

# PRB: java.io.UnsupportedEncodingException is thrown during execution

Article ID: 000016

Last Revised On: 29-Sep-2005

The information in this article applies to:

- Excelsior JET versions 2.5 through 3.5

**Note: this article does not apply to Excelsior JET 3.6 and above.**

## SYMPTOMS

An application that works properly on the Sun JRE throws `java.io.UnsupportedEncodingException` when run as a JET-compiled executable.

## CAUSE

Java uses 16-bit Unicode to represent and manipulate character data, whereas native applications on most operating systems use some other encodings. When a Java application needs to convert character data from/to some encoding, it attempts to dynamically load the respective conversion class.

JET Setup precompiles most of the JRE classes related to internationalization support, including character conversion classes, into a set of separate runtime DLLs (`XL*XXXXX.DLL`). Depending on factors like the target geography and language support, some or all of those DLLs have to be included in the application's installation package.

Another point to consider is that the Sun JRE supports, in particular, rare and legacy character encodings that are not likely to be used by an average program. Therefore JET Setup only precompiles the character conversion classes for commonly used encodings. That considerably reduces the disk footprint of compiled applications, because encodings for languages like Chinese or Japanese may require large conversion tables.

Now, if the conversion class requested by the application does not get precompiled into a JET runtime DLL, or that DLL was not included in the application's installation package, the class load attempt fails and `java.io.UnsupportedEncodingException` is thrown.

## RESOLUTION

- **The problem occurs on enduser systems only, but not on the system where application is built.**

This means that your installation package does not include the respective JET runtime DLL(s). Open your deployment project in JetPackII and do either of the following:

- use the **Trial Run** feature to determine which DLLs are needed
- explicitly include support for the respective locales on the **JET RT** page

- **The problem occurs on the build system.**

This means your application needs to convert character data from/to an encoding that is not commonly used on the underlying operating system and thus the respective conversion classes were not precompiled by JET Setup. You have to include them in the project explicitly as shown below.

The internal classes that perform actual character conversions in packages `java.lang` and `java.io` are named `sun.io.ByteToCharcharset-name` and `sun.io.CharToBytecharset-name`. *charset-name* is the symbolic name of the charset, such as “Big5”, “Cp866”, or “MacArabic”, which you substitute as the `charsetName` parameter to the respective methods of platform API classes, such as `java.lang.String.getBytes()`.

To include conversion classes in the compilation set, do the following:

- Using the JET Control Panel:
  1. Open your project and go to the **Classes** page.
  2. Add the file `lib\charsets.jar` (`lib\i18n.jar` if using JRE 1.3) from the JRE directory to the classpath.
  3. Force the required conversion classes into the project.
  4. Rebuild the executable.
- Using the command-line compiler:

Add to your project the following lines:

```
-lookup = *.class = JRE-1.4-dir\lib\charsets.jar    or
-lookup = *.class = JRE-1.3-dir\lib\i18n.jar
. . .
!module sun/io/ByteToCharcharset-name.class
!module sun/io/CharToBytecharset-name.class
```

## REFERENCES

Java 2 SDK, Standard Edition Documentation

<http://java.sun.com/docs/index.html>

See Guide to Features - Internationalization - Supported Encodings for the complete list of encodings supported by the JRE and their symbolic names.