
Yes, Protection! Professional edition does provide developers with the ability to lock a license to a specific system. Protection! provides support for a variety of licensing models. The "named-user" licensing model will allow you to lock the license to a network card MAC address (default implementation) or other, user-definable attribute.

How would Protection! deal with re-installation of a protected application and evaluation license?
This is controlled by the Secret Storage file(s). Because the Secret Storage file is being generated in the first use of your application, the uninstaller would never remove this file because the uninstaller does not know anything about this file. It is up to you where and how you implement Secret Storage. The simplest way is to put it in user HOME directory. However, you can also modify the Registry, create multiple secret storage files, etc, etc.

Is it possible to use Serial Numbers without contacting software developer / publisher?
No, this is not possible as Serial Numbers represent a very small subset of the license and designed so you can deploy your application and supply a Serial Number on a CD.

How can I test numberCopiesViolation?
Such test needs to be performed on multiple copies on two or more different systems. If you are testing multiple copies on same system - this would never work because of the same computer.

How do I specify custom properties for a given license?
The easiest way is to create custom properties in Control Center (see Properties Tab - Control Center License Screen). Give this property some descriptive name like "Connection Number" (or whatever you want it to be) and set it to a desired value. In your code, after you read a license, use `getProperty(…)` method to read your property and its value. Then you can specify how your application would behave if the property value is outside of its expected range.

How to include line breaks in the license agreement text?
Q. *I cannot seem to include new lines in the license agreement text. I tried \n and \r\n but only get a continuous string that wraps.*

A. Because the License Agreement is treated as HTML, you should use HTML tags to brake your text (e.g.,
</br>; , etc). Using HTML, you can also implement better formatting of your text vs. plain ASCII text.

Is Protection! multi-platform?
Protection! Licensing Framework is multi-platform. Protection! is tested and verified for Microsoft Windows NT/2000/XP, Solaris/x86, Linux (Red Hat/SuSE) and Mac OS X. However, it should work on any Java-Enabled platforms.

Does Protection! show exceptions, if any, during license reading and validation?
No. To see exceptions, if any, you should turn on verbose mode of `LicenseHost` and/or `LicenseReader` by calling `setVerbose(true)` method.
Please note: This should be set for testing purposes only and should be turned off for production.

How can I specify various environmental settings within the license?
Protection! provides you as a developer/publisher with ability to specify an unlimited number of custom properties (key/value pairs). Such properties could be used at any point of license reading/validation in order to allow/disallow some (or all) of you application functionality.

Why am I getting java.lang.NoClassDefFoundError exception?

The following exception:

java.lang.NoClassDefFoundError:
org/doomdark/uuid/NativeInterfaces ...

is caused by missing libraries needed to support “named-user” licensing model. You need to provide *<install_dir>/lib/jug.jar* and native libraries if you are planning to utilize the "name-user" licensing model with the ability to lock a license to a specific computer system. For description of the deployment libraries please see: *<install_dir>/lib/deploy.html.*

Protection! Back-end and Web Services

How do I manage and track generated licenses?

Protection! is not responsible for such tasks. However, you can achieve this by writing a plug-in to Protection! Web Service default implementation.

Protection! provides default implementation of `LicensingFacade` as a Web Services application. Certainly you are free to make any other implementation suitable to your requirements e.g. RMI or EJB based. Licensing Facade Config Dialog provides the ability to configure Protection! WS application so you don't need to play with WS or servlets directly. You should just specify all the options and deploy them along with the product's storage to the Web application using the File | Deploy WS menu in the Products Screen. As a result, you'll get ready to deploy Web application capable of license generation and activation through the use of Web Services. This allows you to quickly and easily start development of a protected application.

While the default implementation provides a good start, you probably need to extend it to have the ability to log issued licenses and/or decide when it is possible to issue or activate a particular license. It is possible by writing a plug-in to the default implementation. Plug-ins should be implementations of either:

`LicensingFacadeDelegate` or `LicensingFacadeExtension` interfaces. You should also provide implementation of `LicensingFacadePluginFactory` responsible of plug-in creation. Then you should pack all your classes to the general use archive and use the Plug-in page of the Licensing Facade Config Dialog to specify factory class, plug-in archive and all other libraries required by plug-in to work (e.g. JDBC driver). Finally, you should put this config to the WS application and deploy to the Web container. It is also possible to specify database options for convenience of database plug-ins development.

Does Protection! Web Service perform any kind of logging?
Yes, Protection! Web Service provides logging using default Log4J based implementation.

Why do I get an exception when trying to connect to Protection! Web Service?
The following exception:

```
Exception occurred during event dispatching:
java.lang.NoClassDefFoundError: org/apache/axis/AxisFault
 at
com.jp.protection.pub.pro.integration.ws.LicensingFacadeWS
...
```

indicates that Apache Axis libraries are missing. You need to add Apache Axis libraries to your deployment. For description and usage of the deployment libraries please see *<install_dir>\lib\deploy.html*.

With the deployed back-end service, is Protection! able to respond to commercial license requests?
Absolutely!

Why am I receiving a corrupted license from Protection! Web Service?
This could be caused if an incorrect or missing license file is specified in Licensing Facade Config Dialog | General Tab | Deployment License. You must specify the location of your commercial or evaluation Protection! License key file (protection.key)

What container can host Protection! Web Service?
Any available web container (free or commercial) supporting Servlet API 2.3 and JSP 1.2 (if you will use JSP) would work.

License Specific Questions

When a license is generated with the 'activation and lock', why don't I see the License Activation dialog?

It seems the issue is caused by missing native libraries. Such libraries are required to obtain the network card address used to lock license to particular computer. To see exceptions, if any, you should turn on verbose mode of `LicenseHost` by calling `setVerbose(true)` method.

Why does the Protection! About Dialog shows "Unlicensed" for a license value?

The About Dialog shows "Unlicensed" state when no license is available, or when license has been read but not yet checked. You can get such a result when you call the About Dialog before checking the license. Most probably you are using different LicenseHost's for license checking, and for showing the About Dialog (this is not the best

implementation pattern).

What is the general license activation sequence for a GUI application?

1. Vendor generates license that requires activation and lock.
2. Vendor provides this license to a customer
3. Customer copies the license file to the right place manually, or starts the application and uses the Licensing Wizard to specify the license file location
4. Customer gets License Activation Wizard prompting him/her to activate license If offline activation is chosen, then he/she should provide the Activation Key shown in the Wizard back to the vendor by e-mail or phone
5. Vendor re-generates the license turning on activation and lock and places supplied by customer Activation Key in the "Activation Key" box in the Protection! Control Center
6. Vendor provides the activated license to the customer

If Protection! Backend is used steps 5-6 will be handled by Protection! Backend automatically.

How do I 'lock' the license to a Network card's MAC address?

You would use the "Activation and Lock" option when generating the license that you would supply to your customer(s) (or embed into the application). When the customer runs your protected application with this license he/she would see License Activation Assistant Wizard and would be able to provide you with a generated activation key. This key is being generated/calculated based on the customer's MAC address.

If you do not use Protection! backend, then you can generate and provide to your customer an activated license by specifying received from the customer activation key in the "Activation Key" box in the Protection! Control Center.

The activation process always requires contacting the vendor/publisher either off-line (e.g. by calling/e-mailing sales) or on-line by using Protection! backend. This allows vendors to:

- Track actual application deployments. For example, if a bundled evaluation license requires an activation, then each evaluator would have to provide his/her credentials before using the application
- Implement Floating User licensing model by enabling usage of only the purchased number of copies by activating only that number of licenses that are specified in the license's number of copies attribute
- Implement the Named User licensing model by locking a license to a particular computer

How does Protection! know that license file is locked to a MAC address and therefore cannot be reused?

Because a License that is being generated is based on the activation key from your customer. The activation key is unique to the customer's system (calculated based on the customer MAC address - default implementation).

If license activation is required, License Host (Pro Edition only) generates Activation Key by calling `getActivationKey()` method and checks whether such property is already present in the license and if so whether it has the same value. Method getActivationKey() actually calls the following methods depending on whether license needs to be locked:

1. `getActivationKey(license)` - generates Activation Key when the license should not be locked. Method's implementation uses concatenation of product and license number hash codes
2. `getActivationLockKey(license)` - generates Activation Key when the license should be locked. The current implementation uses MAC address of the network card to generate this value

If license is moved to a different system, then License host method `licenseLockViolation(…)` can notify listener that the license is activated for use on another computer.

Can Protection! generate specialized keys for our subscribers automatically?
Yes, Protection! can generate specialized license keys. Such license keys could be generated and distributed to application subscribers in several ways:

- Protection! Web Services application could generate a license automatically upon request from the user, and distribute the license either via Direct Internet connection (with support for Proxy) or via e-mail - all of these are controlled by you as a developer
- Protection! could generate either one or multiple Serial Numbers for a given license (e.g. SR84D-VBESQ-GK3RC-F9HM5-ESQGU) You as a developer can also choose if the license, which you provide to the user, needs to be either:
  a) Activated (this would generate a unique activation key for a given license)
  b) Activated and locked (this would lock the license to a specific computer system via the network card MAC address, or some other, specified by your attribute)
  c) You can also specify that the user must provide user's info (name, company, e-mail, etc) during product activation
- Protection! has a built-in sophisticated and highly customizable Licensing and Activation Assistant Wizards that would aid your users in the process of obtaining new/evaluation/extended evaluation/commercial license and/or activating such a license (if necessary)
- And, of course, you can generate a license manually using Protection! Control Center - with all of the above mentioned functionality and attributes

What is the procedure for sending an activated commercial license?
Q. *What exactly is the procedure? Surely you don't want to send out an activated commercial license without having approved the customer's purchase order.*
A. Probably not, but this is again all up to you as a developer/publisher. There are many models that could be implemented on how and when you would decide to distribute an activated commercial license. These models could be manual (require manual license generation and/or license can be generated by backend automatically, but stored in some queue for further review and approval by sales/management before distribution). You can also tie backend to the sales-force application that processes your orders in order for backend to query if a valid purchase was recorded before the commercial license is activated. As you can see, these models could be as complex or as simple as you want them to be. Protection! gives you 100% flexibility to implement any of them by being a Framework, and not imposing any of the models on you as developer/publisher.
In general, the steps could be as follows - for GUI application:

1. Vendor generates license that requires activation and lock
2. Vendor provides this license to a customer
3. Customer copies the license file to the right place manually, or starts the application and uses the Licensing Wizard to specify the license file location
4. Customer gets License Activation Wizard prompting him/her to activate the license If offline activation is chosen, then he/she should provide the Activation Key shown in the Wizard back to the vendor by e-mail or phone
5. Vendor re-generates the license turning on activation and lock and places supplied by customer Activation Key in the "Activation Key" box in the Protection! Control Center
6. Vendor provides the activated license to the customer

If Protection! Backend is used steps 5-6 will be handled by Protection! Backend automatically.

Does the "floating" model require that a separate licensing server to be running?

No, we do not use a licensing server.

How does Activation work at a high-level?

With Activation and Lock: Your user, in order to be able to use your application, would have to provide you an activation key that is generated by Protection! and is *specific* to the user's system. When you generate the license key for this user - this license key would only be able to work on this specific user's system. If license is transferred to another system it would throw necessary events during license reading/validation stage such as (in this case):

```
licenseLockViolation(LicenseHost aSource, License aLicense)
```

In this case, the License host calls this method to notify the listener (your application) that the license is activated for use on another computer. You, as a developer, will need to build some logic into your application to handle (this or other license reading/validation) events. For example, if such event occurs, you can completely disable the application features, or enable only some of them, etc...

Of course, all of the above is applicable if you will use Protection! Web Service to generate and distribute a license key file to your users without your (publisher) interaction.

I like the idea of authenticating the license to a certain machine, but...

***Q**. I like the idea of authenticating the license to a certain machine. However, what happens when the machine is replaced, due to failure or old age? What is the pathway for the user to obtain a valid license for the replacement machine?*

A. This could be handled several ways - however, as you would probably agree, in majority of the cases this is an administrative (your) decision to issue a new license.

In any case, you can create a generic license that would require activation-and-lock. This license could be either evaluation or limited functionality, with a flexible expiration date (when the expiration date is set based on the first run of your application) - whatever. Then, you embed this license inside your application archive.

In brief - you can embed the license into the application archive as follows: You can Specify the resource folder attribute for your product using the Edit Product Dialog | Locations | Resource Folder edit box to let Protection! know where to find the license in your application archive (e.g. /com/acme/app)

Use Protection! code snippet to get a template of the class that should be used to embed Protection! in your application. Generate a new license; make an application archive containing your application classes and the license file.

That's it! Now you can deploy your application with the bundled license. Please note - you can use both licenses (embedded and located within the file system). This way, you can deploy your application with a bundled evaluation license and when your customer purchases a commercial license, he/she should simply copy their license to your <application_root>/license/ folder, for example.

Back to the question: Let's assume that the user reinstalls his system, changes hardware, etc, etc. - in this case, the user either would need to reinstall your application in which case it would be using embedded license until the user requests a full commercial one. Or, another example would be if you lock your license to a network card MAC address and the user changes the network card (or if license is transferred to another system). In this case Protection! would throw necessary events during license reading/validation stage such as (in this case):

- `licenseLockViolation(LicenseHost aSource, License aLicense)`

Protection! License host calls this method to notify listener (your application) that the license is activated for use on another computer. You, as a developer, will need to build some logic into your application to handle (this one and/or other license reading/validation) events. For example if such an event occurs, you can completely

disable the application's features, or enable only some of them, or call Protection! Licensing Assistant Wizard to aid the user on resolving this issue, etc, etc, etc...
Demo Calculator (a sample application that comes with the Protection!) illustrates this and many other examples.

What prevents a user from reusing license keys?

Q. *What prevents a user from reusing license keys, to install and use an unlimited number of copies of the product?*

- Protection! has several built-in mechanisms to prevent this: Number of copies violation - where you indicate how many copies of your product would be able to run on the network (floating license model).
- Activation and/or activation-and-lock mechanism - where you lock the license to a specific computer system (named-user licensing model)

And of course - because Protection! is a framework you can mix-and-match existing mechanisms, extend them, or implement your own.

Can online licensing be run on a Linux server?

Protection! is 100% pure Java so it will run on any platform that supports Java. We have successful implementations on various flavors of UNIX and Linux, MAC OS X and of course Windows.
The one and only time where there are some platform dependencies is when the "activation-and-lock" mechanism is chosen. Protection! default implementation is able to lock a license to a network card MAC address, and therefore uses some native libraries (provided with Protection!). Of course, if other lock mechanisms are chosen by the developer, then native support may or may not be necessary (i.e. locking license to an IP address of the host will not require native support and therefore would be platform independent).

How fast is the Protection! back-end response to user request?

Q. *With respect to a license file or a serial number that requires activation, if the user decides to 'activate online' will the back-end respond immediately and send out an activated license file/sn?*

A. This would generally depend on when/how/where your back-end is deployed. Even though in general I would answer yes, but you would agree, that there are a lot of variables that could delay distribution of an activated license file. All of these variables are 100% in your control, however. These variable could be your backend business logic (what would happen when customer requests license file, would you do this automatically or would you require management/sales approval, would you tie Protection! backend with your sales-force, CRM, ERP applications, etc, etc, etc...) and of course variables like backend server horsepower, network connections, etc, etc, etc...)

Is it possible to transfer a license to another computer system after it has been activated and locked?

No. The license key file would be locked to a specific user system. Default implementation locks the license to a network card MAC address of the user's system. However other, user-definable, attributes could be used by you, as a developer, as well.

Is it possible to embed the license file inside application archive jar file?

You can Specify a resource folder attribute for your product using the Edit Product Dialog | Locations | Resource Folder edit box to let Protection! know where to find the license in your application archive (e.g. */com/acme/app*)
Use Protection! code snippet to get a template of the class that should be used to embed Protection! in your application.
Generate a new license; make an application archive containing your application classes and the license file.

That's it! Now you can deploy your application with the bundled license. Please note - you can use both licenses (embedded and located within the file system). This way you can deploy your application with a bundled evaluation license (for example) and when your customer purchases a commercial license, he/she should simply copy their license to your *<application_root>/license/* folder, for example.

I don't really understand the concept of "activation"

Q. *I don't really understand the concept of "activation" in license generation panel: what is the benefit of the activation key if finally you have to send a second license to your customer?*

A. The reason for such approach is to provide secure storage for the activation key and to prevent customers from saving it manually. Everything on the customer's computer needs to be considered insecure (e.g. file system or registry). Therefore, it is very easy to use a special application to track activity of the protected application - find what it saves, when and where. Our approach of placing the activation key into the license and sending it back to the customer is very safe as the client side is only capable of decoding licenses, therefore making it impossible to place the activation key into the license. This process only looks complicated, however. With the help of Protection! Activation wizards and Protection! Backend it is completely transparent to the customers and vendors.

Activation process always requires contacting the vendor/publisher either off-line (e.g. by calling/e-mailing sales) or on-line by using Protection! backend. This allows vendors to:

- Track actual application deployments. For example, if a bundled evaluation license requires activation, then each evaluator would have to provide his/her credentials before using your application
- Implement Floating User licensing model by enabling usage of only the purchased number of copies by activating only that number of licenses that is specified in the license's number of copies attribute
- Implements Named User licensing model by locking the license to a particular computer

Why does the license return an UNKNOWN_STATE?

This could happen if the developer is simply using License Reader to get license. License Reader is responsible only for discovery and decoding of the license. Developers should be using License Host in order to read and check a license. License Host uses License Reader, so developers do not need to use (even though they can) License Reader directly.

Please see `licenseHost.checkLicense(...)` method and also demo application that comes with Protection (samples directory) - specifically DemoCalcProtectionSupport.java.

How can I implement the automatic license generation and activation mechanism?

This would require use of the Protection! Backend.

CRC Check Support

Problems with CRC checking when packaging is changed

Of course. CRC uses fully qualified class/resource names:

```
class DemoCalcCRCSupport
{
  private static final String[] CRC_ENTRIES =
```

```
  {
    "com/jp/samples/protection/DemoCalcProtectionSupport.class",
    "com/jp/samples/protection/DemoCalcProtectionSupport$1.class"
  };

  public static boolean checkCRC()
  {
      return CRCHost.checkStatic(CRC_ENTRIES, CRC);
  }
}
```

And of course if packaging is changed the code snippet needs to be modified accordingly.

Why did CRC checking fail after classes obfuscation?

Because obfuscation changes names and packages of the classes in most of cases it is not possible to find them and so check their integrity.

How can I prohibit possible tampering with application code?

Developers can prohibit tempering with a protected application by utilizing the CRC verification mechanism. During the product creation, you can assign CRC checking to any number of resources in your application. During runtime, the developer can verify if CRC values for any of these resources were changed - which would indicate a potential tampering with the application code. At this point, the developer can apply his/her own logic, indicating how the developer wants to handle this situation.

Here is the snippet from Protection! User Guide:

"...CRC verification subsystem allows for the checking and validation that the designated key classes of the product have not changed. This type of check is done by comparing the CRC of these classes with the original value stored somewhere in the product code, resources or files. This subsystem significantly increases time and effort needed to diagnose and locate this part of the protection system in order to attempt to break it...."