

HOWTO: Compile and deploy an SWT-based Java application

Article ID: 000008

Last Revised On: 26-Sep-2005

The information in this article applies to:

- Excelsior JET, Professional Edition versions 3.11 through 3.7.

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

SUMMARY

The document contains step-by-step instructions on the use of **Excelsior JET, Professional Edition** for the conversion of a Java GUI application that uses SWT/JFace instead of AWT/Swing into a JRE-independent binary executable. On the first few steps, the application is compiled to native code. The remaining steps show how to create an installation package for the application using setup authoring tools that come with Excelsior JET.

Source and project files for this article may be found [here](#).

A sample installation package created using this article can be found [here](#).

DETAILS

Introduction

Suppose you have a Java application that uses SWT/JFace for the GUI but does **not** use AWT/Swing. Then you may use Excelsior JET, Professional Edition to compile that application into a binary executable (EXE file) that does not require a JRE to run. This document describes how to achieve that and also how to package your application for deployment to other systems.

As an example, we will use `MandSet`, a simple SWT-based program that draws the Mandelbrot set.

Compile the application

Below we assume that the classes of your application are placed into one or more jar files. If your application loads resources at runtime (which is very likely, at least it must be loading some GIF images), pack the resource files into a jar file too, or just add them to one of the jars containing your application's classes.

For instance, the classes of our sample application are packaged into `MandSet.jar` along with the only resource file it needs (that is `MandSet.gif` used to set the icon of the main window).

First off, compile the Java sources of your application and make sure that it works properly under the Sun HotSpot VM (version 1.3.xx or 1.4.xx). For this article's sample, use scripts `buildjar.bat` and `runjar.bat` respectively.

You then need to create an Excelsior JET project file for your app. There is already `MandSet.prj` in the sample zip, but you may wish to create it yourself from scratch. The easiest way to do that is using

the JET Control Panel. Its UI is organized as a series of steps. On each step, you can specify or alter some settings of the resulting executable.

Note: This guide only covers the features of the JET Control Panel that are needed to properly set up a project for the MandSet sample. Refer to the Excelsior JET User's Guide for more information on JET GUI.

To create a project file for your application and compile it, do the following:

1. Launch the JET Control Panel from the Excelsior JET Start Menu folder or by clicking the Excelsior JET desktop icon. On the splash screen, click the **New** button.
2. On the **New** page, select "Manual Settings". You will get directly to the **Classes** page.
3. On the **Classes** page, add the jars where the classes of your application reside, and the file `swt.jar` to the classpath. Use the "+" button to the left from the **Classpath Entries** pane to display the **Add classpath entries** dialog. For each jar you will be adding, answer "yes" to the question whether you want to compile all classes from that jar.

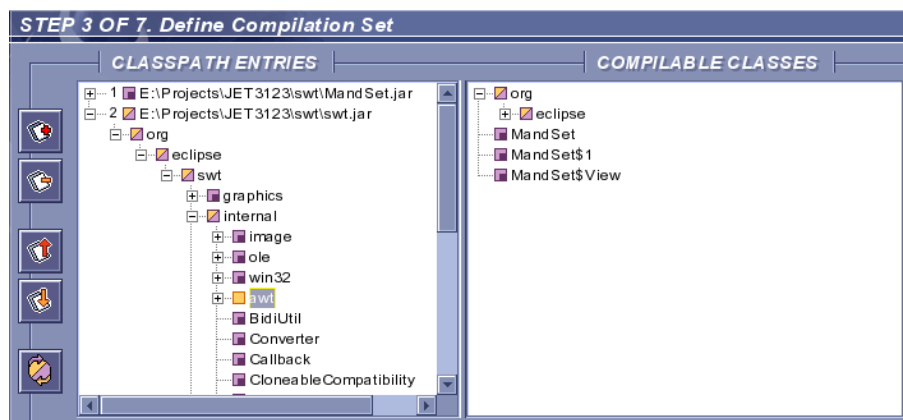
Now comes a somewhat tricky part — getting rid of JRE dependency. There are classes in `swt.jar` that implement an SWT-AWT bridge. Those classes import `java.awt.*`, and are the only part of SWT that require the AWT API. If your app makes no use of that bridge, remove it from the compilation set to enable the creation of a JRE-independent executable.

4. Expand the `swt.jar` node in **Classpath Entries** or the `org` node in **Compilable classes** until you find the package `org.eclipse.swt.internal.awt`.

Right-click that package, and clear the **Force into the project** flag in the popup menu. As a result, all classes of the package will be removed from the compilation set. Visually, their status tags will become orange (meaning "not included").

Figure 1 illustrates the result of this action.

Figure 1: Remove the "awt" package from the compilation set.



5. Click the **Browse** button in the **Application Main Class** pane below. Select the main class of your application (in our sample that would be `MandSet`.) Proceed to the next page, **Resources**.
6. The **Resourcepath** list will include all classpath entries you defined earlier. If there are any resource-only jars in your application, you should add them to the resourcepath now using the "+" button.

7. If you want to associate an icon with the resulting executable, click **Browse** button in the **Application Icon** pane and select the desired `.ico` file. There is the `MandSet.ico` file in the sample.

The next page, **Options**, allows you to modify compiler settings. For our sample app, the defaults are ok, so you may go directly to the **Target** page.

8. By default, your compiled application will have a console window as when you run it using the `java` command. However, Java GUI applications, such as `MandSet`, are normally run using the `javaw` command, which does not display a console window. Check the **Suppress console window** checkbox on the **Target** page if that is the case and proceed to the last page.

9. On the **Compile** page, click **Build**. As you have removed the AWT-SWT bridge classes from the compilation set, you will now be prompted to confirm that the compilation set is complete. Select **Yes**.

After that, you will be asked for the name and location of the project file. Select the directory and type the desired file name. Note that the resulting executable will be created in the same directory and will have the same name (ending with `".exe"` of course).

Now the build process shall start. For our sample, it is likely to take not more than a few minutes on a reasonable system configuration.

You may further optimize your application using JetPerfect Global Optimizer (to reduce installation package size) and executable image optimizer (to reduce application startup time). For more details, consult the respective Chapters of Excelsior JET User's Guide.

10. When the build completes, launch the compiled application by clicking **Run**. Check that it works properly and return to the JET Control Panel.

Create the installation package

Now you need to package your application for deployment to enduser systems. The JetPackII tool will assist you. Its user interface is organized in a manner very similar to that of JET Control Panel, as a series of steps. On each step, you can alter some settings of the resulting installation package.

Note: This guide only covers a few basic features of JetPackII. Please refer to Excelsior JET User's Guide for more information.

To package your optimized SWT-based application for deployment to enduser systems, do the following:

1. Assuming that you are still on the **Compile** page of the JET Control Panel, click **Package**. The JET Control Panel shall close and the **JetPackII** tool shall appear.

The **New** page is skipped when you run JetPackII from the JET Control Panel.

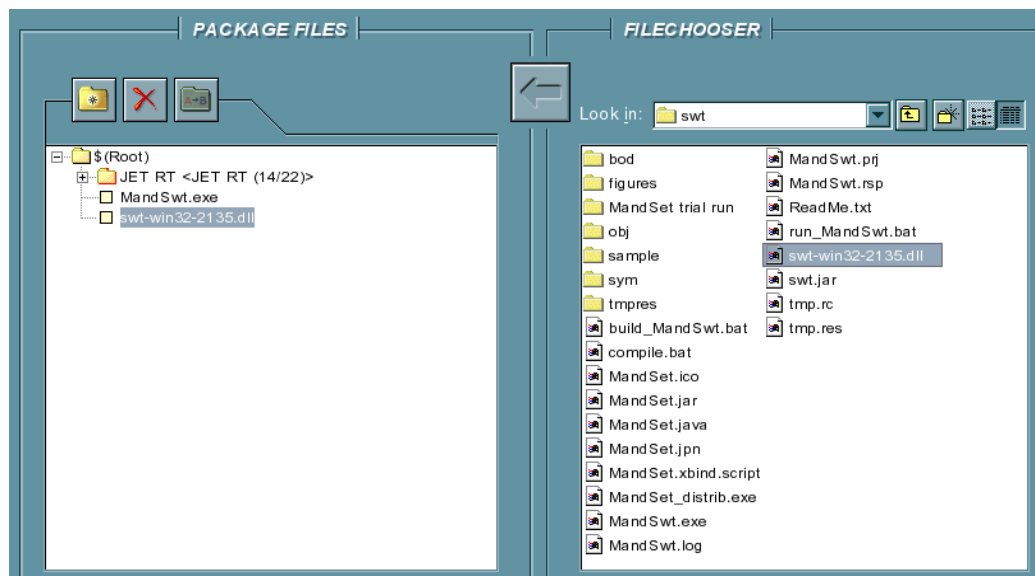
2. First, you have to select the files to be deployed along with the just created executable. Note that if you have put the resource files into jars before compilation, they are already bound to the executable, so you need not add them to the installation.

There is one more file that is necessary for the applicaton to run — the SWT native method library (`swt-win32-2135.dll` in our example).

As shown on Figure 2, that file must be added to the Package Files list, into the same directory where the executable file resides.

In the right panel, navigate to the folder where that file resides. Drag the file to the left panel. Do the same for any other files you wish to deploy with your application (data files, other executables, documentation, etc.).

Figure 2: Drag swt-win32-xxxx.dll to the left panel.



You may associate resource files and properties with JET-compiled executables included in your installation package on the next page, **Resources**. For MandSet, there is no need to change anything on that page, so you may skip it.

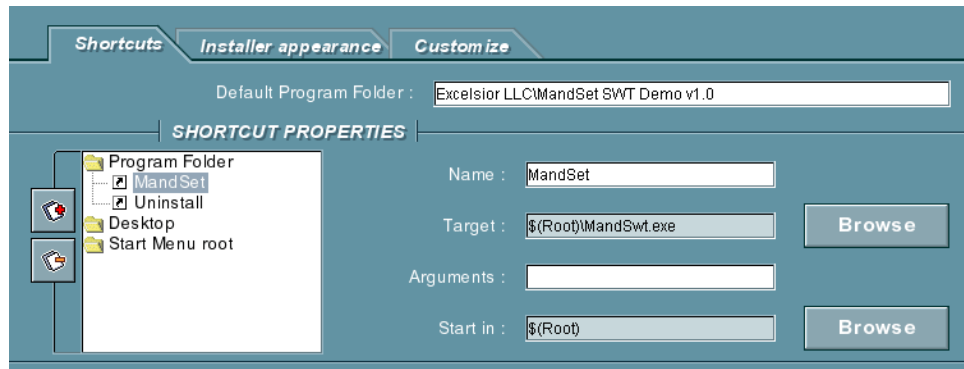
3. On the **JRE** page, select **Do not use JRE**, read the warning message and click **Ok**.
4. In the right panel on the **JET RT** page, select the locales you would like your application to support. Including all locales would increase the size of the installation package approximately by 600KB.
5. On the **Trial Run** page, you run your application as if it was deployed to a enduser system in order to confirm the completeness and integrity of the installation package defined on previous pages.
Click **Run**. You will be prompted for a name and location of the JetPackII project file. Type in the desired name and click **Save**. Your application shall start. Check that it works properly, and close it.
6. On the **Backend** page, select **Excelsior Installer**.
7. On the **Misc** page, enter company name, product name and version as you would like them to be shown during the installation (Figure 3 illustrates this action).

Figure 3: Fill the vendor information.

 The image shows a form titled 'VENDOR INFORMATION'. It contains three text input fields. The first field is labeled 'Company name' and contains the text 'Excelsior LLC'. The second field is labeled 'Product name' and contains the text 'MandSet SWT Demo'. The third field is labeled 'Product version' and contains the text 'v1.0'.

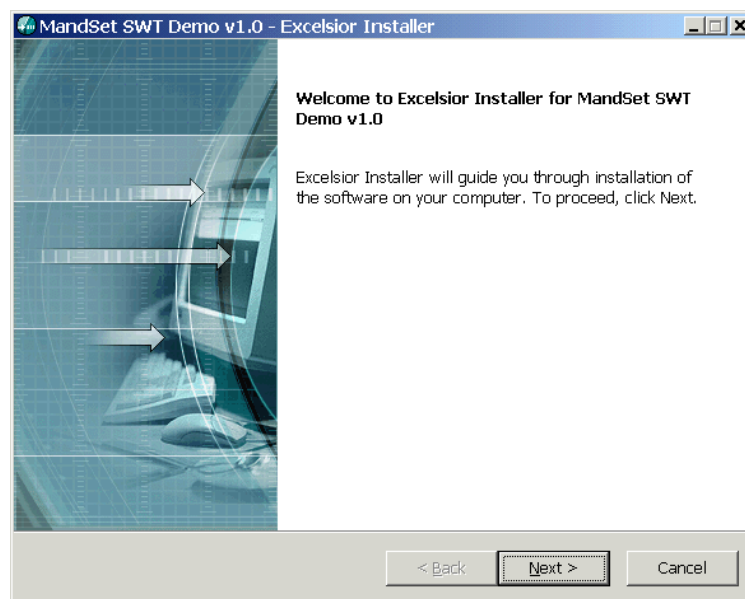
- Now, define a Start Menu shortcut for your application. On the **Shortcuts** tab below, click "+". Enter the desired name for the shortcut. Click **Browse** and select the target executable for the shortcut (Figure 4).

Figure 4: Define a shortcut for the executable.



- On the **Finish** page, click **Create** to initiate the installation package build. After its completion, you shall have a self-installing executable ready for deployment to enduser systems. If you launch it, the Excelsior Installer splash screen shall display, as shown on Fig. 5.

Figure 5: Excelsior Installer splash screen



The size of the MandSet installation package should be about 3.8 MB (if compiled against J2SE 1.3.1 without global optimizations.) Compare this to the size of JRE 1.3.1 alone (7.8 MB).