ILOG JViews License Manager V2.0

License Management through JLM

Copyright

Copyright notice

Trademarks

IBM, the IBM logo, ibm.com, WebSphere, ILOG, the ILOG design, and CPLEX are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at *http://www.ibm.com/legal/copytrade.shtml*

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notices

For further copyright information see ***<installdir>*** **`/license/notices.txt`**.

Table of contents

Using license keys

Describes licenses and keys and, in particular, how to manage and deploy the keys you obtain to enable you to develop your applications in accordance with your licensing agreements.

In this section

About keys
Describes the types of key and explains how to use them with examples.

Managing keys
Explains where and how to install your keys and how to change the `keys.jlm` file.

Deploying an application
Describes how to use the keys you obtain to allow you to deploy your applications in accordance with your licensing agreements.

Troubleshooting
Gives a list of JLM error messages and some hints.

About keys

Describes the types of key and explains how to use them with examples.

In this section

Key types
Describes the license Web site and the types of key you can obtain from it.

The keys.jlm file
Describes the contents of the keys.jlm file and how it is used.

Key fields
Describes the format of a key and gives the meaning of each field.

Additional information about keys
Explains some important properties and rules relating to keys.

Example scenarios
Gives some example scenarios describing the types of key that would be used in the most common types of project.

Key types

The keys can be obtained from IBM®.

Remember, however, that keys may not strictly implement the license agreement. Always refer to your license agreement and the associated license certificate for a complete understanding of your rights. The following table shows the key types

Key types

Key type	Description
EVAL	Allows you to evaluate a product for a few days or weeks on any machine you choose. EVAL keys cannot be used with the JLM deployer, see *About deployment and the deployment tool*.
NODE	Allows you to develop on a declared machine. One key allows one developer to use and develop with one product. It does not allow the deployment of the developed applications.
SITE	Allows you to use and develop with the licensed products on one or more development machines at a single site. It does not allow the deployment of the developed applications **Important***: Even with a SITE key, your agreement may restrict the usage of the licensed products to a limited number of machines. Refer to your license agreement for details.*
RUNTIME	Allows you to deploy applications that use the licensed products so that your final users can run the applications. Although the RUNTIME key does not indicate the number of users for the final application, your agreement may limit this number. Refer to your license agreement for details.

The keys.jlm file

The `keys.jlm` file is a text file containing the keys you need to evaluate the licensed products and to develop and deploy the applications that use them. The initial version of the file is supplied when you first order the Java-based products.

You install only one copy of the `keys.jlm` file on your system and you can edit it if, for example, you need to add more keys.

When you deploy an application, a deployment tool, `jlmdeploy`, copies the required keys from the `keys.jlm` file into the deployed application.

The `keys.jlm` file contains the file name, the license holder name license key and a list of keys, as shown in the example:

```
# keys.jlm
#
LICENSE My-Company
NODE JViews-Diagrammer 6.000 31-May-2010 JGD6F3NGWJ27 2ac1445e applicstion:
myapp
RUNTIME JViews-Diagrammer 6.000 NEVER SR941350WWS2 N application: myapp
```

Key fields

The keys are of the general form:

```
KEYTYPE ILOG-ProductNameAndVersion ExpiryDate KEYCHECKSUM JHostID FLAG,
(application: yourapp OR options:)
```

Note: Not all of these fields will be present in a particular key.

For example:

```
NODE ILOG-Product 6.000 31-May-2008 JGD6F3NGWJ27 2ac1445e, application: yourapp
```

The following table gives the meaning of the various fields. Further information about the significance of these fields is given in *Additional information about keys*.

Description of key fields

Key Field	Description
`KEYTYPE`	The *Key types*, that is, EVAL, NODE, SITE, or RUNTIME.
`ILOG -ProductNameAndVersion`	The product name and version.
`ExpiryDate`	Depending on the licensing contract, keys may have an expiry date.
`KEYCHECKSUM`	Keys are protected by a checksum and editing any of their fields will invalidate the key.
`JHostID`	For NODE keys only. NODE keys are bound to a single machine identified by means of the `jHostID`. To obtain your `jHostID`s you have to run the `jhostid` utility.
`FLAG`	For SITE and RUNTIME keys only. Indicates whether the product displays licensee messages.
`application: yourapp`	NODE and SITE keys optionally contain the name of the application that you are developing. If no application name is declared, the key is valid for an unlimited number of applications. RUNTIME keys must contain the name of your application. You cannot deliver an application if you do not have a valid development key and its associated RUNTIME key.
`options:`	For example, a maintenance termination date.

Additional information about keys

Number of keys required per product and application

A key applies to a single product. Applications using several products use several keys.

Key expiry

Depending on the licensing contract, keys may have an expiry date. This is always the case for EVAL keys, which expire at the end of the trial period. Some keys may also include a maintenance termination date. Past this date you will not be able to upgrade to newer versions of products nor use patches built after this date, although you will still be able to use the older versions. Provided that you renew your maintenance contract before the maintenance termination date, you will be allocated new keys bearing new maintenance termination dates.

Keys bound to applications

NODE and SITE keys may or may not be bound to an application. If they are, the application name is declared in the key.

RUNTIME keys are always bound to an application and can only be used for the deployment of the declared application.

RUNTIME keys

A RUNTIME key is necessary to deploy, that is, distribute, an application developed by means of a development key (either NODE or SITE). For every IBM® ILOG® JViews product used in your development, you must have an associated RUNTIME key. Furthermore, the RUNTIME key is necessarily bound to a given application name.

Example scenarios

Evaluation

Typical EVAL keys for JViews Diagrammer and JViews TGO products:

```
-------------------------------------------------------------
LICENSE ILOG JViews Evaluation License Key
EVAL JViews-Diagrammer 6.000 29-May-2004 KD1G21H6SPNG
EVAL JTGO 4.000 29-May-2004 NA2HG01YGSB0
-------------------------------------------------------------
```

Development only

A JViews Charts and JViews Gantt development on a single machine having a JHostID: aaa04e9e.

```
-------------------------------------------------------------
LICENSE Coolsoft, Inc
SITE JViews-Charts 6.000 31-Dec-2004 2Z70I2JE86CG Y aaa04e9e
SITE JViews-Gantt 6.000 31-Dec-2004 6D20823ACCCS Y aaa04e9e
-------------------------------------------------------------
```

Application development

In this case development and runtime is bound to an application called IceMelting.

```
-------------------------------------------------------------
LICENSE Coolsoft, Inc
NODE JTGO 4.000 31-Jul-2004 5B4JC3Y8CXDX aaa04e9e , application: IceMelting
RUNTIME JTGO 4.000 31-Jul-2004 9X7JZ2H2S8BN Y , application: IceMelting
-------------------------------------------------------------
```

Generic development

This is a development project for JViews Charts. The development key applies to any application and expires after a one year subscription. The runtimes are bound to two distinct applications.

```
-------------------------------------------------------------
LICENSE Coolsoft, Inc
NODE JViews-Charts 6.000 NEVER RU3GE0YM0K30 aaa04e9e , options:
MaintenanceEnd=20050630
RUNTIME JViews-Charts 6.000 NEVER ADVCJ1YY4FM2 Y , application: IceMelting
```

```
RUNTIME JViews-Charts 6.000 NEVER U484S3WECBM2 Y , application: SunReflector
-----------------------------------------------------------------
```

Managing keys

Explains where and how to install your keys and how to change the `keys.jlm` file.

In this section

Where to install the keys
Gives some guidance on where to install the `keys.jlm` file depending on your platform and other requirements.

Installing your keys
Explains how to install your keys.

Viewing your keys
Describes how you can view your license keys file.

Making changes to the keys.jlm file
Describes how to make changes to your keys.jlm file after installation.

Where to install the keys

On Microsoft® Windows® systems, the products are often installed in `c:\Program Files\IBM\ILOG\jviews-framework86\` and so on. The location of `keys.jlm` is then `c:\Program Files\IBM\ILOG\jlm\keys.jlm`.

On **UNIX®** systems, the products are often installed in `$HOME/ibm/ilog/jviews-framework86/` and so on. The location of `keys.jlm` is then `$HOME/ibm/ilog/jlm/keys.jlm` .

More generally, the licensed products look for the `keys.jlm` file anywhere in the CLASSPATH. This means that the file can be located in a directory that is an element of the CLASSPATH or can be a top-level element in a JAR file that is an element of the CLASSPATH. You can do either of the following:

- Set the CLASSPATH environment variable
- Use the option `-classpath` when launching your JVM™

For more information on the use of classpath, consult the Java™ documentation.

In Eclipse™ plug-in applications, the keys file is looked up at the location specified by the user in the preference dialog **Window > Preferences > JLM**.

'Eclipse' is a trademark of Eclipse Foundation, Inc.

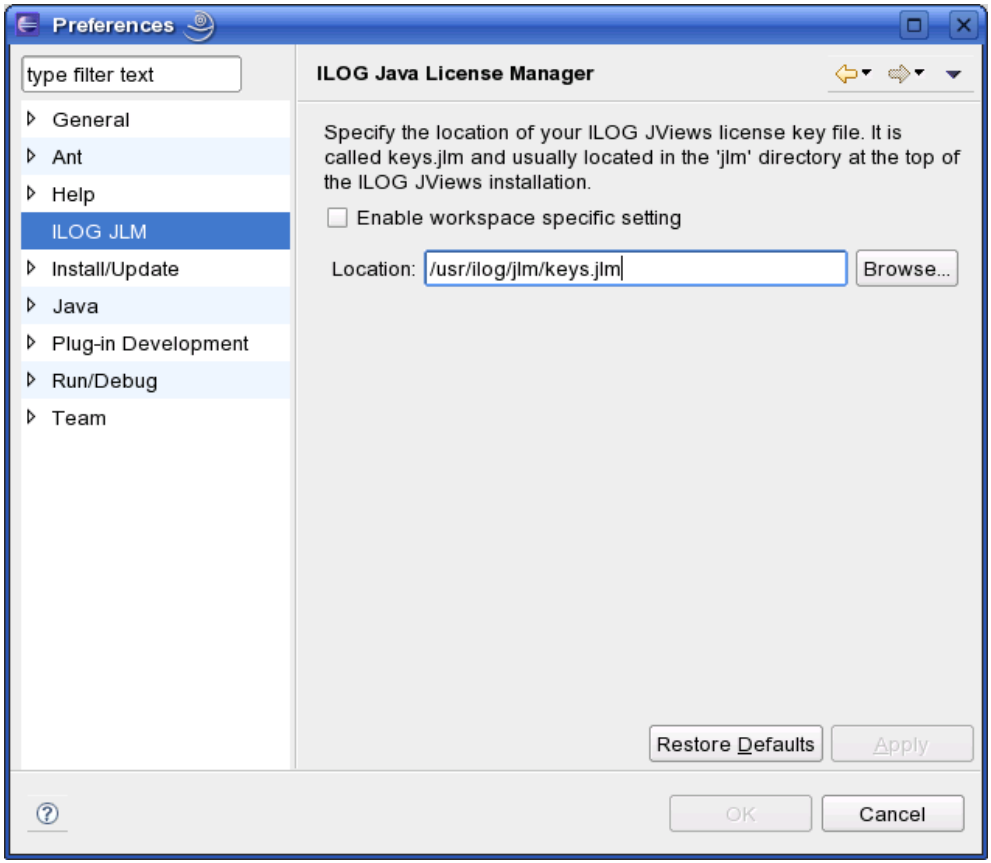
Various environments

If you are working with IBM® ILOG® JViews samples, you can store the `keys.jlm` file in the installation directory. If you are working in other environments, you may need to copy `keys.jlm`. Generally, it is recommended to minimize the number of copies of `keys.jlm`. If you copy the `keys.jlm`, you have to update all the copies when you receive a new development or runtime key. Moreover, it is easier to debug possible license problems when only one `keys.jlm` exists.

When you create a **command-line Java application**, the `keys.jlm` should be in a directory on the classpath. In the simplest case, you append the `ibm/ilog/jlm` directory of the product installation to the classpath.

When you work with the **Eclipse IDE**, you might want to create an Eclipse project that contains only a single resource directory, namely a link to the `ibm/ilog/jlm` directory of the product installation. Adding this project to the dependencies of your other IBM® ILOG® JViews projects will have the effect of extending the classpath so that `keys.jlm` is found.

When you create an **Eclipse plug-in application**, the `keys.jlm` file can be anywhere, but its location has to be specified by the user, using the preference dialog **Window > Preferences > JLM**.



By default, the setting is stored in the Eclipse installation, so it applies to all uses of the same Eclipse installation. The setting can also be specified on a per-workspace basis, making it possible to use different keys files in different workspaces.

When you build a **servlet application**, the `keys.jlm` file should be copied into the `WEB-INF/classes` directory of the servlet. Some servlet containers also provide a directory for classes and resources common to all servlets. The `keys.jlm` filr can be stored there as well. For example, Apache™ Tomcat™ 4 and 5 have a directory `CATALINA_BASE/shared/classes`; you can put `keys.jlm` there.

When you build a BEA Weblogic® application, the `keys.jlm` file should be copied into the `APP-INF/classes` directory of the application.

When you develop an **applet**, you can add the `keys.jlm` to the main classes directory or to the main JAR file, as long as the applet is not accessible for the entire Internet. (You are not allowed to distribute your license keys publicly.) Before deploying an applet for wide access, you need to remove `keys.jlm` from it and deploy it with `jlmdeploy`. For more information, see *Deploying an application*.

Installing your keys

Only one copy of this file should be installed on any machine and you can edit it later, for example, if you need to add more keys. Installation is normally performed automatically by the product installer and the following details are for information only.

Before you can use an IBM® ILOG® product, you have to install the `keys.jlm` file that you received from the commercial support organization.

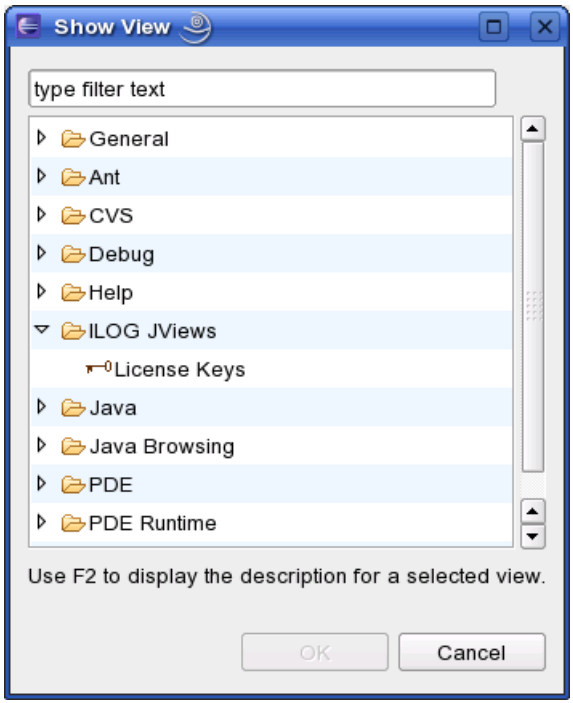
To install the `keys.jlm` file:

1. Create a `jlm` directory beneath the product directories.
2. Copy the `keys.jlm` file into the `jlm` directory.

Viewing your keys

You can view the license keys file as a text file in any text viewer.

In Eclipse™ plug-in applications, you can get a tabular view of the license keys by opening the license keys view from the **Window > Show View** menu item.



In the JViews Diagrammer, JViews Gantt, and JViews Charts products, you can also get a tabular view of the license keys through the **Help > Licenses** menu item.

Making changes to the keys.jlm file

When you have installed your `keys.jlm` file, you can:

- Add new keys or delete unwanted keys.
- Add or delete licenses at the top or bottom of the file. LICENSE is case sensitive.
- Add comments.

When you add new keys, make sure that each key line is on a single line. A line break would invalidate the key. Note also that each key is implicitly connected to the preceding LICENSE line. You should not copy key lines without their respective LICENSE line, nor remove LICENSE lines or move key lines under a different LICENSE line, otherwise the key becomes invalid.

Warning: The keys themselves are protected by a checksum and editing any of their fields will invalidate them.

You must restart your application(s) for the changes to take effect.

Deploying an application

Describes how to use the keys you obtain to allow you to deploy your applications in accordance with your licensing agreements.

In this section

About deployment and the deployment tool
Describes the purpose of deployment and the deployment tool supplied for JLM-licensed products.

Before you start deploying an application
Describes the tasks you need to do before you can deploy an application.

Using jlmdeploy from the command line
Describes how to invoke `jlmdeploy` and the various command line options that you can use to control the deployment of your application.

Using jlmdeploy as an ant task
Describes how to invoke `jlmdeploy` as an Apache™ `ant` task from a `build.xml` file.

Using the JlmDeploy GUI
Describes how to use the JlmDeploy GUI to deploy your application or examine the `deploy.txt` file; the GUI gives access to exactly the same functionality as the other launch methods.

JlmDeploy GUI description
Describes the elements in the JlmDeploy GUI.

Using JlmDeploy for Eclipse plug-ins
Describes how JlmDeploy operates on Eclipse™ plug-ins.

The deploy.txt file

Describes the content of the `deploy.txt` file and gives an example.

About deployment and the deployment tool

Once your application is developed and you are ready to distribute it to your final users, you have to deploy it. Without this stage, the application delivered to your final users will not work; without this stage, you would be distributing your license keys, which you are not allowed to do. Deployment consists mainly of integrating into your JAR file the keys needed to unlock your application. A tool is provided to help you do this.

The deployment tool, which is called `jlmdeploy`, copies the relevant RUNTIME license keys from the `keys.jlm` file to a file named `deploy.txt`, adds the file to the JAR file of the application, and saves the modified JAR file under the given name. The deployed application no longer looks for the `keys.jlm` file, but uses the RUNTIME license keys that have been added to the application through `jlmdeploy`.

Using the functionality in `jlmdeploy`, you can:

- Deploy your JAR.
- Consult the content of `deploy.txt` from a deployed JAR.
- Obtain online Help.
- Consult "About `jlmdeploy`" for version information.

The `jlmdeploy` tool can be launched in three ways:

- As a GUI application.
- As a command line tool.
- As an Apache™ `ant` task.

Each interface offers the same functionality. Every feature that is accessible in one mode is also accessible in the others.

The `jlmdeploy` tool can also be launched from the Eclipse™ IDE, through a popup menu on plug-in projects. In this context, `jlmdeploy` can add a `deploy.txt` to the selected plug-in project or projects, rather than to a JAR file.

Before you start deploying an application

Determine the JAR Files to be deployed

The JAR files that make direct API calls to IBM® ILOG® products need to be deployed. There can be a single JAR file or multiple JAR files. The JAR files of IBM® ILOG® products do not need to be deployed. If you repackage the product JAR files by unpacking them and rejarring them with your application classes, you will need to deploy the repackaged JAR file as well.

If you are going to deploy your application as a Java™ Web Start application without full permissions (also called sandboxed Java Web Start application) only one JAR file can be deployed. If there are several JAR files that make direct API calls to IBM® ILOG® products, you first need to combine them into a single JAR file.

Declare the application name in the application

The deployed application has to have a declared name. This name is the one that you supplied to obtain your RUNTIME keys. Even if you possess one or several development keys not specially bound to a particular application, it is necessary to give a name to the application you want to deploy. This name will be bound to the RUNTIME key used for the deployment. The application name has to be declared through a function call `IlvProductUtil` like this:

```
IlvProductUtil.registerApplication("application name");
```

The call to `registerApplication(java.lang.String)` needs to be in the JAR that is being deployed or in one of the JAR files, if there are several being deployed.

This call must be executed before any service of the product is used; otherwise, the license check may fail. It can be executed in a constructor that is executed once only, at the startup of the application, or in static initializers. Since static initializers can lead to problems that are difficult to debug, the former way of calling `registerApplication` is preferred.

Declare the application name in an Eclipse/RCP application

Inside an Eclipse™ or RCP application, the JViews classloader needs to be able to find the class that contains the `registerApplication` invocation. For this reason, the `registerApplication` call needs to be preceded by a call to `IlvClassLoaderUtil.registerClassLoader` that specifies the classloader of the class currently being executed.

Usually, this code is located in the activator class of a plug-in. For example:

```
public class MyPluginActivator extends AbstractUIPlugin {
    /**
     * This method is called upon plug-in activation
     */
    public void start(BundleContext context) throws Exception {
      super.start(context);
      IlvClassLoaderUtil.registerClassLoader(getClass().getClassLoader());
      IlvProductUtil.registerApplication("application name");
```

```
    }

    /**
     * This method is called when the plug-in is stopped
     */
    public void stop(BundleContext context) throws Exception {
      super.stop(context);
      IlvClassLoaderUtil.unregisterClassLoader(getClass().getClassLoader());
    }

}
```

The method `stop()` is overridden to undo the side effects of the plug-in activation.

Check you have the relevant keys

For everyIBM® ILOG® JViews product used by your application, you have to possess *two* keys: a development key and its associated RUNTIME key.

Check that keys.jlm is accessible

Check that the `keys.jlm` file is accessible from one of the `CLASSPATH` directories.

Note: When the `jlmdeploy` tool is run from the command line or as a GUI application, it looks in the `jlm/` directory in the installation directory, where the `keys.jlm` file is normally located, only after a search of the `CLASSPATH` that is provided as an environment variable.

Complete all development tasks

If you perform some other processing on your JAR files, such as obfuscation or removal of unused methods, these steps must be performed *before* you run `jlmdeploy`. In other words, `jlmdeploy` is the last step before shipping a JAR.

The `jlmdeploy` tool must be the last tool in the deployment chain, except for `jarsigner`.

- The `jarsigner` tool can be run before or after `jlmdeploy` when producing signed applets.
- The `jarsigner` tool must be run after `jlmdeploy` when producing signed Java™ Webstart applications.

If you intend to deploy your JAR using the Pack200 utility of JDK™ 1.5, you will need to normalize the JAR *before* running `jlmdeploy`. Normalizing a JAR relative to the Pack200 compressor can be done by using the `pack200 --repack` command.

Using jlmdeploy from the command line

Invoke jlmdeploy

To use `jlmdeploy` in command line mode you supply the `run.bat` command (on Microsoft® Windows® systems) or the `run.sh` command (on UNIX® systems), with one or more options. These command files are located in the `bin/jlmdeploy` directory. If no options are supplied, the GUI version is launched.

Invoke the functionalities

When you use the command line, you can use a number of different options. You should note that the different functionalities, for example, deploying a JAR and viewing a `deploy.txt` file, are exclusive. Therefore, if you use options belonging to different functionalities in the same invocation, an error is signaled.

Short-form and long-form options

Options can have two forms:

- A long version with a full name and two leading `--` characters.
- A short form with only one `-` and the first letter of the option. For example, `-a` is the short form of `--application` option.

Required arguments

You must supply at least the name of the deployed application with the `--application` option and the JAR that contains your own code. Thus, the minimum required form to deploy an application is:

```
Windows-prompt> cd jlmdeploy
Windows-prompt> run.bat --application "myapp" MyApplication.jar
Unix-prompt> jlmdeploy/run.sh --application "myapp" MyApplication.jar
```

--output deployed.jar option

By default, the deployed JAR is named by changing the suffix of the initial JAR file from '`.jar`' to '`.deployed.jar`'. Thus, given the above command, the deployed JAR will be named `MyApplication.deployed.jar`. You can change the output name of the deployed JAR with the `--output` option followed by the new name you want to give. You cannot give the same name as the undeployed JAR file.

--product product_name version option

The normal behavior is to write the `deploy.txt` with any valid development and runtime key pair found in your `keys.jlm` file. You can control this more closely with the following options:

`--product product_name version` will check that `deploy.txt` contains a valid key for `product_name` and that its version is equal to or higher than the declared version. You can repeat the `--product` option for as many products as you want. For example:

```
--product JViews-Charts 6.5 --product JTGO 4.5
```

--no-warning product_name option

The `jlmdeploy` command warns you if the `keys.jlm` contains valid development keys but no associated RUNTIME keys. This is because if you deliver your deployed JAR to your customers without the RUNTIME keys the application will fail to launch. If you are not going to use some of the JViews products, you can specify the `--no-warning` option for each of those unused products. For example:

```
--no-warning JViews-Charts --no-warning JViews-Maps
```

--sandboxed-jnlp option

If you are going to deploy your application as a Java™ Web Start application without full permissions (also called sandboxed Java Web Start application) you need to pass the option `--sandboxed-jnlp` to `jlmdeploy`. This option will put additional information into the `deploy.txt` file, that is necessary for running in JNLP contexts without full permissions.

--warnings-errors option

If you want to ensure that your application JAR has been correctly deployed, you should pass `--warnings-errors` as a command line option. This can be useful with automatic building tools like `ant` or `make`. With this option, warnings are considered as errors and the JAR is not deployed. When there are errors, the `jlmdeploy` command returns an exit code equal to 1. By convention, a nonzero status indicates abnormal termination.

Miscellaneous options

The two options in this section must be supplied separately.

`--list` *app.deployed.jar* displays the content of `deploy.txt` file on the console.

`--help` displays a short description of the allowed syntax.

Using jlmdeploy as an ant task

Parameters

`jlmdeploy` can take the parameters in the following table as attributes.

For additional information about the options, see *Using jlmdeploy from the command line*.

ant tool attributes

Attribute	Description	Required
`application`	Name of the application, as mentioned in the license keys.	Yes
`input`	File name of the input JAR.	Yes
`output`	File name of the JAR to be created.	No (default is based on the input JAR file, ending in `.deployed.jar`)
`products`	A comma separated list of products and corresponding version numbers.	Yes
`nowarnings`	A comma separated list of products.	No (default is empty)
`warningserrors`	A Boolean value.	No (default is false)
`sandboxedJNLP`	A boolean value. It determines whether to include support for running as part of Java™ Web Start (JNLP) applications without full permissions.	No
`classpath`	The classpath to use. It must include the path to `jlm-tools.jar` and the directory containing the `keys.jlm` file.	No
`classpathref`	The classpath to use, given as a reference to a PATH defined elsewhere. It must include the path to `jlm-tools.jar` and the directory containing the `keys.jlm` file.	No
`fork`	A boolean value. It determines whether jlmdeploy is executed in a different Java VM.	No

It can also take the following parameters as nested elements:

ant tool nested elements

Nested Element	Description	Required
`classpath`	A PATH like structure that can also be set through the `classpath` attribute.	No

Ant syntax

Here is an example in `ant` syntax, suitable for a `build.xml`.

Note: The classpath in the `taskdef` is needed so that this ant task can be found. The classpath in the `jlmdeploy` element is needed so that `jlmdeploy` can find the `keys.jlm` file.

```
<taskdef name="jlmdeploy"
     classpath="${jviews.framework}/lib/jlm-tools.jar"
     classname="ilog.tools.ant.JlmDeploy" />
<jlmdeploy products="JViews-Diagrammer 6.0"
     application="MaxiMap"
     input="myapp.jar"
     output="myapp.deployed.jar">
  <classpath>
     <pathelement location="${jviews.framework}/lib/jlm-tools.jar" />
     <pathelement location="${jlmdir}/" />
  </classpath>
</jlmdeploy>
```

Ant project example

See also **`<installdir>/jviews-framework86/codefragments/deployment`**, an example ant project, which contains a ready-to-run `build.xml` for a JViews Diagrammer sample.

Using the JlmDeploy GUI

The following are all valid ways to launch the JlmDeploy GUI:

- From the `bin/jlmdeploy` directory:
  - On Microsoft® Windows® systems run `run.bat`.
  - On UNIX® systems run `run.sh`.

For a description of the JlmDeploy GUI and the display modes, see *JlmDeploy GUI description*.

To deploy your application by adding a deploy.txt file to a JAR file:

1. On the **Deploy** menu, click **Deploy**.

   A file chooser opens.

   Note: You can cancel the procedure at any time and return to the empty JlmDeploy GUI by clicking **Cancel** in the JlmDeploy GUI. The **Status** bar displays the message “No file chosen”.

2. In the file chooser, select your Input JAR file. `jlmdeploy` checks whether the JAR already has a `deploy.txt`:
   - If yes, an error dialog appears.
   - If no, the **Input JAR file** box displays the name of the file you selected.
3. In the **Application Name** box, type the application name.

   When the application field is validated, the Display Area lists the JViews products associated with the application. The list is organized as follows:
   - If development is possible for a product but it does not have RUNTIME license keys matching the given application name, it is grayed and put at the bottom of the list.
   - If deployment is possible for a product, it is not grayed and has an editable check box. The initial state of the check box is determined by the last run of `jlmdeploy` for this application name; if a product was not checkable in the last run, the initial state of the check box depends on whether the JAR appears to be using the product.
4. Select the products you want to include in the deployment by clicking the check boxes.
5. Select "Support running as a sandboxed Java Web Start (JNLP) application" if your application will run as a Java™ Web Start application without full permissions.
6. In the **Output jar file** box, do one of the following:
   - Accept the default destination JAR filename.
   - Type a new name.
   - Click **Browse** to open a file chooser and choose a file.

Note: You can cancel the browse procedure at any time and return to default file choice by clicking **Cancel** in the JlmDeploy GUI.

7. To generate the destination JAR, click **OK**.
   - Any warnings are shown in a dialog box with **Continue** and **Cancel** options. If you choose to cancel the **Status** bar displays an "operation canceled" message.
   - If the operation is successful, the **Status** bar displays a "success" message.

To display the contents of a deploy.txt file:

1. If you have not already done so, launch the JlmDeploy GUI.
2. On the **Info** menu, click **List**. A file chooser opens.

Note: You can cancel this procedure at any time and return to the empty JlmDeploy GUI by clicking **Cancel** in the JlmDeploy GUI. The **Status** bar displays the message "No file chosen".

3. Choose a JAR name. The contents of the JAR's `deploy.txt` file is displayed.

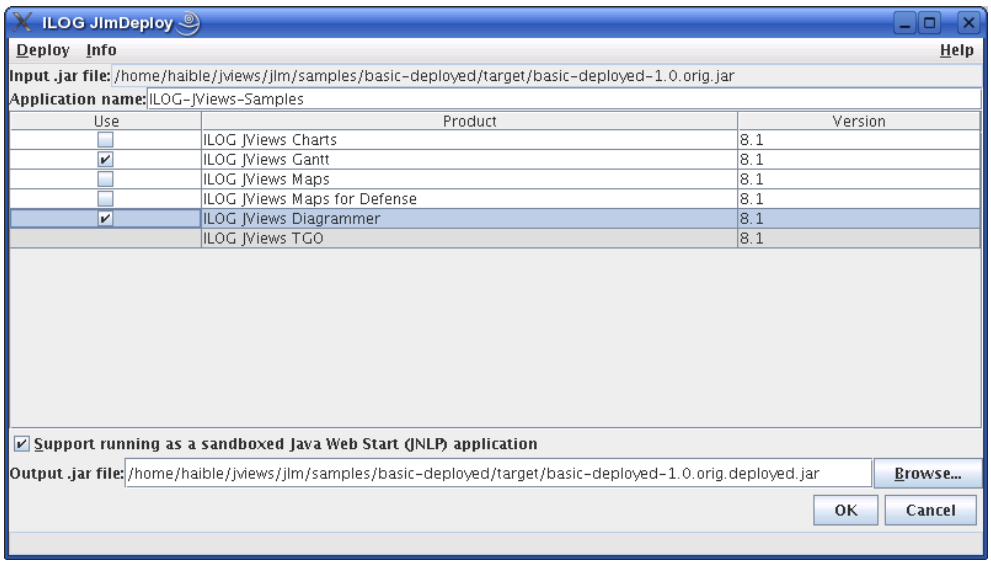
JlmDeploy GUI description

The display area

This is the scrollable area of the GUI. There is a display mode for deploying products and one for displaying the `deploy.txt` file.

Product list mode

In this mode the area displays the current list of IBM® ILOG® JViews products in preparation for deployment.



JlmDeploy GUI showing a product list

The area contains three columns:

- **Use**: for selecting products for deployment.
- **Product**: the product name.
- **Version**: the product version

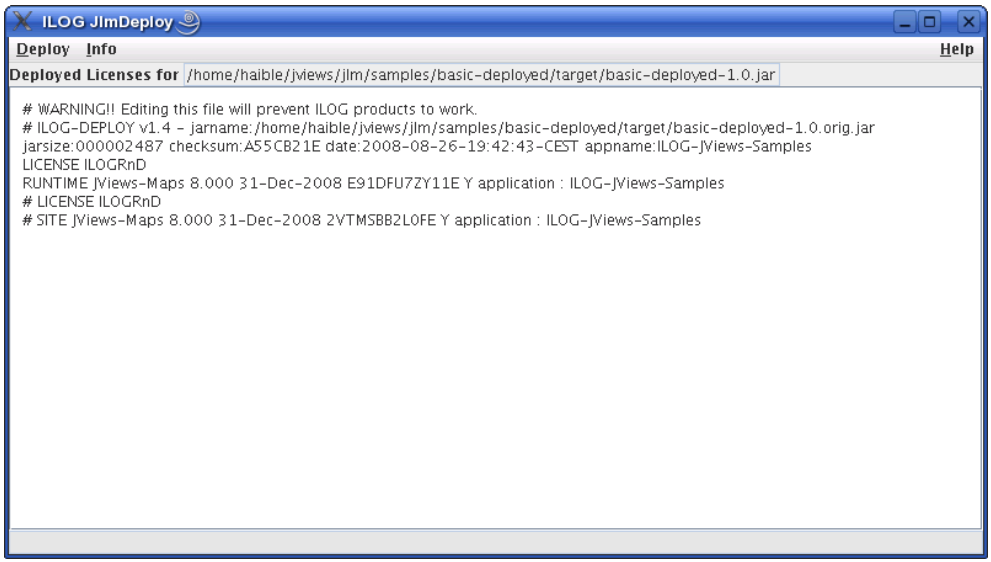
The product list is organized as follows:

- When you are deploying a JAR, if no application name has been retained from the last run, this area is initially empty. When the application field is validated, the scrollable area of the panel is filled with JViews products, one product per line. You can select a product by clicking the line.
- If development is possible for a product but it does not have RUNTIME license keys matching the given application name, it is grayed and put at the bottom of the list.

- If deployment is possible for a product, it has an editable check box. The initial state of the check box is determined by the last run of `jlmdeploy` for this application name; for those products which were not checkable in the last run, the initial state of the check box depends on whether the JAR appears to be using the product.

deploy.txt file mode

In this mode you can view the `deploy.txt` file.



JlmDeploy GUI showing a deploy.txt File

Menus

- **Deploy**. There is only one command on this menu. The **Deploy** command opens a file chooser so that you can select an Input JAR file.
- **Info**. There is only one command on this menu. The **List** command opens a file chooser so that you can choose a JAR and list the contents of its `deploy.txt` file.
- **Help**:
  - **Help** command: displays `jlmdeploy` commands and options.
  - **About** command: displays `jlmdeploy` version and copyright information.

Text boxes

- **Input jar file**. A noneditable text box containing the source JAR filename.
- **Application name**. An editable text box for the application name.
- **Output jar file**. Destination JAR filename.

Buttons

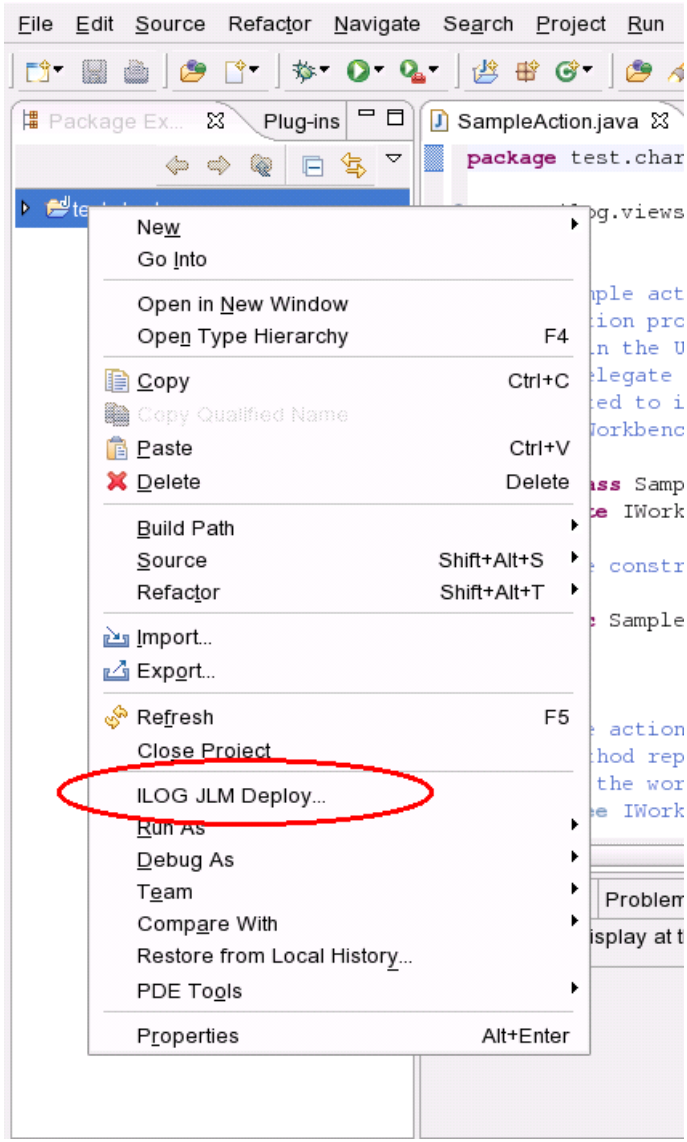
- **Browse** Opens a file chooser for the destination JAR filename.
- **OK**. Generates the destination JAR. Warnings are shown in a dialog box with **Continue** and **Cancel** options.
- **Cancel**. Click this button any time you use a file chooser to cancel choice of an input JAR file, an output JAR file, or a JAR containing a `deploy.txt` file, see *Using the JlmDeploy GUI*.

Using JlmDeploy for Eclipse plug-ins

JlmDeploy can also operate on Eclipse™ plug-ins instead of JAR files. In this mode, it adds a `deploy.txt` file to the sources of the plug-in.

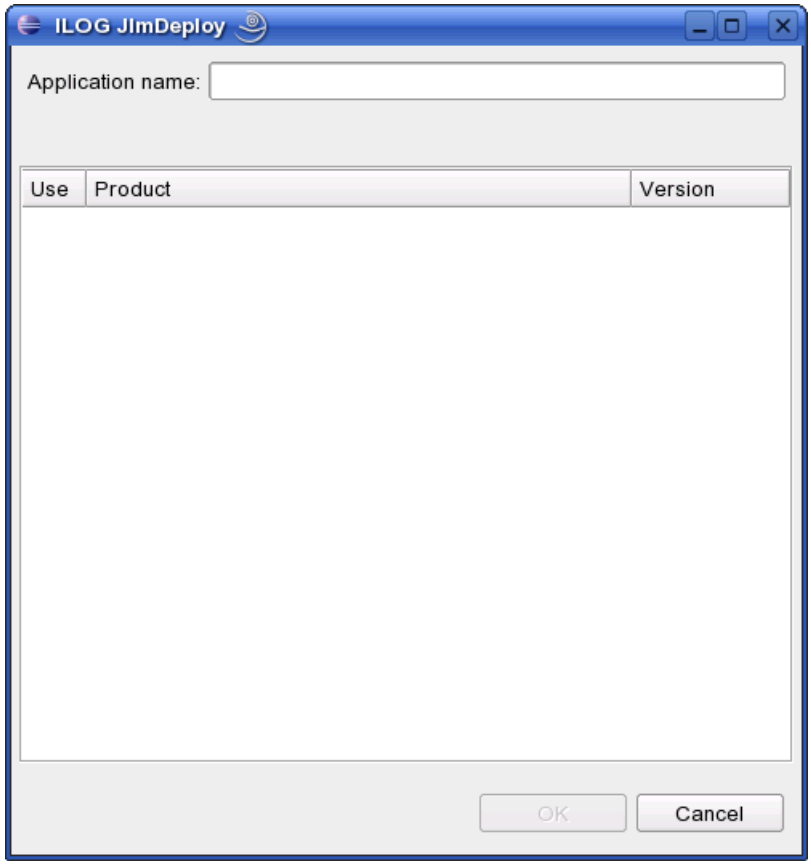
To launch JlmDeploy in this mode, perform these steps in the Eclipse IDE:

1. Select one or more Java™ plug-in projects in the **Package Explorer** view. This view is visible in the Java or Plug-in Development perspective.
2. Request the popup menu. You should see a menu entry **JLM Deploy**.

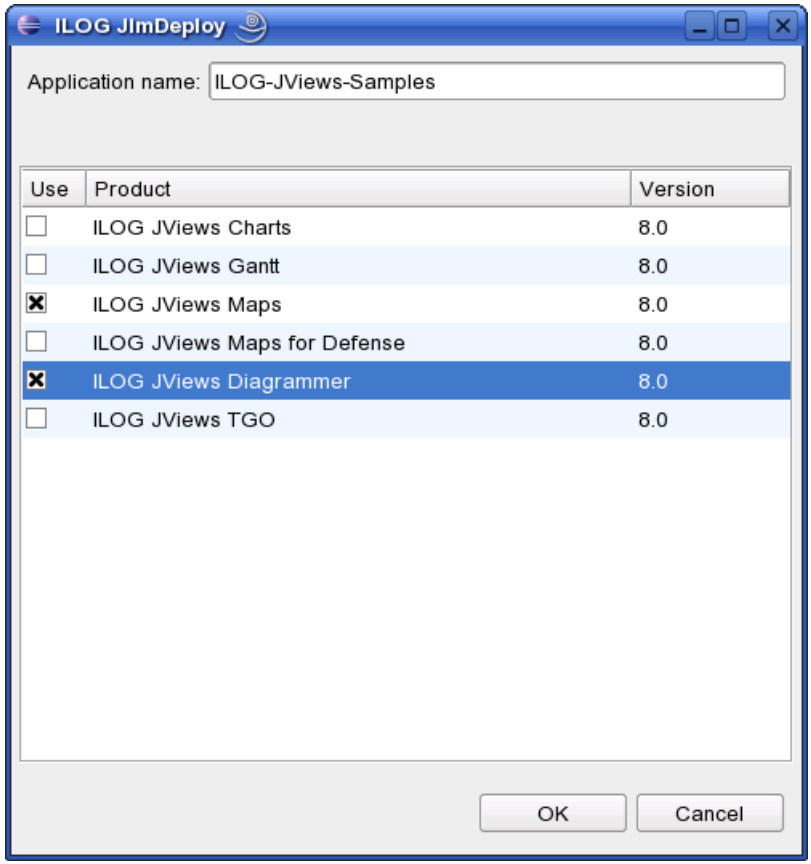


If this menu entry is not visible, verify in the **Help > About Eclipse SDK > Plug-in Details** view that the plug-in `ilog.views.eclipse.jlm.tools` is present. If it is not, you can install it using **Help > Software Updates** from the local update site at <installdir>/jviews-framework86/tools/ilog.views.eclipse.update.site.

3. Choose this menu entry. A dialog box prompts you to enter an application name.



4. Enter the application name. This name should be the same as the one to which the RUNTIME keys in your keys file are tied. The dialog box then presents a list of products for which the plug-in projects can be deployed.



5. Select the required product names and click OK. If no error message is shown, the deployment operation is successful and a `deploy.txt` file is added to each project.

Now you can deploy the projects as normal by using the Eclipse export wizard, available under **File > Export > Deployable plug-ins and fragments**.

The `deploy.txt` files contain information about the contents of the Java™ source files in the project. Therefore, if the contents of the plug-in projects have changed, it is necessary to repeat this deployment step before every use of the Eclipse export wizard.

The deploy.txt file

You cannot edit the `deploy.txt` file, but it contains a record of your deployment and is useful if you encounter problems deploying your applications.

Content

The `deploy.txt` file contains:

- For each product:
  - A copy of all valid runtime licenses found by `jlmdeploy` that apply to the application, including those of earlier versions.
  - In comments, a copy of one of the valid contractual development licenses that apply to the application. The hash code of this key is encrypted/obfuscated.
- The name of the original JAR used for deployment.
- The sum of all uncompressed sizes of the elements of the original JAR.
- The date and time of the deployment.

Example

A typical `deploy.txt` looks like this:

```
# WARNING!! Editing this file will prevent IBM ILOG products from working.
# ILOG-DEPLOY v1.0 - jarname:YouApp.jar jarsize:000001225 checksum:92DBB201
date:2004-03-17-19:06:36-CET appname:myapp
LICENSE My-Company
RUNTIME  JViews-Diagrammer 6.000 NEVER SR941350WWS2 N application: myapp
# LICENSE ILOG-Test
# SITE JViews-Diagrammer 6.000 31-May-2010 BH35ZW8QXQ7C Y
```

Privacy

The `deploy.txt` file contains the RUNTIME keys needed by your application and the development keys in a modified form: the original key password has been encrypted so that no one can reuse the keys. Consequently the file does not compromise your privacy.

Troubleshooting

Error messages

Error messages

Error	Meaning	Suggested Action
File '`keys.jlm`' not found.	This error occurs when `keys.jlm` is looked for, but not found in the CLASSPATH.	In development mode, add the directory or JAR containing `keys.jlm` to your CLASSPATH used for running the application. Note that for applets, the directories in the classpath elements have to be terminated with a slash. If your application is already deployed (and thus should not even look at the `keys.jlm` file any more), the deployment must have failed in some way.
The license has expired.	The license is a time-limited one and it has expired.	Contact your sales representative to renew the license.
Key found but maintenance expired.	The license has time-limited maintenance. You are attempting to use a release or patch of an IBM® ILOG® product that was issued after the maintenance period ended.	Either go back to a product version that was issued before your maintenance period ended or contact your sales representative to renew the maintenance contract.
Key found for an older version.	The license key is for a given product version or minor revisions of it. You are trying to use a new major version of the product with it.	Contact your sales representative to renew the license.
Key found for another machine.	The license key is bound to a given computer. You are trying to use the product on a different computer.	The `hostid` for which the license is valid is mentioned in the key. You can determine the `hostid` of your computer by executing the '`jhostid`' program. After checking that these two are really different, either use a computer that has a matching `hostid`, or contact your sales representative to order a new NODE license for your computer.
Key not found for this application name.	The license key is bound to an application name, but the application name registered through `registerApplication()` is different.	Check the spelling of the `registerApplication()` argument versus the application name found in the license key.
Your `keys.jlm` file has no valid key.	The `keys.jlm` file has only invalid keys, or no keys at all.	Check that the `keys.jlm` is at the place where you expect it to be (for example, by moving it away and seeing whether you get

Error	Meaning	Suggested Action
		a different error message). Then look at the `keys.jlm` file with a text editor. If it contains keys that are no longer usable, contact your sales representative to order new licenses.
Key found with a corrupted format. Or: Invalid character in a checksum.	The `keys.jlm` file has keys that do not match the expected syntax of JLM keys.	Check that the `keys.jlm` is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then either restore the file using an earlier, working backup or look at the `keys.jlm` file with a text editor and eliminate malformed keys from it.
Key found for another product.	You are trying to use a functionality that is not covered by the valid license keys.	Contact your sales representative to buy licenses for the added product that you are trying to use.
Key found for another key type.	The `keys.jlm` file contains a key whose type is not one of the following: EVAL, NODE, SITE, RUNTIME.	Check that the `keys.jlm` file is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then look at it with a text editor and eliminate malformed keys from it.
EVAL key found, but a key of type SITE or NODE is required.	Some operations, such as deployment, require a key of a particular type. While for development, EVAL, SITE and NODE keys give the same rights, for deployment, EVAL keys cannot be used.	Contact your sales representative to buy the required type of license key.
Can't access your key file.	The `keys.jlm` file present but not accessible.	Check that the `keys.jlm` file is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then check its access permissions and whether you can view it in a text editor.
Missing application name, initialization function was not called. Or: The function `IlvProductUtil.registerApplication` was not called.	In a deployed application, an IBM® ILOG® product is used without `IlvProductUtil.registerApplication` having been called before.	Add an `IlvProductUtil.registerApplication` call to the JAR and redeploy it. Make sure this call is executed early enough during the initialization of the application.
Key found with an expired date.	The validity period of a key is terminated.	Check that the `keys.jlm` file is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then look at it with a text editor and eliminate expired keys from it. If you then lack license keys, contact

Error	Meaning	Suggested Action
		your sales representative to buy new license keys.
The required 'SITE' name is missing in your keys file.	The first key in the `keys.jlm` file is not preceded by a LICENSE line that contains the site name.	Check that the `keys.jlm` file is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then look at it with a text editor and add the LICENSE line that was present when you were sent the license keys.
Wrong banner flag, please contact IBM® technical support.	The banner flag of a license is not among the valid values. You must have been sent incorrect keys.	Contact technical support, to get corrected license keys. When doing so, send a copy of the malformed `keys.jlm` file.
Key found with an invalid checksum code.	The checksum code of a key is not valid. The keys must have been corrupted since you received them.	Check that the `keys.jlm` is at the place where you expect it to be (for example, by moving it away and seeing whether you get a different error message). Then either restore the file using an earlier, working backup. Or contact technical support. When doing the latter, please send a copy of the malformed `keys.jlm` file.
Unknown license error ID#xyz, Please contact IBM® technical Support.	An internal malfunctioning of JLM has occurred.	Contact technical support. When doing so, send a copy of the malformed `keys.jlm` file.

Hints

Remove unwanted popup dialog boxes

If you are using the licensed products from within a web server, and do not want the server to display popup dialog boxes (for example, because the console of the web server is unattended or because some web servers do not exit cleanly after an AWT/Swing window has been shown), you can prevent JLM from displaying them by setting the system property `java.awt.headless=true`. If this is set, JLM will write any possible error messages to the log files of the web server, instead of displaying a popup dialog.

Index