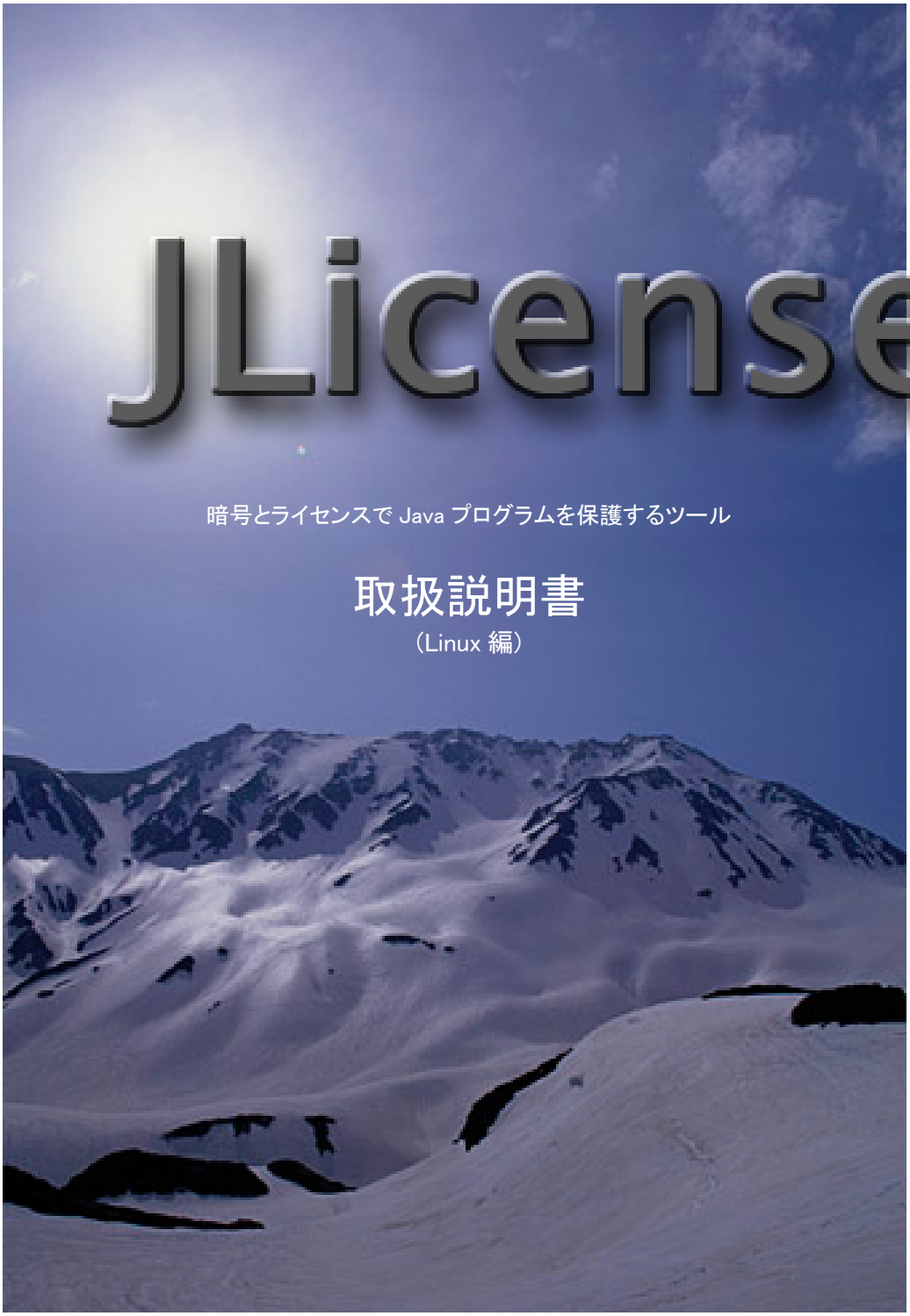


対応するソフトウェアのバージョンは、JLicenser 0.3.4です。

目次

1. はじめに

大金システム設計事務所が提供する「JLicenser – 暗号化とライセンスで Java プログラムを保護するツール」(以下「本ソフトウェア」という。)を使用する前に必ず以下の使用条件をお読み下さい。

- **n** 本ソフトウェア及び本ソフトウェアに付属する全てのマニュアル、書類等は、大金システム設計事務所が著作権、その他の知的財産権を持ちます。従って、利用者は本ソフトウェアを他の著作物と同様に取り扱わなければなりません。
- **n** 本ソフトウェアは、大金システム設計事務所が定めた配布元から無償で配布されなければなりません。大金システム設計事務所の許可なく、本ソフトウェアを第三者に対して再配布してはなりません。
- **n** 大金システム設計事務所の承諾無しに本ソフトウェアの一部または全部を転載または複写することは固くお断りいたします。
- **n** 本ソフトウェアを解析、逆アセンブル、逆コンパイルなどをすることを禁止します。
- **n** 本ソフトウェアを運用した結果や影響につきましては上項にかかわらず責任を負いかねますのでご了承下さい。

(1)　Java およびすべての Java 関連の商標およびロゴは、米国およびその他の国における米国 Sun Microsystems, Inc. の商標または登録商標です。

(2)　その他、記載されている会社名・製品名・サービス名は、各社の商標または登録商標です。

2. 機能

本ソフトウェアは、暗号化とライセンスで Java プログラムを保護するツールです。Java プログラムは、バイトコードの仕様上、簡単に逆コンパイルできてしまいます。これまで、曖昧化などの方法が使われていますが完全ではありません。また、作成した Java プログラムを特定の相手にだけ使用権を認めて配布することが必要になる場合もあります。そこで、次のような機能を持ったツールを開発しました。

(1)　クラスを暗号化することによって、逆コンパイルが完全に不可能になります。
(2)　暗号化の鍵は、公開鍵暗号方式により暗号化して使用者に渡すので、正当な使用者のみが安全に復号できます。
(3)　使用者に付与する使用権(ライセンス)には、次の条件を組み合わせることができます。

- ホスト名
- 物理アドレス
- IP アドレス
- 使用期限
- 独自のライセンス条件

(4)　開発者はプログラムを追加することによって独自のライセンス条件を追加することができます。
(5)　使用権(ライセンス)は、署名付きのファイルに記述して使用者に配布します。正当な使用権を持たない者が、これを入手し、更に改竄などの不正な操作を行ってもプログラムは起動できません。

3. 動作環境

開発環境：JDK1.5 以上が動作する Windows、Linux
実行環境：JRE1.5 以上が動作する Windows、Linux

4. インストール方法

（1） 本ソフトウェアの配布ファイルはZIP形式で圧縮してありますので、各種のツールを使用して任意のディレクトリに解凍して下さい。解凍すると次のファイルが展開されます。

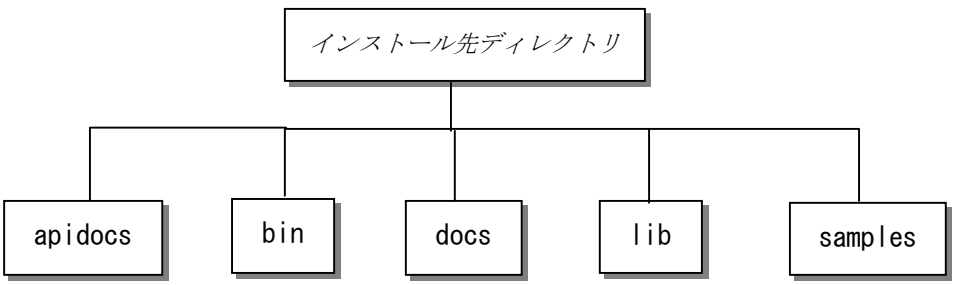
- setup.bin
- readme.txt
- license.txt
- usersman-linux.pdf（この文書ファイル）

（2） 解凍先のディレクトリにてインストーラを起動してください。インストーラは、（1）で展開したsetup.binです。

```
# chmod +x setup.bin
# ./setup.bin
```

（3） インストールが正常に終了すると、下の図のようにインストール先ディレクトリの下に５つのサブディレクトリが展開されます。なお、標準のインストール先ディレクトリは、次の通りです。

/usr/local/JLicenser-*バージョン番号*



５．アンインストール方法

（１）インストール先ディレクトリを削除するだけです。

```
# rmdir インストール先ディレクトリ
```

6. 初期設定

この章では、インストール後にご使用の環境に合わせて一回だけ行う初期設定の方法について説明します。

（1） JL_HOME 環境変数およびコマンド実行パスを設定する以下のコマンドを、シェル起動スクリプトに追加します。

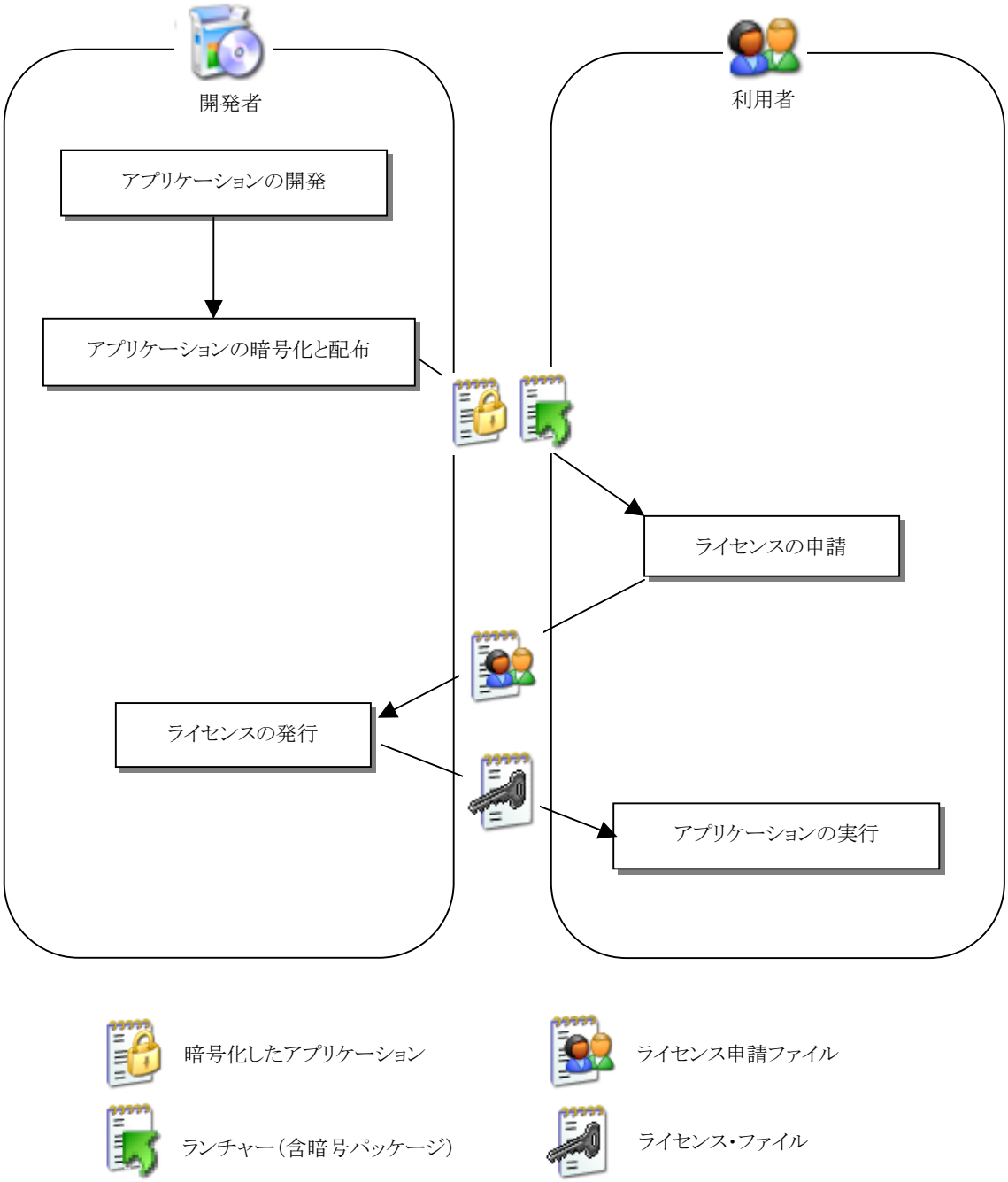
```
export JL_HOME=インストール先ディレクトリ
export PATH=$PATH:$JL_HOME/bin
```

（2） 引数なしで jl コマンドを入力して、コマンドの使用法が表示されれば初期設定は完了です。

```
# jl
JLicenser v0.3.1 (c) 2003-2008 Yasuo Ogane <y.ogane@ogane.com>
使用法-1: ツールを初期化する

jl -init

使用法-2: JAR ファイルまたはクラス・ファイルを暗号化する

jl filenames...

使用法-3: ライセンスを発行する

jl -lic [file1] [-out file2]
[-vend vendor] [-prod product] [-ver version]
[-user username] [-ext "name=value[,name=value]..."]

使用法-4: コマンド使用方法を表示する

jl -help

#
```

7. アプリケーションの開発から実行まで

あなたが開発したアプリケーションが利用者の手元で実行されるまでの流れは下の図の通りです。この章では、これら一連の手順について順番に説明します。



７．１　アプリケーションの開発

この手順は開発者側で行います。

（1）　当然のことですが、先ずはアプリケーションを製作しなければなりません。製作する方法は、次のいずれでもかまいません。

- 統合開発環境を使う
- JDK のコマンドを直接叩く
- Ant を使う
- その他

ただし、結果として利用者に配布する形体としては、ばらばらのクラスファイルにするよりは、JAR ファイルにパッケージする方法をお勧めします。以下の説明でも、JAR ファイルにパッケージした場合についてだけ説明します。

７．２　アプリケーションの暗号化と配布

この手順は開発者側で行います。

作業ディレクトリにて次のコマンドを入力します。

```
jl アプリケーションの JAR ファイル名
```

このコマンドを実行すると次の例のように、指定した JAR ファイルが暗号化されます。

例)アプリケーションのパッケージ myapp.jar を暗号化します。

```
> jl myapp.jar
[INFO] 暗号化しました mypackage/MyAppClass.class (myapp.jar)
                    :
```

次のファイルを利用者に配布します。ここで、ランチャーとは、暗号化されたクラスを起動するために使用するツールです。

- (1)で暗号化したアプリケーションの JAR ファイル
- ランチャー(*インストール先ディレクトリ*¥lib¥launcher.jar)
- 暗号パッケージ(*インストール先ディレクトリ*¥lib¥bcprov-*.jar)

７．３　ライセンスの申請

この手順は利用者側で行います。

(1)　利用者は、受取ったファイルを任意のディレクトリに配備します。ただし、ランチャーと暗号パッケージは同じディレクトリに配備すること。

(2)　利用者は、次のコマンドでライセンス申請ファイルを作成します。

```
java -jar launcher.jar -reg [filename]
[-user username] [-prod product] [-ver version]
```

ここで、filename は作成するライセンス申請ファイルの名前(既定値は user.properties)です。ライセンス申請ファイルには、自動的に次の項目が記入されます。

- 申請した日付
- 使用期限(＝申請した日付)
- 申請したマシンのホスト名
- 申請したマシンの IP アドレス
- 申請したマシンの物理アドレス
- 申請を実行したカレントユーザー名

username を指定すると、上記のカレントユーザー名が上書きされて、username が申請者名となります。この他に、コマンド行からいくつかの項目を追加できます。product ではプログラム名、version ではプログラムのバージョンが追加できます。このコマンドを実行した結果、ライセンス申請ファイルが作成されます。利用者は、このライセンス申請ファイルを開発者へ送付します。

例)カレントディレクトリにライセンス申請ファイルを作成します。

```
> java -jar "%JL_HOME%¥lib¥launcher.jar" -reg
[INFO] ライセンス申請ファイルを作成しました user.properties
```

７．４　ライセンスの発行

この手順は開発者側で行います。

(1)　利用者から送付されたライセンス申請ファイルを任意のディレクトリに保存します。

(2)　次に、このライセンス申請ファイルをテキストエディターで配布するアプリケーションの使用条件に合わせて変更します。

```
# 使用期限（制限しない場合は*）
license.expired=07-Jun-2008
# 使用できるホスト名（制限しない場合は*）
license.hostname=moon
# 使用できる IP アドレス（制限しない場合は*）
license.inetaddr=192.168.1.4
# 申請日
license.issued=07-Jun-2008
# 申請者
license.licensee=¥u5927¥u91D1¥u5EB7¥u592B
# 使用できる物理アドレス（制限しない場合は*）
license.macaddr=00¥:11¥:11¥:BF¥:C2¥:88
# 利用者鍵（変更してはなりません）
secure.userkey=305C300D06092A864886F70D0101010500034B003048024100B8AD09DC0C643
836C646DC5BFD1A8079B3A4362F969109391088CD19B0B1B605D9C133FA1EADD864227C8C69CDD
430A5DB720968ABD08A61D013F91EA4D0AF150203010001
```

(3)　次のコマンドでライセンス・ファイルを作成します。

```
jl -lic [file1] [-out file2]
[-vend vendor] [-prod product] [-ver version]
[-user username] [-ext "name=value[,name=value]..."]
```

ここで、file1 は、使用するするライセンス申請ファイルの名前（既定値は user.properties）です。file2 は、作成するライセンス・ファイルの名前（既定値は license.properties）です。このコマンドを実行した結果、ライセンス・ファイルが作成されます。このライセンス・ファイルを申請してきた利用者に返送します。-vend、-prod、-ver オプションを指定すると、それぞれ開発者の名前、プログラムの名称、プログラムのバージョンをライセンス・ファイルに追加することができます。また、-user オプションを指定すると、プログラムの使用者の名前を指定できます。さらに、-ext オプションを指定することによって、作成するライセンス・ファイルに任意の項目を追加できます。追加項目の名前を name で、それに対する値を value で指定します。ただし、本ソフトウェアがライセンス条件として検査する項目は、次の項目だけです。

- ホスト名
- 物理アドレス
- IP アドレス
- 使用期限

追加した項目を独自のライセンス条件として利用したり、その他の目的で利用することも可能です。そのためには、配布するソフトウェアの中に独自のプログラムを書く必要があります。詳しくは「10．4 独自のライセンス条件を加える」をご覧下さい。

例）カレントディレクトリにライセンス・ファイルを作成します。

```
> jl -lic
[INFO] ライセンス・ファイルを作成しました license.properties
```

例）ライセンス・ファイルにソフトウェアの名前とバージョンを追加します。

```
> jl -lic -prod SimpleApp -ver 1.0.0
[INFO] ライセンス・ファイルを作成しました license.properties
```

７．５　アプリケーションの実行

この手順は利用者側で行います。

（１）利用者は、受取ったライセンス・ファイルを任意のディレクトリに配備します。
（２）利用者は、あなたのアプリケーションを次のようにして起動します。

```
java [オプション] -classpath クラスパス com.ogane.jl.Launcher メインクラス [引数...]
```

ここで、コマンド行内の各要素は次の表の通り指定します。クラスパスはWindowsの場合は「;」(セミコロン)、Linux の場合は「:」(コロン)で区切ります。なお、ライセンス・ファイルは、既定ではカレントディレクトリに license.properties という名前であるものとします。これ以外の場所にライセンス・ファイルがある場合は、-D オプションを使って次のように指定します。

-Djl.license.file=*ライセンス・ファイルのフルパス*

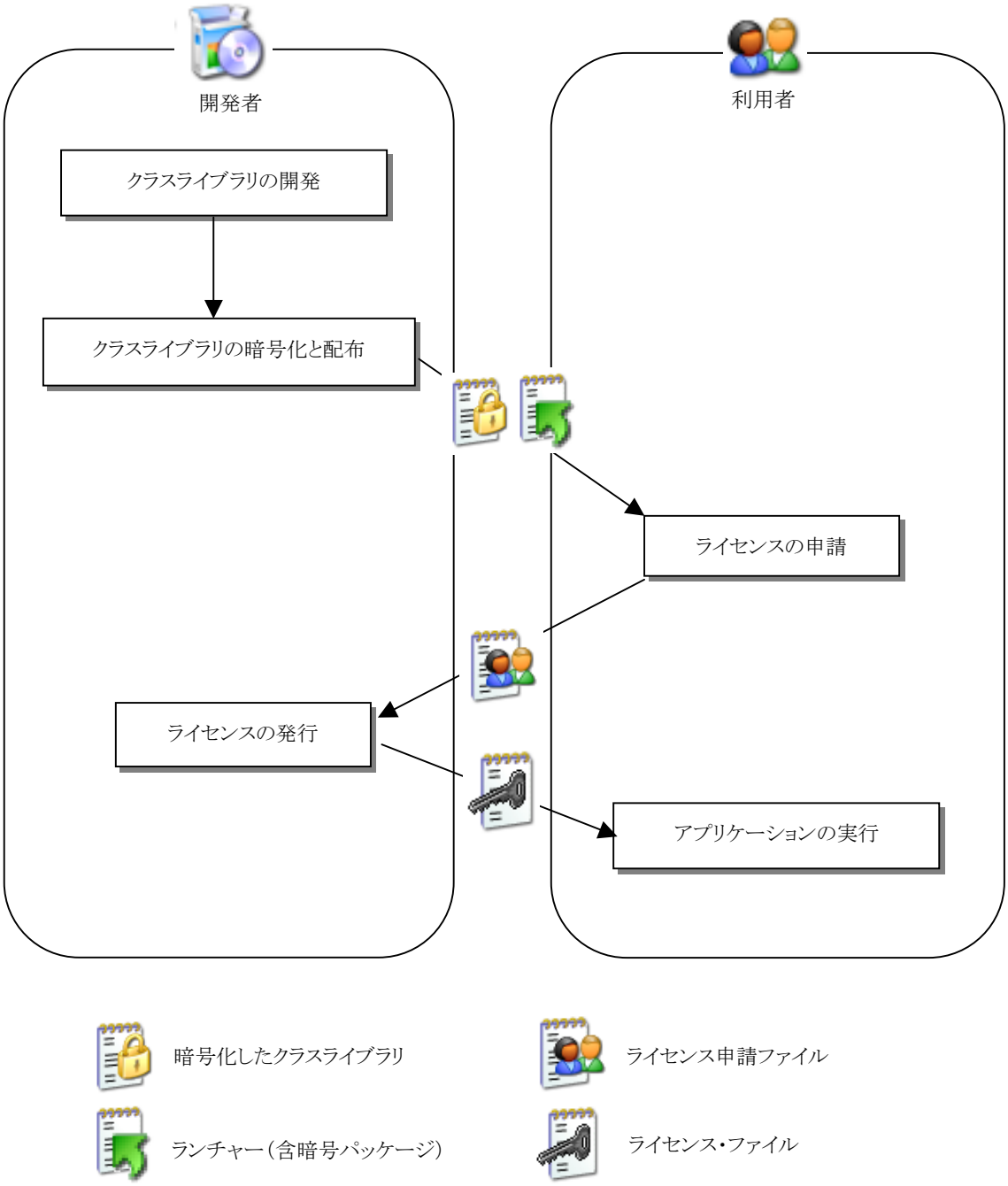
コマンド内要素	指定内容	例	適用
オプション	Java コマンドのオプション	-Dmyapp.option=123	適宜
クラスパス	配布されたランチャー	launcher.jar	必須
	暗号化したアプリケーション	myapp.jar	必須
	アプリケーションに必要な API など	j2ee.jar	適宜
メインクラス	アプリケーションのメインクラスの名前	mypackage.MyAppMain	必須
引数	アプリケーションの引数	arg1 arg2 arg3	適宜

例）暗号化したアプリケーション mypackage.MyAppMain を起動します。

```
> java -classpath launcher.jar;myapp.jar com.ogane.jl.Launcher mypackage.MyAppMain arg1 arg2 arg3
```

8. クラスライブラリの開発から実行まで

あなたが開発したクラスライブラリが利用者の手元で実行されるまでの流れは下の図の通りです。この章では、これら一連の手順について順番に説明します。



８．１　クラスライブラリの開発

この手順は開発者側で行います。

(1)　当然のことですが、先ずはクラスライブラリを製作しなければなりません。製作する方法は、次のいずれでもかまいません。

- 統合開発環境を使う
- JDK のコマンドを直接叩く
- Ant を使う
- その他

一般的にクラスライブラリは JAR ファイルにパッケージします。以下の説明でも、JAR ファイルにパッケージした場合についてだけ説明します。

８．２　クラスライブラリの暗号化と配布

この手順は開発者側で行います。

（１）　作業ディレクトリにて次のコマンドを入力します。

```
jl クラスライブラリの JAR ファイル名
```

このコマンドを実行すると次の例のように、指定した JAR ファイルが暗号化されます。

例）クラスライブラリのパッケージ mylib.jar を暗号化します。

```
> jl mylib.jar
[INFO] 暗号化しました mypackage/MyLibClass.class (mylib.jar)
                    :
```

(2) 次のファイルを利用者に配布します。ここで、ランチャーとは、暗号化されたクラスを起動するために使用するツールです。

- (1)で暗号化したクラスライブラリの JAR ファイル
- ランチャー(*インストール先ディレクトリ*¥lib¥launcher.jar)
- 暗号パッケージ(*インストール先ディレクトリ*¥lib¥bcprov-*.jar)

８．３　ライセンスの申請

この手順は利用者側で行います。

(1)　利用者は、受取ったファイルを任意のディレクトリに配備します。ただし、ランチャーと暗号パッケージは同じディレクトリに配備すること。

(2)　利用者は、次のコマンドでライセンス申請ファイルを作成します。

```
java -jar launcher.jar -reg [filename]
[-user username] [-prod product] [-ver version]
```

ここで、filename は作成するライセンス申請ファイルの名前(既定値は user.properties)です。ライセンス申請ファイルには、自動的に次の項目が記入されます。

- 申請した日付
- 使用期限(＝申請した日付)
- 申請したマシンのホスト名
- 申請したマシンの IP アドレス
- 申請したマシンの物理アドレス
- 申請を実行したカレントユーザー名

username を指定すると、上記のカレントユーザー名が上書きされて、username が申請者名となります。この他に、コマンド行からいくつかの項目を追加できます。product ではプログラム名、version ではプログラムのバージョンが追加できます。このコマンドを実行した結果、ライセンス申請ファイルが作成されます。利用者は、このライセンス申請ファイルを開発者へ送付します。

例)カレントディレクトリにライセンス申請ファイルを作成します。

```
> java -jar "%JL_HOME%¥lib¥launcher.jar" -reg
[INFO] ライセンス申請ファイルを作成しました user.properties
```

８．４　ライセンスの発行

この手順は開発者側で行います。

(1)利用者から送付されたライセンス申請ファイルを任意のディレクトリに保存します。

(2)次に、このライセンス申請ファイルをテキストエディターで配布するアプリケーションの使用条件に合わせて変更します。

```
# 使用期限（制限しない場合は*)
license.expired=07-Jun-2008
# 使用できるホスト名（制限しない場合は*)
license.hostname=moon
# 使用できる IP アドレス（制限しない場合は*)
license.inetaddr=192.168.1.4
# 申請日
license.issued=07-Jun-2008
# 申請者
license.licensee=¥u5927¥u91D1¥u5EB7¥u592B
# 使用できる物理アドレス（制限しない場合は*)
license.macaddr=00¥:11¥:11¥:BF¥:C2¥:88
# 利用者鍵（変更してはなりません）
secure.userkey=305C300D06092A864886F70D0101010500034B003048024100B8AD09DC0C643
836C646DC5BFD1A8079B3A4362F969109391088CD19B0B1B605D9C133FA1EADD864227C8C69CDD
430A5DB720968ABD08A61D013F91EA4D0AF150203010001
```

(3)次のコマンドでライセンス・ファイルを作成します。

```
jl -lic [file1] [-out file2]
[-vend vendor] [-prod product] [-ver version]
[-user username] [-ext "name=value[,name=value]..."]
```

ここで、file1 は、使用するするライセンス申請ファイルの名前（既定値は user.properties）です。file2 は、作成するライセンス・ファイルの名前（既定値は license.properties）です。このコマンドを実行した結果、ライセンス・ファイルが作成されます。このライセンス・ファイルを申請してきた利用者に返送します。-vend、-prod、-ver オプションを指定すると、それぞれ開発者の名前、プログラムの名称、プログラムのバージョンをライセンス・ファイルに追加することができます。また、-user オプションを指定すると、プログラムの使用者の名前を指定できます。さらに、-ext オプションを指定することによって、作成するライセンス・ファイルに任意の項目を追加できます。追加項目の名前を name で、それに対する値を value で指定します。ただし、本ソフトウェアがライセンス条件として検査する項目は、次の項目だけです。

- ホスト名
- 物理アドレス
- IP アドレス
- 使用期限

追加した項目を独自のライセンス条件として利用したり、その他の目的で利用することも可能です。そのためには、配布するソフトウェアの中に独自のプログラムを書く必要があります。詳しくは「10．4 独自のライセンス条件を加える」をご覧下さい。

例）カレントディレクトリにライセンス・ファイルを作成します。

```
> jl -lic
[INFO] ライセンス・ファイルを作成しました license.properties
```

例）ライセンス・ファイルにソフトウェアの名前とバージョンを追加します。

```
> jl -lic -prod SimpleApp -ver 1.0.0
[INFO] ライセンス・ファイルを作成しました license.properties
```

８．５　アプリケーションの実行

この手順は利用者側で行います。

（1）　利用者は、受取ったライセンス・ファイルを任意のディレクトリに配備します。

（２）利用者は、暗号化されたクラスライブラリを使うアプリケーションを製作します。暗号化されたクラスライブラリのクラスは、Jlicenser の com.ogane.jl.Launcher クラスによって操作します。Launcher クラスの仕様については、別途「API 仕様書」をご覧下さい。

Launcher クラスのメソッド	機能
getInstance	Launcher クラスのインスタンスを取得する
construct	暗号化されたクラスのインスタンスを生成する
execute	暗号化されたクラスのメソッドを実行する

アプリケーションをコンパイルするときは、必ずクラスパスに配布されたランチャーを指定すること。

例）利用者が作成したアプリケーション UserApp.java をコンパイルします。

```
> javac -claspass launcher.jar UserApp.java
```

例)暗号化されたクラスライブラリを使用する利用者側アプリケーション UserApp.java

```
import java.io.*;
import java.util.*;
import com.ogane.jl.*;

/**
 * 暗号化されたクラスライブラリを使用するコンソールアプリケーション。
 */
public class UsersApp
{
	public static void main(String[] args)
	{
		try
		{
			new UsersApp();
		}
		catch(Exception ex)
		{
			ex.printStackTrace();
		}

	}

	public UsersApp() throws Exception
	{
		/*
		 * 使用するクラスライブラリが暗号化されていない普通のコーディングでは...
		 *
		 *    foobar.Foo foo = foobar.Foo( 28 );
		 *    String name = foo.getName();
		 *    int age = foo.getAge();
		 *
		 *    System.out.println(name + " is " + age + " years old.");
		 *
		 * と書くところを、暗号化されている場合は次のように書く。
		 *
		 */

		// ライセンスファイル
		File licfile = new File("license.properties");

		// Launcher のインスタンスを取得する
		Launcher launcher = Launcher.getInstance(licfile);

		// 暗号化されたクラス Foo のインスタンスを生成する
		Object foo = launcher.construct("foobar.Foo", new Object[]{ 28 });

		// 暗号化されたクラス Foo のメソッドを実行する
		Object name = launcher.execute(foo, "getName");
		Object age  = launcher.execute(foo, "getAge");

		System.out.println(name + " is " + age + " years old.");

	}

}
```

(3)　利用者は、作成したアプリケーションを次のようにして起動します。

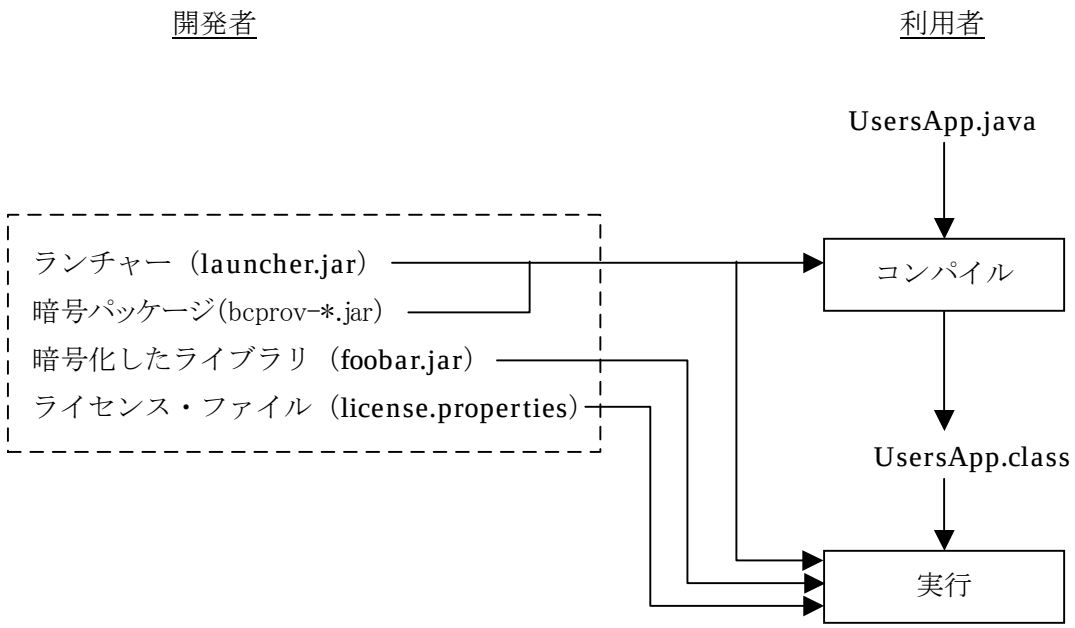
```
java [オプション] -classpath クラスパス メインクラス [引数...]
```

ここで、コマンド行内の各要素は次の表の通り指定します。クラスパスはWindowsの場合は「;」(セミコロン)、Linux の場合は「:」(コロン)で区切ります。なお、ライセンス・ファイルは、既定ではカレントディレクトリに license.properties という名前であるものとします。これ以外の場所にライセンス・ファイルがある場合は、-D オプションを使って次のように指定します。

コマンド内要素	指定内容	例	適用
オプション	Java コマンドのオプション	-Dmyapp.option=123	適宜
クラスパス	配布されたランチャー	launcher.jar	必須
	暗号化されたクラスライブラリ	foobar.jar	必須
	アプリケーションに必要な API など	j2ee.jar	適宜
メインクラス	アプリケーションのメインクラスの名前	mypackage.MyAppMain	必須
引数	アプリケーションの引数	arg1 arg2 arg3	適宜

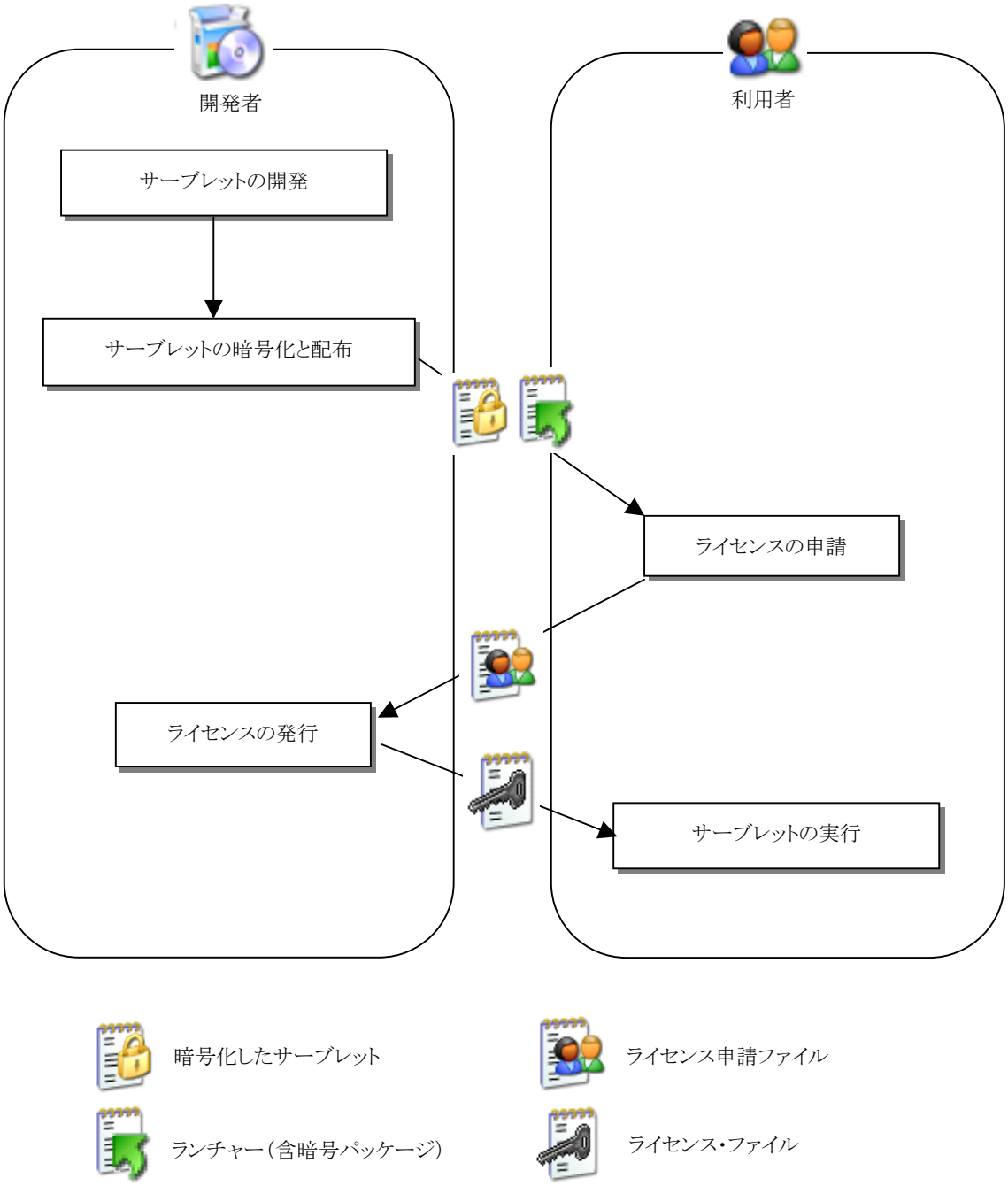
例) 暗号化されたクラスライブラリ foobar.jar を使用するアプリケーション UsersApp を起動します。

```
> java -classpath "launcher.jar;foobar.jar;." UsersApp
```



9. サーブレットの開発から実行まで

あなたが開発したサーブレットが利用者の手元で実行されるまでの流れは下の図の通りです。この章では、これら一連の手順について順番に説明します。



９．１　サーブレットの開発

この手順は開発者側で行います。

(1) 当然のことですが、先ずは暗号化の対象となるサーブレットを製作しなければなりません。製作する方法は、次のいずれでもかまいません。

- 統合開発環境を使う
- JDK のコマンドを直接叩く
- Ant を使う
- その他

一般的にサーブレットはサーブレットコンテナへ配備できる標準の WAR(Web ARchive)形式にパッケージします。以下の説明でも WAR パッケージした場合について説明します。

９．２　サーブレットの暗号化と配布

この手順は開発者側で行います。

（１）　作業ディレクトリにて次のコマンドを入力します。

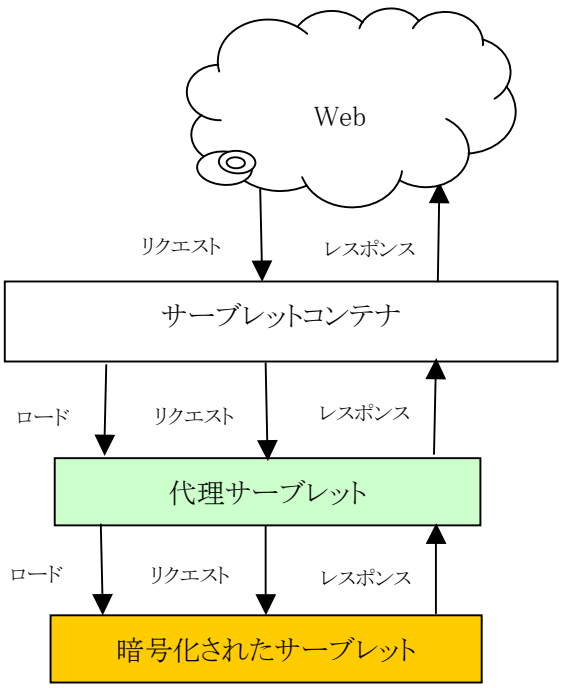
```
jl サーブレットの class ファイル名
```

このコマンドを実行すると次の例のように、指定したサーブレットのクラスファイルが暗号化されます。暗号化したサーブレットのクラスファイルは、WAR の標準フォルダ WEB-INF¥classes に配置します。

例）サーブレット foobar.FooBarServlet を暗号化します。

```
> jl warbase¥WEB-INF¥classes¥foobar¥FooBarServlet.class
[INFO] 暗号化しました warbase¥WEB-INF¥classes¥foobar¥FooBarServlet.class
```

(2) WARの展開記述子(web.xml)を作成します。サーブレットコンテナには暗号化されたサーブレットを復号する機能がないので、暗号化されたサーブレットを直にサーブレットコンテナに配備したのではエラーになります。その代わりに、本ソフトウェアが用意している、代理サーブレットを使用します。この代理サーブレットは、サーブレットコンテナと暗号化されたサーブレット間を中継することにより、暗号化されたサーブレットの代理として働きます。代理サーブレット自体は暗号化されていないためサーブレットコンテナに配備して動作します。



代理サーブレットのクラス名は、com.ogane.jl.ProxyServlet で、必ず次の初期パラメータを指定する必要があります。

初期パラメータ	値
ProtectedServletClass	暗号化されたサーブレットのクラス名
LicenseFile	ライセンスファイルの名前

例）warbase¥WEB-INF¥web.xml

foobar.FooBarServlet を暗号化して使用する例。

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

  <servlet>
    <servlet-name>ProxyServlet</servlet-name>
    <servlet-class>com.ogane.jl.ProxyServlet</servlet-class>
    <init-param>
      <param-name>ProtectedServletClass</param-name>
      <param-value>foobar.FooBarServlet</param-value>
    </init-param>
    <init-param>
      <param-name>LicenseFile</param-name>
      <param-value>C:/Program Files/JLicenser-0.3.2/samples/servlet/license.properties</param-value>
    </init-param>
    <init-param>
      <param-name>foobar.message</param-name>
      <param-value>&lt;i&gt;Hello world! ようこそ！&lt;/i&gt;</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>ProxyServlet</servlet-name>
    <url-pattern>FooBarServlet</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
          <welcome-file>index.html</welcome-file>
  </welcome-file-list>

</web-app>
```

（３）　WAR 形式にパッケージを作成します。

例）warbase ディレクトリ以下に作成した、暗号化したサーブレットを含む Web アプリケーションをパッケージして foobar.war を作成します。

```
> jar cvf foobar.war -C warbase .
```

(4) 次のファイルを利用者に配布します。ここで、ランチャーとは、暗号化されたクラスを起動するために使用するツールです。

- (3)で作成した WAR ファイル
- ランチャー(*インストール先ディレクトリ*¥lib¥launcher.jar)
- 暗号パッケージ(*インストール先ディレクトリ*¥lib¥bcprov-*.jar)

９．３　ライセンスの申請

この手順は利用者側で行います。

(1)利用者は、受取ったWARファイルをサーブレットコンテナ上にあるWebアプリケーション配備用ディレクトリに配備します。また、ランチャーと暗号パッケージは、同じクラスライブラリ配備用ディレクトリに配備します。

例）Apache Tomcat/6.0.14の場合

Webアプリケーション配備用ディレクトリ	%CATALINA_HOME%¥webapps
クラスライブラリ配備用ディレクトリ	%CATALINA_HOME%¥lib や %CATALINA_HOME%¥webapps¥foobar¥WEB-INF¥lib など

(2)利用者は、次のコマンドでライセンス申請ファイルを作成します。

```
java -jar launcher.jar -reg [filename]
[-user username] [-prod product] [-ver version]
```

ここで、filenameは作成するライセンス申請ファイルの名前(既定値はuser.properties)です。ライセンス申請ファイルには、自動的に次の項目が記入されます。

- 申請した日付
- 使用期限(＝申請した日付)
- 申請したマシンのホスト名
- 申請したマシンの IP アドレス
- 申請したマシンの物理アドレス
- 申請を実行したカレントユーザー名

usernameを指定すると、上記のカレントユーザー名が上書きされて、usernameが申請者名となります。この他に、コマンド行からいくつかの項目を追加できます。product ではプログラム名、version ではプログラムのバージョンが追加できます。このコマンドを実行した結果、ライセンス申請ファイルが作成されます。利用者は、このライセンス申請ファイルを開発者へ送付します。

例)カレントディレクトリにライセンス申請ファイルを作成します。

```
> java -jar "%JL_HOME%¥lib¥launcher.jar" -reg
[INFO] ライセンス申請ファイルを作成しました user.properties
```

９．４　ライセンスの発行

この手順は開発者側で行います。

(1)利用者から送付されたライセンス申請ファイルを任意のディレクトリに保存します。

(2)次に、このライセンス申請ファイルをテキストエディターで配布するアプリケーションの使用条件に合わせて変更します。

```
# 使用期限（制限しない場合は*)
license.expired=07-Jun-2008
# 使用できるホスト名（制限しない場合は*)
license.hostname=moon
# 使用できる IP アドレス（制限しない場合は*)
license.inetaddr=192.168.1.4
# 申請日
license.issued=07-Jun-2008
# 申請者
license.licensee=¥u5927¥u91D1¥u5EB7¥u592B
# 使用できる物理アドレス（制限しない場合は*)
license.macaddr=00¥:11¥:11¥:BF¥:C2¥:88
# 利用者鍵（変更してはなりません）
secure.userkey=305C300D06092A864886F70D0101010500034B003048024100B8AD09DC0C643
836C646DC5BFD1A8079B3A4362F969109391088CD19B0B1B605D9C133FA1EADD864227C8C69CDD
430A5DB720968ABD08A61D013F91EA4D0AF150203010001
```

(3)次のコマンドでライセンス・ファイルを作成します。

```
jl -lic [file1] [-out file2]
[-vend vendor] [-prod product] [-ver version]
[-user username] [-ext "name=value[,name=value]..."]
```

ここで、file1 は、使用するするライセンス申請ファイルの名前（既定値は user.properties）です。file2 は、作成するライセンス・ファイルの名前（既定値は license.properties）です。このコマンドを実行した結果、ライセンス・ファイルが作成されます。このライセンス・ファイルを申請してきた利用者に返送します。-vend、-prod、-ver オプションを指定すると、それぞれ開発者の名前、プログラムの名称、プログラムのバージョンをライセンス・ファイルに追加することができます。また、-user オプションを指定すると、プログラムの使用者の名前を指定できます。さらに、-ext オプションを指定することによって、作成するライセンス・ファイルに任意の項目を追加できます。追加項目の名前を name で、それに対する値を value で指定します。ただし、本ソフトウェアがライセンス条件として検査する項目は、次の項目だけです。

- ホスト名
- 物理アドレス
- IP アドレス
- 使用期限

追加した項目を独自のライセンス条件として利用したり、その他の目的で利用することも可能です。そのためには、配布するソフトウェアの中に独自のプログラムを書く必要があります。詳しくは「10．4 独自のライセンス条件を加える」をご覧下さい。

例）カレントディレクトリにライセンス・ファイルを作成します。

```
> jl -lic
[INFO] ライセンス・ファイルを作成しました license.properties
```

例）ライセンス・ファイルにソフトウェアの名前とバージョンを追加します。

```
> jl -lic -prod SimpleApp -ver 1.0.0
[INFO] ライセンス・ファイルを作成しました license.properties
```

９．５　サーブレットの実行

この手順は利用者側で行います。

(1)利用者は、受取ったライセンス・ファイルを代理サーブレットが読み込む場所に合わせて配備します。

(2)利用者は、サーブレットコンテナを起動します。

（３）　ブラウザーを使って代理サーブレットに対応付けた URL にアクセスすると、暗号化したサーブレットが起動します。

10. 応用

ひとくちに、「アプリケーションを配布する」と言っても様々な形体があります。『7. アプリケーションの開発から実行まで』から『9. サーブレットの開発から実行まで』で説明したのは、基本的な配布手順です。本ソフトウェアは、この基本手順を応用することによって、次のような配布形態に対応できます。

- **n** ソフトウェア・アーカイブに掲載する
- **n** 特定の利用者に対してだけ提供する
- **n** 使用上の制限をつけないで不特定多数の利用者に提供する

この章ではさらに標準のライセンス条件を拡張する方法についても説明します。

- **n** 独自のライセンス条件を加える

１０．１　ソフトウェア・アーカイブに掲載する

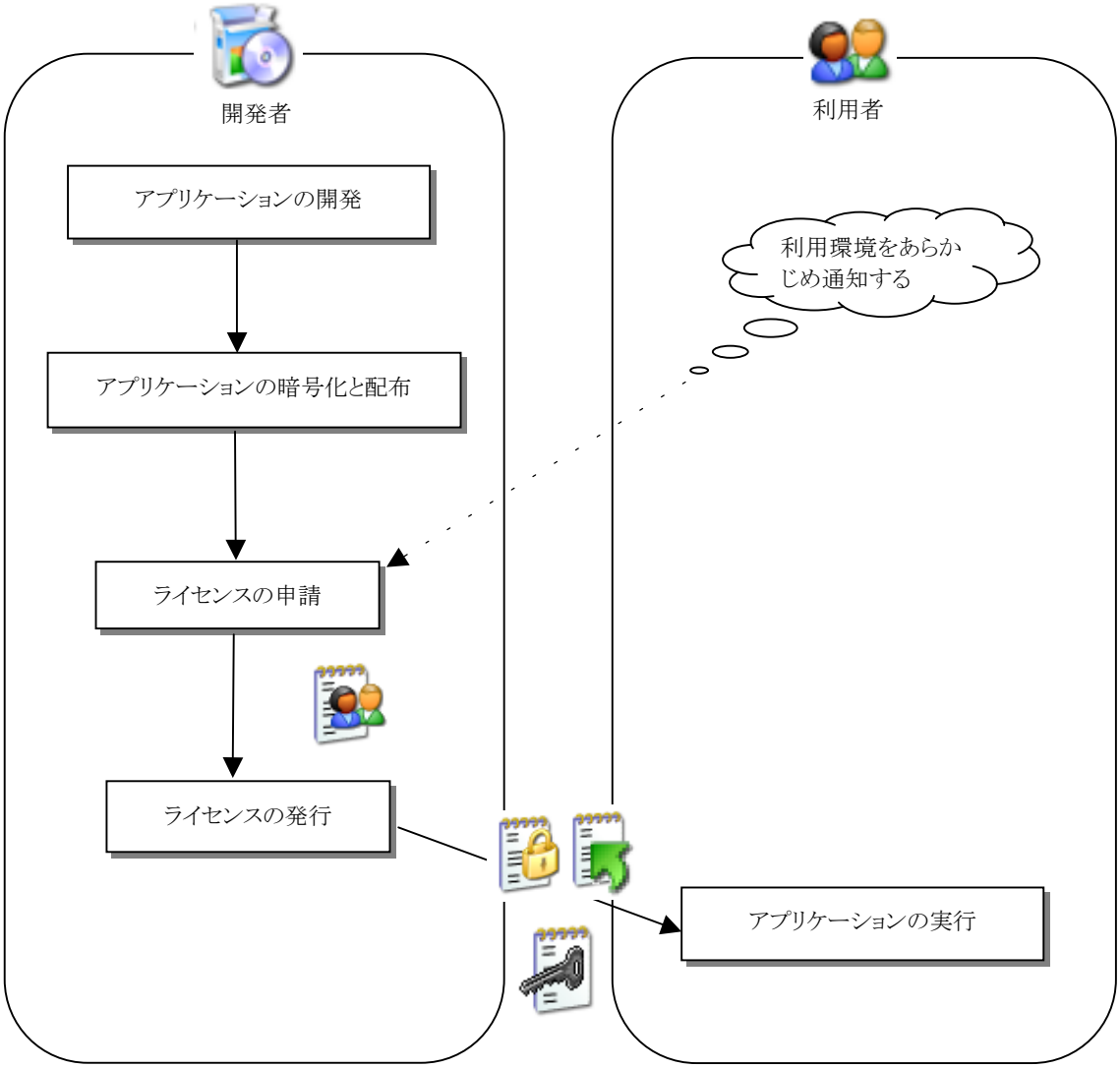
各社が提供するインターネット上のソフトウェア・アーカイブにあなたのアプリケーションを掲載する場合が典型的な例です。この配布形体は、『7. アプリケーションの開発から実行まで』で説明した、基本的な配布手順そのものです。

(1)　先ず利用者は、暗号化されたアプリケーション、ランチャー、暗号化パッケージをダウンロードします。
(2)　利用者は、ダウンロードしたランチャーを使ってライセンス申請ファイルを作り、アプリケーションの開発者宛に電子メール等で送付します。
(3)　開発者は、受けとったライセンス申請ファイルを元にライセンス・ファイルを作成し、利用者に電子メール等で返送します。

１０．２　特定の利用者に対してだけ提供する

この配布形体は、あらかじめ利用者の利用条件、例えばホスト名や物理アドレスなどが契約などによって定義されている場合です。この配布形体では、利用者によるライセンス申請の手続きを、開発者であるあなたが利用者に代わって実施することよって、配布物一式を一括して利用者に配布することができます。

(1)　ホスト名や物理アドレスなど使用上の制限のために必要な情報を、あらかじめ利用者から電子メールや書面等で送ってもらいます。そして、開発者の環境下で『7．3　ライセンスの申請』を実施します。
(2)　開発者は、作成したライセンス申請ファイルを元にライセンス・ファイルを作成します。
(3)　開発者は、暗号化されたアプリケーション、ランチャー、暗号化パッケージ、そしてライセンス・ファイルをまとめて利用者に配布します。




暗号化したアプリケーション


ライセンス申請ファイル


ランチャー（含暗号パッケージ）


ライセンス・ファイル

１０．３　使用上の制限をつけないで不特定多数の利用者に提供する

この配布形体は、利用条件に制限を加えないので、利用者のホスト名などをあらかじめ知る必要がありません。従って、ライセンス申請の手続きを開発者であるあなたが代行する事ができます。この結果、配布物一式を一括して利用者に配布することができます。

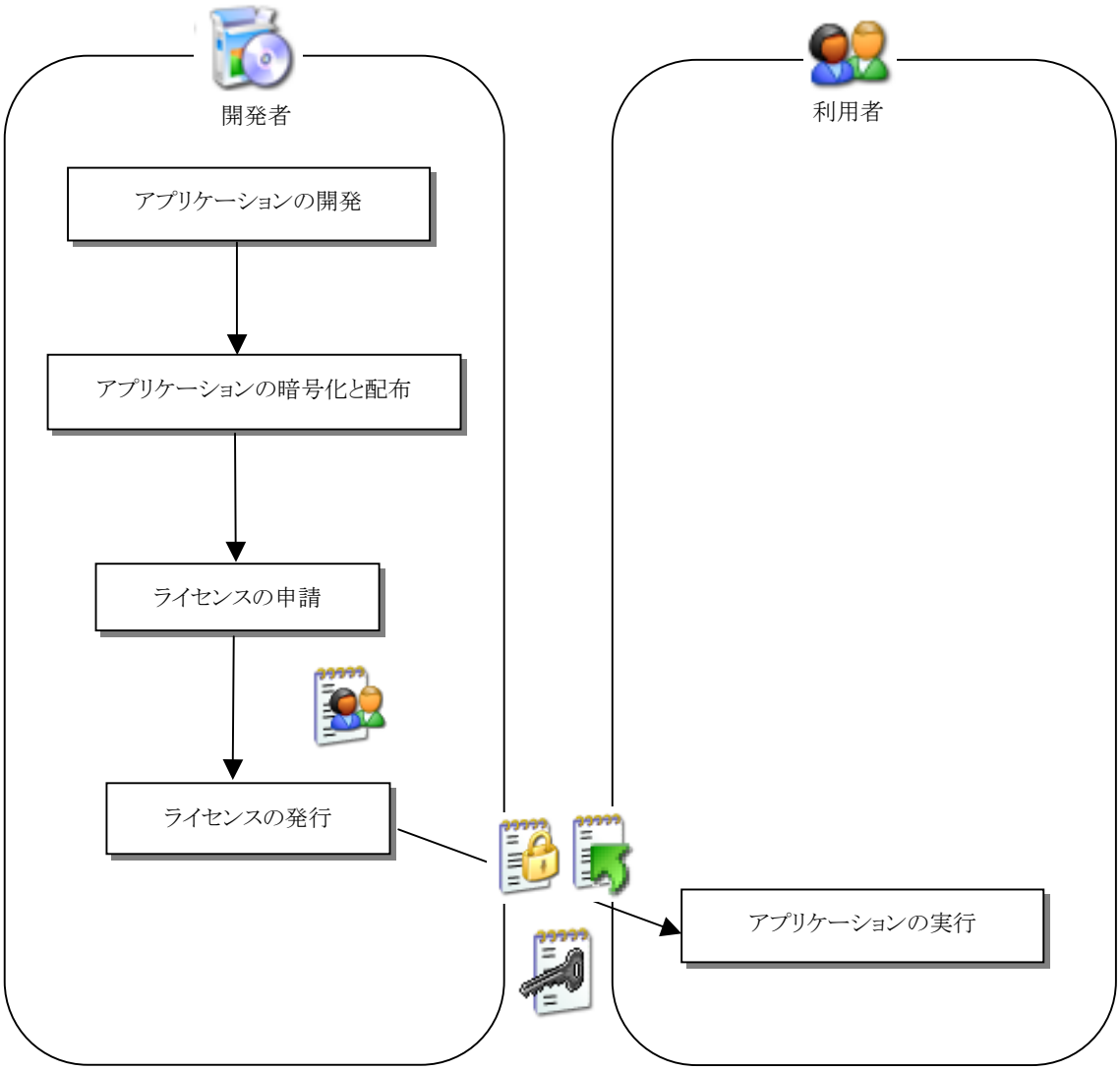
（1）　『7．3　ライセンスの申請』の手順を開発者の環境下で実施します。

（２）『7．4　ライセンスの発行』の手順で、ライセンス申請ファイル user.properties を次のように、「無制限（＊）」に設定します。

```
# 使用期限（制限しない場合は*）
license.expired=*
# 使用できるホスト名（制限しない場合は*）
license.hostname=*
# 使用できる IP アドレス（制限しない場合は*）
license.inetaddr=*
# 申請日
license.issued=07-Jun-2008
# 申請者
license.licensee=¥u5927¥u91D1¥u5EB7¥u592B
# 使用できる物理アドレス（制限しない場合は*）
license.macaddr=*
# 利用者鍵（変更してはなりません）
secure.userkey=305C300D06092A864886F70D0101010500034B003048024100B8AD09DC0C
643836C646DC5BFD1A8079B3A4362F969109391088CD19B0B1B605D9C133FA1EADD864227C8
C69CDD430A5DB720968ABD08A61D013F91EA4D0AF150203010001
```

（3）　開発者は、作成したライセンス申請ファイルを元にライセンス・ファイルを作成します。

（4）　開発者は、暗号化されたアプリケーション、ランチャー、暗号化パッケージ、そしてライセンス・ファイルをまとめて利用者に配布します。



暗号化したアプリケーション

ライセンス申請ファイル

ランチャー（含暗号パッケージ）

ライセンス・ファイル

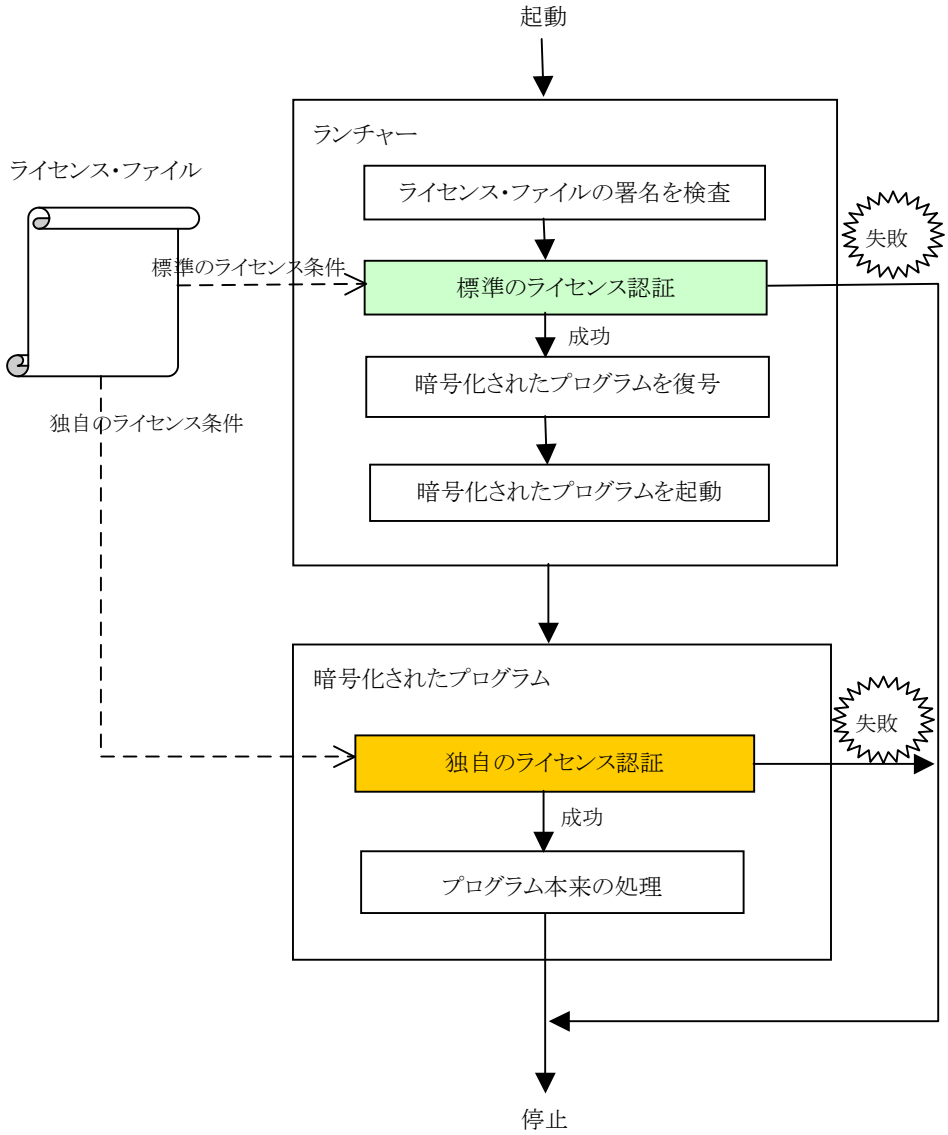
１０．４　独自のライセンス条件を加える

本ソフトウェアがライセンス条件として検査する項目は、次の項目だけです。

- ホスト名
- 物理アドレス
- IP アドレス
- 使用期限

これらの条件に加え独自のライセンス条件を加えることができます。ここでは、例としてアプリケーションの製品名とバージョンが一致することを条件に加える場合について説明します。

独自のライセンス条件を加えるためには、配布する暗号化されたプログラムの中に独自のライセンス認証の処理を書き加える必要があります。そして、この独自のライセンス認証の処理の結果が成功であった場合のみ本来の処理が行われるように書き直します。



独自のライセンス認証処理の書き方を説明します。独自のライセンス認証を行うためには、ライセンス・ファイルからライセンス条件を読み込む必要があります。そのために、com.ogane.jl.Launcher クラスが利用できます。先ず、Launcher クラスの getInstance メソッドでこのクラスのインスタンスを生成し、getLicenseProperty メソッドでライセンス条件を取得します。この Launcher クラスは、ランチャーつまり launcher.jar に含まれていますので、コンパイルするときにはクラスパスに launcher.jar を加える必要があります。Launcher クラスの詳細煮については「API 仕様書」をご覧下さい。

例） 配布するプログラム
製品名が SimpleApp、バージョンが 1.0.0 でライセンス

```
import java.io.File;
import com.ogane.jl.Launcher;

/**
 * アプリケーションの製品名とバージョンをライセンス条件に加える例。このクラス自体も暗号化される。
 */
public class SimpleApp
{
	public static void main(String[] args)
	{
		try
		{
			/* ライセンス・ファイルから製品名とバージョンを読み込む */
			Launcher  launcher = Launcher.getInstance(new File("license.properties"));
			String product = launcher.getLicenseProperty(Launcher.PRODUCT);
			String version = launcher.getLicenseProperty(Launcher.VERSION);

			/* 製品名とバージョンが一致するか検査する */
			if (product != null && product.equals("SimpleApp") &&
			version != null && version.equals("1.0.0"))
			{
				/* ライセンス認証が成功した場合の処理 */
				System.out.println("ようこそ、" +
				launcher.getLicenseProperty(Launcher.LICENSEE) + "さん");
			}
			else
			{
				/* ライセンス認証が失敗した場合の処理 */
				throw new Exception("この製品はライセンスされていません");
			}
		}
		catch(Exception ex)
		{
			ex.printStackTrace();
		}
	}
}
```

ライセンス・ファイルを発行するとき、コマンドの-prod オプションおよび-ver オプションを次のように指定する。

```
jl -lic -prod SimpleApp -ver 1.0.0
```

このように、製品名やバージョンなどよく使われるライセンス条件は、あらかじめ専用のコマンドオプションとして用意されています。しかし、これ以外の任意のライセンス項目を設定することもできます。この場合は、-ext オプションを指定します。例えば、試用版の起動回数を30回に制限するとしましょう。そして、このライセンス項目名を traial.count という名前にするとします。

配布するプログラム

```
String trial_count = launcher.getLicenseProperty("trial.count");
```

ライセンス発行コマンド

```
jl -lic -ext "trial.count=30"
```

この結果、配布するプログラムの trial_count には 30 が読み込まれます。何らかの方法で、プログラムの起動回数がカウントできれば、カウントした起動回数がこの起動制限回数以上に達したときにプログラムを起動できないようにすることができるでしょう。

11. 安全性

１１．１　暗号アルゴリズムの安全性

(1)　本ソフトウェアは次の暗号アルゴリズムを使用しています。これらの暗号は、今日の計算機技術では計算量的に安全である事が一般的に認知されています。

- 対称鍵暗号
  AES - DES に代わる次世代の暗号標準です。(既定)
  DESede - トリプル DES 暗号化です。(オプションとして利用可能だが非推奨)

- 非対称鍵暗号
  RSA - PKCS #1 に定義されている RSA 暗号化アルゴリズムです。

- 署名
  SHA1withDSA - このアルゴリズムは、NIST FIPS 186 に記述の署名アルゴリズムです。DSA は SHA-1 メッセージダイジェストアルゴリズムを使います。

(2)　本ソフトウェアは、これらの暗号の実装として下記の Java 対応暗号パッケージを使用しています。

- Java Cryptography Extension - Sun Microsystems Inc.
- Bouncy Castle Crypto APIs - Bouncy Castle

１１．２　本ソフトウェア自体の安全性

(1)　本ソフトウェア自体が、Java で作成されています。従って、本ソフトウェアを攻撃することによって、本ソフトウェアが保護しようとしているJavaプログラムの安全性を脅かす攻撃が存在します。本ソフトウェアは、これらの攻撃に対して、作業量的な安全性によって防御しています。
(2)　本ソフトウェアは、いくつかの典型的な Java 逆コンパイラを使った攻撃に耐えることを確かめています。
(3)　上記に関わらず、将来にわたって本ソフトウェアの安全性を保障するものではありません。

12. 使用上の注意

１２．１　本ソフトウェアをインストールするマシン

本ソフトウェアは開発者が所有するマシンにインストールすること。暗号化した Java プログラムを実行する利用者のマシンにはインストールしません。開発者が複数のマシンを所有し、Java プログラムの開発環境があるマシンと本ソフトウェアをインストールするマシンが異なってもかまいません。ただし、この場合は、開発環境マシンで作成した暗号化されていない*.jar や*.class ファイルを、本ソフトウェアをインストールしたマシンへコピーなどの方法で持ち込む必要があります。

１２．２　本ソフトウェアのバージョンアップ

本ソフトウェアを新しいバージョンに更新する場合、古いバージョンをアンインストールしてから新しいバージョンをインストールします。この時、環境変数 JL_HOME も新しいバージョンのインストール先に合わせて変更することも忘れずに行うこと。

１２．３　ライセンス・ファイルを作成するマシン

ライセンス・ファイルを作成するマシンは、必ず本ソフトウェアをインストールしたマシンでなければなりません。そして jl コマンドを使ってライセンス・ファイルを作成すること。

１２．４　配布するランチャー

暗号化した Java プログラムとともに利用者に配布するランチャーは、必ず本ソフトウェアのインストール先の lib サブディレクトリにあるランチャーであること。