

# Getting started with Java native methods

Last Updated: 2021-04-14




You should only use native methods in cases where pure Java™ cannot meet your programming needs.

Limit the use of native methods by only using them under these circumstances:

- To access system functions that are not available using pure Java.
- To implement performance-sensitive methods that can benefit significantly from a native implementation.
- To interface to existing application programming interfaces (API) that allow Java to call other APIs.

The following instructions apply to using the Java Native Interface (JNI) with the C language. For information about using JNI with the RPG language, see Chapter 11 of the WebSphere® Development Studio: ILE RPG Programmer's Guide, SC09-2507.

 **Note:** The term native library or native method library refers to integrated language environment (ILE) service programs when used in the context of ILE native methods, and AIX® static or shared libraries when used in the context of PASE for i native methods.

To create Java native methods, do these steps:

1. Create the Java class and specify which methods are native methods using the standard Java language syntax.

In the static initializer for the class, you need to add code that loads the native library which contains the C implementation of the native methods. You can use either the `System.load()` or `System.loadLibrary()` Java methods to load the native library. The `System.load()` method takes as a parameter a fully qualified path to the native library and loads the specified native library. The `System.loadLibrary()` takes as parameter a library name, locates a native library that corresponds to that name, and loads the native library. For information about how a native library is located by the `System.loadLibrary()` method, see [Managing native method libraries](#).

You need to be aware of the following library naming conventions:

- If the native methods are [ILE native methods](#) and the Java code loads a library named `Sample`, the corresponding executable file must be an ILE service program named `SAMPLE`. The following shows how you would load the ILE native library:

```
System.loadLibrary("Sample");  
  
System.load("/qsys.lib/mylib.lib/Sample.srvpgm");
```

**Note:** A symbolic link to a service program can be used in these library loading methods.

- If the native methods are [PASE for i native methods](#) and the Java code loads a library named `Sample`, the corresponding executable file must be an AIX library named either `libSample.a` or `libSample.so`. The following shows how you would load the PASE for i native library:

```
System.loadLibrary("Sample");  
  
System.load("/somedir/libSample.so");
```

2. Use the `javac` tool to compile the Java source into a class file.
3. Use the `javah` tool to create the header file (.h). This header file contains the exact prototypes for creating the native method implementations. The `-d` option specifies the directory where you should create the header file.
4. Write the C implementation code for the native method. See the [Java native methods and threads considerations](#) topic for details about the languages and functions that are used for native methods.
  - a. Include the header file that was created in the previous steps.
  - b. Match the prototypes in the header file exactly.
  - c. If your native method must interact with the Java virtual machine, use the functions that are provided with JNI.
  - d. For ILE native methods only, convert strings to American Standard Code for Information Interchange (ASCII) if the strings are to be passed to the Java virtual machine. For more information, see [Strings in ILE native methods](#).
5. Compile your C implementation code for the native method into a native library.
  - For ILE native methods, use the Create C Module (CRTCMOD) command to compile source files into module objects. Then bind one or more module objects into a service program by using the Create Service Program (CRTSRVPGM) command. The name of this service program must match the name that you supplied in your Java

code that is in the `System.load()` or `System.loadLibrary()` Java method calls.

**Note:** The implementation code for the native method must be compiled with teraspace storage enabled. For more information about teraspace and native methods, see [Teraspace storage model native methods for Java](#).

- For PASE for i native methods, use the `xlc` or `xlc_r` commands to compile and build an AIX library. For more information about compiling and building libraries for PASE for i, see [Compiling your AIX source](#) topic.

6. If you used the `System.loadLibrary()` call in your Java code to load the native library, perform one of the following tasks:

- Include the list of the native library paths that you need in the `LIBPATH` environment variable. You can change the `LIBPATH` environment variable in QShell and from the IBM i command line.
  - From the Qshell command prompt, type in:
 

```
export LIBPATH=/QSYS.LIB/MYLIB.LIB
java myclass
```
  - Or, from the command line:
 

```
ADDENVVAR LIBPATH '/QSYS.LIB/MYLIB.LIB'
JAVA myclass
```
- Or, supply the list in the **java.library.path** property. You can change the `java.library.path` property in QShell and from the IBM i command line.
  - From the Qshell command prompt, enter:
 

```
java -Djava.library.path=/QSYS.LIB/MYLIB.LIB myclass
```
  - Or, from the IBM i command line, type in:
 

```
JAVA PROP((java.library.path '/QSYS.LIB/MYLIB.LIB')) myclass
```

Where `/QSYS.LIB/MYLIB.LIB` is the path that contains the native library you want to load using the `System.loadLibrary()` call, and `myclass` is the name of your Java application.

For information about how a native library is located by the `System.loadLibrary()` method, see [Managing native method libraries](#).

For an example of an ILE native method, see [Example: ILE native method for Java](#). For an example of a PASE for i native method, see [Example: IBM PASE for i native method for Java](#).

**Parent topic:**

→ [Native methods and the Java Native Interface](#)

- [Websphere Development Studio: ILE RPG Programmer's Guide, SC09-2507.](#)
- [Java native methods and threads considerations](#)
- [Java Native Interface by Sun Microsystems, Inc.](#)
- [Example: ILE native method for Java](#)
- [Strings in ILE native methods](#)
- [Java character encodings](#)