

launch4j

3.0.2

[Home](#)
[Docs](#)
[Changelog](#)
[Screenshots](#)
[Download](#)
[Support](#)
[Project summary](#)
[Bug tracker](#)
[Links](#)

[Running launch4j](#)
[Configuration file](#)
[Importing 1.x configuration](#)
[Ant Task](#)
[Additional JVM options at runtime](#)
[Runtime options](#)
[Settings](#)

Running launch4j

Run *launch4j.exe* or *launch4j* script without command line arguments to enter the GUI mode.

```
launch4j.exe
```

To wrap a jar in console mode use *launch4jc.exe* and specify the configuration file.

```
launch4jc.exe config.xml
```

On Linux use the *launch4j* script.

```
launch4j ./demo/l4j/config.xml
```

Configuration file

Launch4j requires an xml configuration file for each output executable. You can create and edit it conveniently using the graphic user interface or your favorite editor. Alternatively it's possible to pass all of the configuration parameters through the Ant task. All files may be absolute paths or relative to the configuration file path.

```
<!-- Bold elements are required -->
<launch4jConfig>
    <headerType>gui|console</headerType>
    <outfile>file.exe</outfile>
    <jar>file</jar>
    <dontWrapJar>true|false</dontWrapJar>
    <errTitle>text</errTitle>
    <downloadUrl>http://java.com/download</downloadUrl>
    <supportUrl>url</supportUrl>
    <cmdLine>text</cmdLine>
    <chdir>path</chdir>
    <priority>normal|idle|high</priority>
    <customProcName>true|false</customProcName>
    <stayAlive>true|false</stayAlive>
    <icon>file</icon>
    <obj>header object file</obj>
    ...

```

```
<lib>w32api lib</lib>
...
<var>var=text</var>
...
<classPath>
  <mainClass>main class</mainClass>
  <cp>classpath</cp>
  ...
</classPath>
<singleInstance>
  <mutexName>text</mutexName>
  <windowTitle>text</windowTitle>
</singleInstance>
<jre>
  <!-- Specify one of the following or both -->
  <path>bundled JRE path</path>
  <minVersion>x.x.x[_xx]</minVersion>
  <maxVersion>x.x.x[_xx]</maxVersion>
  <jdkPreference>jreOnly|preferJre|preferJdk|jdkOnly</jdkPreference>
  <!-- Heap sizes in MB and % of free memory -->
  <initialHeapSize>MB</initialHeapSize>
  <initialHeapPercent>%</initialHeapPercent>
  <maxHeapSize>MB</maxHeapSize>
  <maxHeapPercent>%</maxHeapPercent>
  <opt>text</opt>
  ...
</jre>
<splash>
  <file>file</file>
  <waitForWindow>true|false</waitForWindow>
  <timeout>seconds [60]</timeout>
  <timeoutErr>true|false</timeoutErr>
</splash>
<versionInfo>
  <fileVersion>x.x.x.x</fileVersion>
  <txtFileVersion>text</txtFileVersion>
  <fileDescription>text</fileDescription>
  <copyright>text</copyright>
  <productVersion>x.x.x.x</productVersion>
  <txtProductVersion>text</txtProductVersion>
  <productName>text</productName>
  <companyName>text</companyName>
  <internalName>filename</internalName>
  <originalFilename>filename.exe</originalFilename>
</versionInfo>
<messages>
  <startupErr>text</startupErr>
  <bundledJreErr>text</bundledJreErr>
  <jreVersionErr>text</jreVersionErr>
  <launcherErr>text</launcherErr>
</messages>
</launch4jConfig>
```

```
<headerType>
```

Type of the header used to wrap the application.

Header type	Launcher	Splash screen	Wait for the application to close
gui	javaw	yes	wrapper waits only if <i>stayAlive</i> is set to true, otherwise it terminates immediately or after closing the splash screen.
console	java	no	always waits and returns application's exit code.

<outfile>

Output executable file.

<jar>

Optional, by default specifies the jar to wrap. To launch a jar without wrapping it enter the runtime path of the jar relative to the executable and set <*dontWrapJar*> to true. For example, if the executable launcher and the application jar named *calc.exe* and *calc.jar* are in the same directory then you would use <*jar*>*calc.jar*</*jar*> and <*dontWrapJar*>true</*dontWrapJar*>.

<*dontWrapJar*>

Optional, defaults to false. Launch4j by default wraps jars in native executables, you can prevent this by setting <*dontWrapJar*> to true. The exe acts then as a launcher and starts the application specified in <*jar*> or <*classPath*><*mainClass*>

<errTitle>

Optional, sets the title of the error message box that's displayed if Java cannot be found for instance. This usually should contain the name of your application. The console header prefixes error messages with this property (myapp: error...)

<cmdLine>

Optional, constant command line arguments.

<chdir>

Optional. Change current directory to an arbitrary path relative to the executable. If you omit this property or leave it blank it will have no effect. Setting it to . will change the current dir to the same directory as the executable... will change it to the parent directory, and so on.

```
<chdir>.</chdir>
```

```
<chdir>../somedir</chdir>
```

<customProcName>

Optional, defaults to false. Set the process name as the executable filename and use Xp style manifests (if any). Creates a temporary file in *launch4j-tmp* directory inside the used JRE. These files are deleted by any launch4j wrapped application, which sets the process name and

uses the same JRE. The removal takes place when the application **starts**, so at least one copy of this file will always be present.

`<stayAlive>`

Optional, defaults to false in GUI header, always true in console header. When enabled the launcher waits for the Java application to finish and returns it's exit code.

`<icon>`

Application icon in ICO format. May contain multiple color depths/resolutions.

`<obj>`

Optional, custom headers only. Ordered list of header object files.

`<lib>`

Optional, custom headers only. Ordered list of libraries used by header.

`<singleInstance>`

Optional, allow to run only a single instance of the application.

`<mutexName>`

Unique mutex name that will identify the application.

`<windowTitle>`

Optional, recognized by GUI header only. Title or title part of a window to bring up instead of running a new instance.

`<jre>`

Required element that groups JRE settings.

`<path>, <minVersion>, <maxVersion>`

The `<path>` property is used to specify the absolute or relative path (to the executable) of a bundled JRE, it does not rely on the current directory or `<chdir>`. Note that this path is not checked until the actual application execution. If you'd like the wrapper to search for a JRE (public or SDK private) use the `<minVersion>` property, you may also specify the `<maxVersion>` to prevent it from using higher Java versions. Launch4j will always use the highest version available (in the min/max range of course). If a Sun's JRE is not available or does not satisfy the search criteria, the search will be repeated on IBM runtimes. You can also combine these properties to change the startup process...

`<path>`

Run if bundled JRE and javaw.exe are present, otherwise stop with error.

`<path> + <minVersion> [+ <maxVersion>]`

Use bundled JRE first, if it cannot be located
search for Java, if that fails display error
message and open the Java download page.

<minVersion> [+ <maxVersion>]

Search for Java, if an appropriate version
cannot be found display error message and
open the Java download page.

<jdkPreference>

Optional, defaults to preferJre; Allows you to specify a
preference for a public JRE or a private JDK runtime.

Valid values are:

jreOnly

Always use a public JRE (equivalent to the
old option dontUsePrivateJres=true)

preferJre

Prefer a public JRE, but use a JDK private
runtime if it is newer than the public JRE
(equivalent to the old option
dontUsePrivateJres=false)

preferJdk

Prefer a JDK private runtime, but use a
public JRE if it is newer than the JDK

jdkOnly

Always use a private JDK runtime (fails if
there is no JDK installed)

HeapSize, HeapPercent

If size and percent are specified, then the setting which
yields more memory will be chosen at runtime. In other
words, setting both values means: percent of free
memory no less than size in MB.

<initialHeapSize>

Optional, initial heap size in MB.

<initialHeapPercent>

Optional, initial heap size in % of free
memory.

<maxHeapSize>

Optional, max heap size in MB.

<maxHeapPercent>

Optional, max heap size in % of free
memory.

<opt>

Optional, accepts everything you would normally pass to

java/javaw launcher: assertion options, system properties and X options. Here you can map environment and special variables *EXEDIR* (exe's runtime directory), *EXEFILE* (exe's runtime full file path) to system properties. All variable references must be surrounded with percentage signs and quoted.

```
<opt>-Dlaunch4j.exeDir="%EXEDIR%"</opt>
<opt>-Dlaunch4j.exeFile="%EXEFILE%"</opt>
<opt>-Denv.path="%Path%"</opt>
<opt>-Dsettings="%HomeDrive%%HomePath%\settings.ini"</
```

<splash>

Optional, groups the splash screen settings. Allowed only in GUI header.

<file>

Splash screen image in BMP format.

<waitForWindow>

Optional, defaults to true. Close the splash screen when an application window or Java error message box appears. If set to false, the splash screen will be closed on timeout.

<timeout>

Optional, defaults to 60. Number of seconds after which the splash screen must be closed. Splash timeout may cause an error depending on *<timeoutErr>*.

<timeoutErr>

Optional, defaults to true. True signals an error on splash timeout, false closes the splash screen quietly.

<versionInfo>

Optional, version information to be displayed by the Windows Explorer.

<fileVersion>

Version number 'x.x.x.x'

<txtFileVersion>

Free form file version, for example '1.20.RC1'.

<fileDescription>

File description presented to the user.

<copyright>

Legal copyright.

<productVersion>

Version number 'x.x.x.x'

<txtProductVersion>

Free form file version, for example '1.20.RC1'.

<productName>

Text.

<companyName>

Optional text.

<internalName>

Internal name without extension, original filename or module name for example.

<originalFilename>

Original name of the file without the path. Allows to determine whether a file has been renamed by a user.

Importing 1.x configuration

It's possible to import a 1.x configuration file using the GUI interface. Open the file, correct the paths and save it as a new xml configuration.

Ant task

You may set a launch4j directory property or change the task definition.

```
<property name="launch4j.dir" location="/opt/launch4j" />
```

Define the task in your Ant build script.

```
<taskdef name="launch4j"
  classname="net.sf.launch4j.ant.Launch4jTask"
  classpath="${launch4j.dir}/launch4j.jar
  :${launch4j.dir}/lib/xstream.jar" />
```

Execute the task!

```
<launch4j configFile=".//14j/demo.xml" />
```

You can set or override the following configuration properties...

jar="absolute path or relative to *basedir*"
jarPath="relative path"
outfile
fileVersion
txtFileVersion
productVersion
txtProductVersion
bindir="[alternate bin directory...](#)"
tmpdir="[alternate working directory...](#)"

```
<launch4j configFile=".//14j/demo.xml" outfile="mydemo.exe"
  fileVersion="1.0.0.0" txtFileVersion="1.0 RC2" />
```

You can also define the entire configuration in the task, but it will not be possible to edit such a file in the GUI mode. All paths except for `<chdir>`, `<jre><path>` and `jarPath` are calculated using the `basedir` project attribute.

```
<launch4j>
  <config headerType="gui" outfile="demo.exe"
    dontWrapJar="true" jarPath="demo.jar" >
    <var>SETTINGS="%HomeDrive%%HomePath%\\settings.ini"</var>
    <classPath mainClass="org.demo.DemoApp">
      <cp>./lib/looks.jar</cp>
      <cp>%USER_LIBS%/*.jar</cp>
    </classPath>
    <jre minVersion="1.4.0">
      <opt>-Dlaunch4j.exedir="%EXEDIR%"</opt>
      <opt>-Dlaunch4j.exefile="%EXEFILE%"</opt>
    </jre>
  </config>
</launch4j>
```

Additional JVM options at runtime

When you create a wrapper or launcher all configuration details are compiled into the executable and cannot be changed without recreating it or hacking with a resource editor. Launch4j 2.1.2 introduces a new feature that allows to pass additional JVM options at runtime from an .l4j.ini file. Now you can specify the options in the configuration file, ini file or in both, but you cannot override them. The ini file's name must correspond to the executable's (*myapp.exe : myapp.l4j.ini*). The arguments should be separated with spaces or new lines, environment variable expansion is supported, for example:

```
# Launch4j runtime config
-Dswing.aatext=true
-Dsomevar="%SOMEVAR%"
-Xms16m
```

Runtime options

--14j-debug

To make sure the output executable is configured correctly you can use the debug launching mode to log various information to the launch4j.log file.

--14j-default-proc

Use default process name.

--14j-dont-wait

Disable the "stay alive" function.

--14j-no-splash

Disable the splash screen.

--14j-no-splash-err

Disable splash screen error on timeout, might be useful on very slow computers.

Settings

Alternate bin directory: launch4j.bindir

It's possible to override the default bin directory location which contains windres and ld tools using the *launch4j.bindir* system property. The property can have two forms: a path relative to Launch4j's directory (*altbin* for example) or an absolute path.

Working directory: launch4j.tmpdir

Change the working directory if the default path contains spaces which windres cannot handle.

All trademarks mentioned are properties of their respective owners.

Copyright © 2005-2011 Grzegorz Kowal

