**JAVA的相关问题**

# 1、抛出java.lang.NoClassDefFoundError错误

Exception in thread "main" java.lang.NoClassDefFoundError: net/netca/pki/Base64

at Base64File.main(Base64File.java:32)

Caused by: java.lang.ClassNotFoundException: net.netca.pki.Base64

at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(Unknown So

urce)

at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(U

nknown Source)

at java.base/java.lang.ClassLoader.loadClass(Unknown Source)

... 1 more

这个通常是找不到要加载的jar包。上面例子是net/netca/pki/Base64这个类找不到，也就是net.netca.pki.Base64这个类。这个类是在NetcaJCrypto.jar里的，因此，要检查为何NetcaJCrypto.jar不能加载。JAR的加载由JAVA的类装载器决定。通常，在JRE目录/lib/ext目录（该目录在高版本的JAVA中不存在）下，或者由CLASSPATH环境变量指定。对应Web容器，也可以是Web容器的lib目录或者Web应用的WEB-INF/lib目录（使用JNI在该目录可能会存在问题）。根据不同的Web容器的配置，搜索的顺序是不同的，可能最先找Web应用的WEB-INF/lib目录，也可能是最后才找Web应用的WEB-INF/lib目录。

另外一种可能是一开始加载JNI失败导致无法初始化net.netca.pki.Base64这个类。以后再加载的时候，直接抛出NoClassDefFoundError错误。也就是第一次抛出UnsatisfiedLinkError错误，以后抛出NoClassDefFoundError错误。有时候错误信息会更详细一些，比如：

java.lang.NoClassDefFoundError: Could not initialize class net.netca.pki.Base64。

另外，还要留意权限问题，可能应用没有NetcaJCrypto.jar的权限。

# 2、SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

## 情况1：

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

这个不是错误。只是告诉你没有找到slf4j的后端日志库，因此使用默认的不做任何日志记录的后端日志库，这时是不会写日志的。

slf4j日志在NetcaJCrypto.jar新版本中是可以通过系统属性或者环境变量来关闭的。通过启动时设置net.netca.pki.log.enabled系统属性的值为false，可以关闭slf4j日志，这时也不依赖于slf4j-api-x.y.z.jar。同样地，启动时设置环境变量NetcaJCryptoLogEnabled的值为false，可以关闭slf4j日志。

## 情况2：

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further detail

s.

Exception in thread "main" java.lang.NoClassDefFoundError: org/slf4j/impl/Static

LoggerBinder

at org.slf4j.LoggerFactory.<clinit>(LoggerFactory.java:60)

at net.netca.pki.LoggerFactory.getLogger(LoggerFactory.java:39)

at net.netca.pki.Base64.<init>(Base64.java:66)

at Base64File.main(Base64File.java:32)

Caused by: java.lang.ClassNotFoundException: org.slf4j.impl.StaticLoggerBinder

at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(Unknown So

urce)

at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(U

nknown Source)

at java.base/java.lang.ClassLoader.loadClass(Unknown Source)

... 4 more

注意，没有SLF4J: Defaulting to no-operation (NOP) logger implementation这一行，多了java.lang.NoClassDefFoundError: org/slf4j/impl/StaticLoggerBinder的异常。这是因为使用了旧版本的slf4j库。更新新版本的slf4j库或者添加一个对应版本的slf4j后端日志库即可解决问题。

# 3、抛出java.lang.UnsatisfiedLinkError错误

## 情况1：

Exception in thread "main" java.lang.UnsatisfiedLinkError: no NetcaJCrypto in java.library.path

at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1864)

at java.lang.Runtime.loadLibrary0(Runtime.java:870)

at java.lang.System.loadLibrary(System.java:1122)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

no NetcaJCrypto in java.library.path表示没有找到NetcaJCrypto这个JNI。新版本的java还会把java.library.path的值列出来。首先，要确定java.library.path的值（新版的JAVA会在异常信息中把java.library.path的值也列出来的），然后把NetcaJCrypto.dll（windows）或者libNetcaJCrypto.so（linux）放到对应的路径中。

NetcaJCrypto.jar新版本可以通过系统属性和环境变量来打印java.library.path的值。通过在启动的时候设置net.netca.pki.printstartinfo系统属性的值为ON，或者设置环境变量NetcaJCryptoPrintStartInfo的值为ON，可以在标准输出中打印出一些相关的设置信息，其中包括java.library.path的值。NetcaJCrypto.jar新版本还可以通过net.netca.pki.jnipath系统属性的值或者NetcaJCryptoJniPath环境变量的值直接指定JNI的包含全路径的文件名。

另外，还要留意权限问题，可能应用没有NetcaJCrypto.dll或者libNetcaJCrypto.so的权限。

这个错误会导致以后的java.lang.NoClassDefFoundError错误。解决后，可能需要重启应用。

## 情况2：

Exception in thread "main" java.lang.UnsatisfiedLinkError: C:\Windows\System32\NetcaJCrypto.dll: Can't load IA 32-bit .dll on a AMD 64-bit platform

at java.base/java.lang.ClassLoader$NativeLibrary.load0(Native Method)

at java.base/java.lang.ClassLoader$NativeLibrary.load(Unknown Source)

at java.base/java.lang.ClassLoader$NativeLibrary.loadLibrary(Unknown Source)

at java.base/java.lang.ClassLoader.loadLibrary0(Unknown Source)

at java.base/java.lang.ClassLoader.loadLibrary(Unknown Source)

at java.base/java.lang.Runtime.loadLibrary0(Unknown Source)

at java.base/java.lang.System.loadLibrary(Unknown Source)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是告诉你，NetcaJCrypto.dll是32位的版本，但JAVA需要的是64位的版本。因此，使用对应版本的JNI即可

## 情况3：

Exception in thread "main" java.lang.UnsatisfiedLinkError: /home/test/lib64/libNetcaJCrypto.so: /home/test/lib64/libNetcaJCrypto.so: 错误 ELF 类: ELFCLASS32 (Possible cause: architecture word width mismatch)

at java.lang.ClassLoader$NativeLibrary.load(Native Method)

at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1968)

at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1893)

at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1883)

at java.lang.Runtime.loadLibrary0(Runtime.java:849)

at java.lang.System.loadLibrary(System.java:1088)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是告诉你，libNetcaJCrypto.so是32位的版本，但JAVA需要的是64位的版本。因此，使用对应版本的JNI即可

## 情况4：

Exception in thread "main" java.lang.UnsatisfiedLinkError: C:\Windows\System32\NetcaJCrypto.dll: Can't find dependent libraries

at java.base/java.lang.ClassLoader$NativeLibrary.load0(Native Method)

at java.base/java.lang.ClassLoader$NativeLibrary.load(Unknown Source)

at java.base/java.lang.ClassLoader$NativeLibrary.loadLibrary(Unknown Sou

rce)

at java.base/java.lang.ClassLoader.loadLibrary0(Unknown Source)

at java.base/java.lang.ClassLoader.loadLibrary(Unknown Source)

at java.base/java.lang.Runtime.loadLibrary0(Unknown Source)

at java.base/java.lang.System.loadLibrary(Unknown Source)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个告诉你，没有找到NetcaJCrypto.dll依赖的库，安装中间件可以解决。NetcaJCrypto.dll本身所在的路径是由JAVA的JNI路径决定(通常是java.library.path)，而NetcaJCrypto.dll依赖的dll的路径却是由windows的dll加载路径决定，通常是PATH变量的路径。

## 情况5：

Exception in thread "main" java.lang.UnsatisfiedLinkError: /home/test/lib64/libNetcaJCrypto.so: libnetca\_crypto.so.1: 无法打开共享对象文件: 没有那个文件或目录

at java.lang.ClassLoader$NativeLibrary.load(Native Method)

at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1968)

at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1893)

at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1883)

at java.lang.Runtime.loadLibrary0(Runtime.java:849)

at java.lang.System.loadLibrary(System.java:1088)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是告诉你libNetcaJCrypto.so依赖的libnetca\_crypto.so.1找不到。检查libnetca\_crypto.so.1文件是否存在，libnetca\_crypto.so.1文件通常是符合连接，还要检查libnetca\_crypto.so.1指向的文件是否存在。libnetca\_crypto.so.1通常应该在LD\_LIBRARY\_PATH路径下。

libNetcaJCrypto.so本身所在的路径是由JAVA的JNI路径决定(通常是java.library.path)，而libNetcaJCrypto.so依赖的so的路径却是由linux的so加载路径决定，通常是LD\_LIBRARY\_PATH变量的路径。

可以用ldd libNetcaJCrypto.so检查一下依赖关系

## 情况6：

Exception in thread "main" java.lang.UnsatisfiedLinkError: C:\Windows\System32\NetcaJCrypto.dll: %1 不是有效的 Win32 应用程序。

at java.base/java.lang.ClassLoader$NativeLibrary.load0(Native Method)

at java.base/java.lang.ClassLoader$NativeLibrary.load(Unknown Source)

at java.base/java.lang.ClassLoader$NativeLibrary.loadLibrary(Unknown Sou

rce)

at java.base/java.lang.ClassLoader.loadLibrary0(Unknown Source)

at java.base/java.lang.ClassLoader.loadLibrary(Unknown Source)

at java.base/java.lang.Runtime.loadLibrary0(Unknown Source)

at java.base/java.lang.System.loadLibrary(Unknown Source)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是使用64为java和NetcaJCrypto.dll，但是有32位的NETCA\_CRYPTO.dll，抛出的异常。也就是JNI和中间件版本不同导致的。

## 情况7：

Exception in thread "main" java.lang.UnsatisfiedLinkError: /home/test/lib64/libNetcaJCrypto.so: libnetca\_crypto.so.1: 错误 ELF 类: ELFCLASS32

at java.lang.ClassLoader$NativeLibrary.load(Native Method)

at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1968)

at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1893)

at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1883)

at java.lang.Runtime.loadLibrary0(Runtime.java:849)

at java.lang.System.loadLibrary(System.java:1088)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是使用64为java和libNetcaJCrypto.so，但libnetca\_crypto.so.1却是32位的。也就是JNI和中间件版本不同导致的。

## 情况8：

Exception in thread "main" java.lang.UnsatisfiedLinkError: C:\Windows\System32\NetcaJCrypto.dll: 找不到指定的程序。

at java.base/java.lang.ClassLoader$NativeLibrary.load0(Native Method)

at java.base/java.lang.ClassLoader$NativeLibrary.load(Unknown Source)

at java.base/java.lang.ClassLoader$NativeLibrary.loadLibrary(Unknown Sou

rce)

at java.base/java.lang.ClassLoader.loadLibrary0(Unknown Source)

at java.base/java.lang.ClassLoader.loadLibrary(Unknown Source)

at java.base/java.lang.Runtime.loadLibrary0(Unknown Source)

at java.base/java.lang.System.loadLibrary(Unknown Source)

at net.netca.pki.Util.loadJNI(Util.java:36)

at net.netca.pki.Util.<clinit>(Util.java:98)

at net.netca.pki.Base64.<clinit>(Base64.java:36)

at Base64File.main(Base64File.java:32)

这个是中间件版本比JNI版本低，JNI引用的某些函数，在旧版本的中间件中没有。升级中间件即可。如果是Linux的话，如果是libNetcaJCrypto.so没使用到的函数，则不会出问题

## 情况9：

java.lang.UnsatisfiedLinkError: Native Library C:\Windows\System32\NetcaJCrypto.dll already loaded in another classloader

java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1904)

java.lang.ClassLoader.loadLibrary(ClassLoader.java:1854)

java.lang.Runtime.loadLibrary0(Runtime.java:870)

java.lang.System.loadLibrary(System.java:1122)

net.netca.pki.Util.loadJNI(Util.java:37)

net.netca.pki.Util.<clinit>(Util.java:105)

net.netca.pki.Hash.<clinit>(Hash.java:90)

SM3abc.doGet(SM3abc.java:23)

javax.servlet.http.HttpServlet.service(HttpServlet.java:622)

javax.servlet.http.HttpServlet.service(HttpServlet.java:729)

org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)

这个是说JNI已经在另一个类加载器加载过了，因此不能加载，这个是JNI的限制，不能由多个类装载器加载。最常见的情况是tomcat的两个web应用的lib目录都放有NetcaJCrypto.jar，访问第一个应用正常，访问第二个应用就会出现这个错误。默认情况下，tomcat是最优先加载web应用的lib目录的jar，不同的web应用使用不同的类加载器，因此会出现这个错误。解决的办法就是让同一类加载器加载NetcaJCrypto.jar。比如，把NetcaJCrypto.jar放在web容器的lib目录，或者JRE的ext目录，或者CLASSPATH的路径下。

如果是用新版本的NetcaJCrypto.jar还可以通过net.netca.pki.jnipath系统属性的值或者NetcaJCryptoJniPath环境变量的值直接指定JNI的包含全路径的文件名。这是通过把JNI文件拷贝到临时目录下再加载的，由于是不同路径的JNI，因此，不同类加载器加载也没问题。

这个错误会导致以后的java.lang.NoClassDefFoundError错误。解决后，可能需要重启应用。

## 情况10：

Exception in thread "main" java.lang.UnsatisfiedLinkError: net.netca.pki.SESeal.decodeSeal([BII)J

at net.netca.pki.SESeal.decodeSeal(Native Method)

at net.netca.pki.SESeal.<init>(SESeal.java:86)

at net.netca.pki.SESeal.<init>(SESeal.java:101)

at AttachSESealTimeStamp2.main(AttachSESealTimeStamp2.java:22)

这个是说JNI 中没有decodeSeal这个方法。通常是JAR包和JNI不匹配，JNI太旧还没实现新方法，一般情况下，使用对应的版本的JNI即可。

# 4、抛出java.lang.NoSuchMethodError错误

## 情况1：

Exception in thread "main" java.lang.NoSuchMethodError: net.netca.pki.PublicKey.getSM2PublicKey(Ljava/math/BigInteger;Ljava/math/BigInteger;)Lnet/netca/pki/PublicKey;

at TestSM2PublicKey.main(TestSM2PublicKey.java:22)

这里说明net.netca.pki.PublicKey类的getSM2PublicKey方法不存在。通常是因为编译产生的class文件使用的jar包比较新，部署的时候使用的jar包比较旧，没有那个新增的方法。更新新的jar包即可。net.netca.pki.PublicKey是NetcaJCrypto.jar里的，因此要更新NetcaJCrypto.jar。

## 情况2：

Exception in thread "main" java.lang.NoSuchMethodError: net.netca.pki.Hash.close

()V

at HashFile2.main(HashFile2.java:47)

这种情况是编译的时候使用netcapki-jni.jar，运行的时候使用NetcaJCrypto.jar。通常是使用了try-with-resources的写法。由于只有拆分的版本net.netca.pki.Freeable接口继承了AutoCloseable，它存在close方法，默认实现就是调用free方法。而NetcaJCrypto.jar里net.netca.pki.Freeable接口是没有close方法的，因此会出现这样的异常。解决的办法就是使用拆分的版本，也就是netcapki-jni.jar和netcapki-common.jar。

# 5、抛出java.lang.UnsupportedClassVersionError错误

Exception in thread "main" java.lang.UnsupportedClassVersionError: Base64File : Unsupported major.minor version 54.0

at java.lang.ClassLoader.defineClass1(Native Method)

at java.lang.ClassLoader.defineClass(ClassLoader.java:803)

at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)

at java.net.URLClassLoader.defineClass(URLClassLoader.java:442)

at java.net.URLClassLoader.access$100(URLClassLoader.java:64)

at java.net.URLClassLoader$1.run(URLClassLoader.java:354)

at java.net.URLClassLoader$1.run(URLClassLoader.java:348)

at java.security.AccessController.doPrivileged(Native Method)

at java.net.URLClassLoader.findClass(URLClassLoader.java:347)

at java.lang.ClassLoader.loadClass(ClassLoader.java:425)

at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)

at java.lang.ClassLoader.loadClass(ClassLoader.java:358)

at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:482)

这个是因为class文件是用新版本的javac编译的，而运行使用的java的版本比较低。升级java的版本或者用低版本的javac重新编译class文件。

# 6、symbol lookup error

java: symbol lookup error: /home/test/lib64/libNetcaJCrypto.so: undefined symbol: NetcaPKIMsgNewJWS

这个是告诉你libNetcaJCrypto.so需要调用NetcaPKIMsgNewJWS这个函数，但是找不到它的实现。NetcaPKIMsgNewJWS是libnetca\_crypto.so.1的函数。这是因为使用了旧版本的中间件的缘故，更新中间件版本可以解决。

# 7、抛出java.security.InvalidKeyException异常

Exception in thread "main" java.security.InvalidKeyException: Illegal key size or default parameters

at javax.crypto.Cipher.checkCryptoPerm(Cipher.java:1026)

at javax.crypto.Cipher.implInit(Cipher.java:801)

at javax.crypto.Cipher.chooseProvider(Cipher.java:864)

at javax.crypto.Cipher.init(Cipher.java:1249)

at javax.crypto.Cipher.init(Cipher.java:1186)

at JCEAsyEncrypt.main(JCEAsyEncrypt.java:34)

这个是告诉你JCE的策略不能支持该加密算法或者这个强度的加密算法。这个是JAVA本身的代码做检查的，不是你自己实现的JCE处理的。通常的解决办法是安装不限制强度的策略文件（Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files）。另外一种方式是使用不做该类检查的JDK（比如：openjdk、zulu jdk等）。