

HOWTO: Optimize and deploy your Resin applications with Excelsior JET

Article ID: 000020

Last Revised On: 22-Sep-2005

The information in this article applies to:

- Excelsior JET Professional Edition for Linux, versions 3.6 and 3.7
- Caucho Resin 3.0.X and above

This article **does not** apply to Excelsior JET 4.0 and above.

SUMMARY

This document contains step-by-step instructions for using Excelsior JET to compile the Caucho Resin application server along with your deployed Web applications. Deployment of the entire system to end users is also covered.

Project files and build/run scripts for this article may be found [here](#).

INTRODUCTION

Today's state-of-the-art server-side Java applications usually run on a so-called application server. The application server itself is also a Java program and thus can be converted to an executable using Excelsior JET.

A typical application server, such as Caucho Resin, loads applications for further execution through server-specific classloading that cannot be defined before the actual server execution. As a result, deployed applications can not be precompiled together with the server, but have to be dynamically compiled during the server run. The Caching JIT compiler, available only in Excelsior JET, Professional Edition, is capable of compiling classes loaded by those server-specific classloaders, and caching the results of compilation to disk. For better performance, you may optimize the accumulated JIT cache using the JIT cache optimizer.

To sum up, in order to compile a typical Java application server together with deployed applications into binary executables using Excelsior JET, you have to perform three major steps:

1. Compile the application server engine to an executable. During this step, all jars constituting the engine will be precompiled, so they need not be included in the deployment package.
2. Launch the compiled server and put some load on your applications to accumulate the JIT cache.
3. Optimize the JIT cache to a single binary executable.

Note: You would still have to include the original application jars in the deployment package, since it can not be guaranteed that all application classes were used during load testing and thus put in the JIT cache. Moreover, even if some class was compiled and cached, it may happen that it has to be compiled again.

DETAILS

We will now illustrate the process of optimizing Web applications deployed onto Caucho Resin application server. We used Caucho Resin version 3.0.8, so if you are using another version of Resin, details may be slightly different.

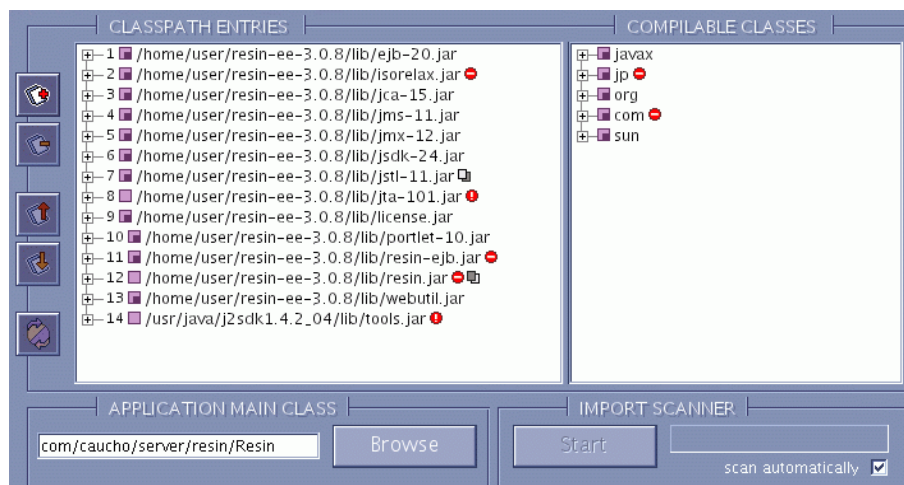
Compiling Resin

Below we assume that *Resin-dir* is the Resin installation directory (`/home/user/resin-ee-3.0.8` in the screenshots.)

1. First of all, create the directory *Resin-dir/Excelsior*. The JET-compiled Resin executable will be placed there along with startup scripts and auxiliary files.
2. Start the JET Control Panel, click **New** on the splash screen and select **Manual settings** on the **Source** page.
3. On the **Classes** page, add to the classpath all jars from *Resin-dir/lib* and also the file `lib/tools.jar` from your Sun JDK installation directory.

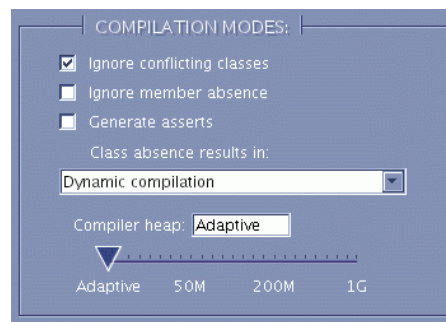
For each jar file you add, answer “**Yes**” in reply to the question “**Force the entire jars into the project?**” to make sure that all classes are compiled.

4. Specify the main class: `com.caucho.server.resin.Resin`

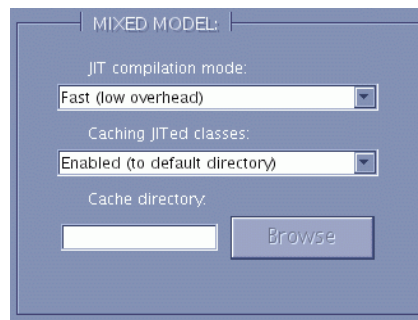


5. You will see some warnings on the **Classes** page. The reasons and remedies are as follows:

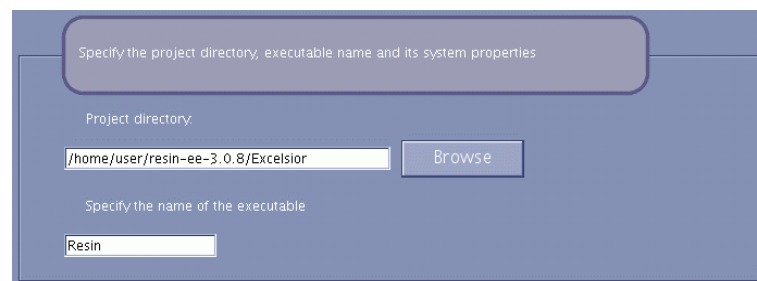
- A few Java platform classes are duplicated in Resin jars and in the JDK `tools.jar`. To prevent compilation errors, check the option **Ignore conflicting classes** on the **Options** page.
- Some Resin classes, even though explicitly imported, are generated and loaded on-the-fly, so they are absent in Resin jars, and cannot be compiled statically. To ensure proper handling of such classes at run time, select **Dynamic compilation** in the **Class absence results in** listbox.



6. On the same **Options** page, select **Fast** as the preferred **JIT compilation mode** and **Enabled (to default directory)** in the **Caching JITed classes** listbox. Use of the fast JIT is preferred, because all cached classes will be re-compiled later anyway, with the highest level of optimization.



7. On the **Target** page, choose *Resin-dir/Excelsior* as the project directory and set “Resin” as the name of the resulting executable.



8. Finally, on the **Compile** page click the **Build** button answering “Yes” in the dialog window that appears.

After successful completion of the compilation process, close the JET Control Panel.

Patching startup scripts

Resin is a Java application and thus is normally started using the `java` tool, which allows you to specify Java system properties on the command line. JET-compiled Resin is a binary executable that is run by itself, so it reads Java system properties from the `JETVMPROP` environment variable. Furthermore, JET-compiled applications recognize a slightly different format of Java properties than the Sun `java` tool does. Thus, you have to patch one of Resin startup scripts. This section explains the workflow.

- First, copy all files from the directory *Resin-dir/bin* to *Resin-dir/Excelsior*, then open the file *Resin-dir/Excelsior/wrapper.pl* in a text editor.

- Excelsior JET runtime uses ':' instead of '=' as a separator between system property name and value. So you need to change the lines where `$JAVA_ARGS` and `$EXTRA_JAVA_ARGS` are defined.

You also need to temporarily set the property `jet.jit.save.classes` to enable subsequent recompilation of the JIT cache.

Replace the line

```
$EXTRA_JAVA_ARGS="-Djava.util.logging.manager=com.caucho.log.LogManagerImpl";
```

with

```
$EXTRA_JAVA_ARGS="-Djet.jit.save.classes \  
-Djava.util.logging.manager:com.caucho.log.LogManagerImpl";
```

Replace the line

```
$JAVA_ARGS .= " -Dresin.home=$SERVER_ROOT $EXTRA_JAVA_ARGS";
```

with

```
$JAVA_ARGS .= " -Dresin.home:$SERVER_ROOT $EXTRA_JAVA_ARGS";
```

- The JET runtime reads system properties from the `JETVMPROP` environment variable. Thus, replace the lines

```
$ENV{"CLASSPATH"} = $CLASSPATH;  
if ($JAVA_HOME) {  
    $ENV{"JAVA_HOME"} = $JAVA_HOME;  
}
```

with the line

```
$ENV{"JETVMPROP"} = $JAVA_ARGS;
```

This Perl statement is analogous to the shell command `export JETVMPROP=$JAVA_ARGS`.

- Finally, replace the lines

```
exec("$JAVA_EXE $JAVA_ARGS $class $conf $RESIN_ARGS");  
die("Can't start java: $JAVA_EXE $JAVA_ARGS $class $conf $RESIN_ARGS");
```

with

```
exec("$RESIN_HOME/Excelsior/Resin $conf $RESIN_ARGS");  
die("Can't start java: $RESIN_HOME/Excelsior/Resin $conf $RESIN_ARGS");
```

so that the Resin executable is launched instead of the `java` tool.

Note: The above are the basic changes required to run Resin properly on the next step. Before using JET-compiled Resin in production, you should edit the script more thoroughly. That is, edit the `if ($verbose) { ... }` block, check all the parameters passed to the script and so on.

Accumulating JIT cache

Make sure Resin is not running and invoke the script `Resin-dir/Excelsior/http.sh` to launch the JET-compiled Resin.

Now, stress test your Web application(s) to achieve the largest possible code coverage. Ideally, you should use all features, all modes of operation, and all their combinations on large amounts of heterogeneous input data. After that, shut down the Resin server using the `Ctrl-C` key sequence.

Note however that non-exhaustive testing does not imply that your application will not function properly after JET-compilation. The idea is that thorough testing will result in caching of more classes, so more classes will be optimized on the next step.

When you are done with stress testing, remove `jet.jit.save.classes` property setting from `Resin-dir/Excelsior/wrapper.pl` again:

Replace the line

```
$EXTRA_JAVA_ARGS="-Djet.jit.save.classes \  
-Djava.util.logging.manager:com.caucho.log.LogManagerImpl";
```

with

```
$EXTRA_JAVA_ARGS="-Djava.util.logging.manager:com.caucho.log.LogManagerImpl";
```

Optimizing JIT cache

1. In the directory `Resin-dir/Excelsior` create the file `webapps.prj` with the following content:

```
-lookup=*.sym=appsym;sym  
-lookup=*.bod=appbod;bod  
-lookup=*.obj=appobj  
-lookup=*.${dllex}=jittestemp  
-lookup=*.cache=jittestemp  
  
!module jit$(jetver)$(profile).cache
```

2. Invoke the command

```
jc =p webapps.prj
```

to create the JET-compiled version of your application.

A single, optimized binary containing your web application code and a JIT cache descriptor will be created in the directory `Resin-dir/Excelsior/jittestemp`.

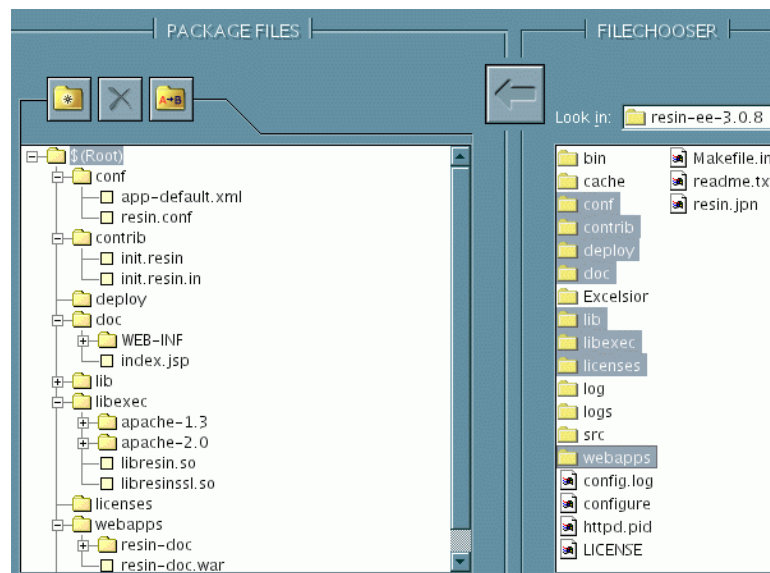
3. Invoke `Resin-dir/Excelsior/http.sh` again to make sure your optimized applications work.

Deploying

Now, if you need to deploy the optimized applications to other systems, prepare an installation package containing Resin and your applications. To do that, launch JetPackII and follow the instructions below.

- Splash screen. Click **New**.
- On the **New** page, click the **New** button.
- On the **Files** page, add files to the installation package as follows:
 1. From the **Filechooser** panel, drag the following subdirectories of the Resin directory (with their contents) onto the **Package Files** panel:

```
conf          lib  
contrib       libexec  
deploy        licenses  
doc           webapps
```



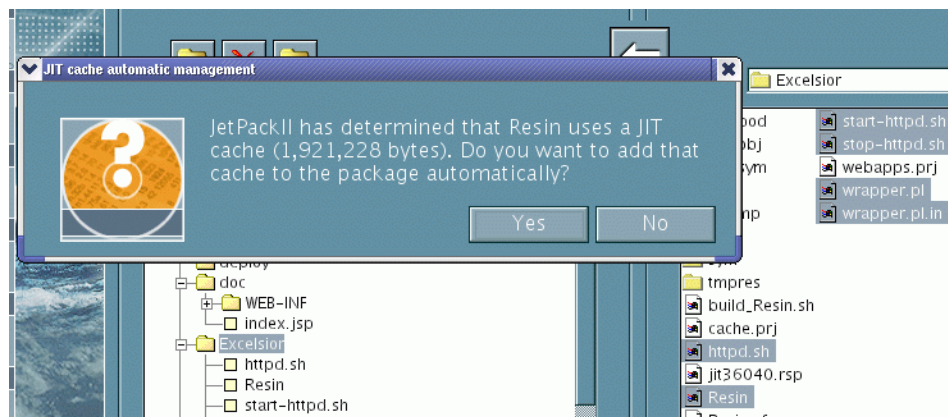
2. In the **Package Files** panel, create an empty directory `Excelsior` in the root of the package.
3. Drag the following files from the `Excelsior` directory in the **Filechooser** panel to the `Excelsior` package directory you just created:

```

Resin
httpd.sh
start-httpd.sh
stop-httpd.sh
wrapper.pl
wrapper.pl.in

```

JetPackII shall prompt you for automatic inclusion of the existing JIT cache (which is used by Resin executable compiled with JET) into the installation package. Click **Yes**.



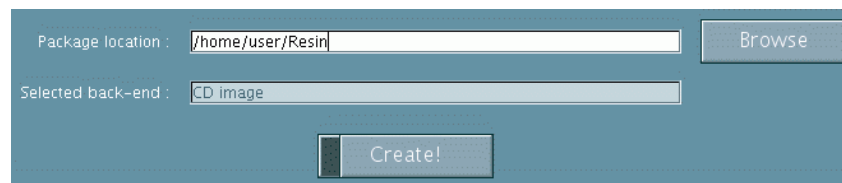
- Skip the **Resources** page.
- On the **JRE** page, choose "Use internal JRE".
- On the **JET RT** page, select additional locales as appropriate.

Note: The **JetPackII** tool can only detect the JET runtime components that are directly used by Resin and precompiled Web applications. If your application uses some Java 2 platform API, but the classes accessing that API were not loaded during JIT cache accumulation, there is a chance that the respective JET runtime shared libraries will not be included in the deployment package.

To prevent such situation, select "**Include JET runtime in whole**".



- On the **Trial Run** page, check the "Skip trial run" checkbox and click "Yes" in the dialog appeared. (Trial Run must be skipped due to a limitation of the current version of JetPackII: it may only launch executables directly, whereas the Resin application server is run from a script.)
- On the **Backend** page, select "CD image" as the preferred backend type.
- On the **Finish** page, select the desired location of the image and click the **Create!** button.



After the packaging completes, you can tar/gzip the resulting image and test it on a machine not having J2SE/Resin installed.

REFERENCES

1. Excelsior JET User's Guide: <http://www.excelsior-usa.com/doc/jc.html>, Chapter "Mixed Compilation Model".