

Java Dependency Analysis Tool

jdeps is a new command-line tool added since JDK 8 for developers to use to understand the static dependencies of their applications and libraries. **jdeps** is a static analysis tool on the given class files and dynamic class dependencies (Class.forName or loading of service providers etc) are not reported.

It also provides an **-jdkinternals** option to find dependencies to any JDK internal APIs that are unsupported and private to JDK implementation (see [Why Developers Should Not Write Programs That Call 'sun' Packages](#)).

Prepare for JDK 9

Most JDK's internal APIs are encapsulated in JDK 9. [JEP 261](#) specifies the critical internal APIs that remain accessible until a replacement API is available in a future release. Other internal APIs are inaccessible by default.

To prepare for JDK 9, download [JDK 9 early-access build](#) and run **jdeps** to find out if your application and libraries depends on any JDK's internal API.

```
$ jdeps -dotoutput <dot-file-dir> -jdkinternals <one-or-more-jar-files...>
```

This **jdeps** command will output the dependencies in DOT file format and one output .dot file per JAR file.

Replace use of JDK's internal APIs

Below lists some of the JDK's internal APIs and the recommended way to replace their usage. See [JEP 261](#) for the `--add-exports` command-line option to break in the encapsulation as a short-term migration purpose

Unsupported API (not for use)	Supported APIs (please use instead)	Note
core-libs		
sun.io	java.nio.charsets @since 1.4	
sun.misc.BASE64Decoder, sun.misc.BASE64Encoder, com.sun.org.apache.xml.internal.security.utils.Base64	java.util.Base64 @since 8	See http:
sun.misc.ClassLoaderUtil	java.net.URLClassLoader.close() @since 7	
sun.misc.Cleaner	java.lang.ref.PhantomReference @since 1.2	JDK-6411 resource mapped l in a timel sun.misc direct byt sun.misc jdk.intern See JDK
sun.misc.Service	java.util.ServiceLoader @since 1.6	
sun.misc.Timer	java.util.Timer @since 1.3	
sun.misc.Unsafe	sun.misc.Unsafe consists of a number of use cases. The following features are identified to provide support in the future releases: <ul style="list-style-type: none">JEP 193 Enhanced VolatileJEP 187 Serialization 2.0Value typesJEP 189 Shenandoah:Low-Pause GCArrays 2.0LayoutsProject PanamaJEP 191 FFI	In progre <ul style="list-style-type: none">JEPJDKcomJDKcom
sun.reflect.Reflection.getCallerClass	java.lang.StackWalker::getCallerClass @since 9	See JDK

sun.util.calendar.ZoneInfo	java.util.TimeZone or java.time API @since 8	
security-libs		
sun.security.action.*	java.security.PrivilegedAction to call System.getProperty or other action @since 1.1	AccessC (Privil -> Syst
sun.security.krb5.*	Some provided in com.sun.security.jgss javax.security.auth.kerberos.EncryptionKey @since 1.9 javax.security.auth.kerberos.KerberosCredMessage @since 1.9 javax.security.auth.kerberos.KerberosTicket.getSessionKey() @since 1.9	If internal session k have Ext purpose. JDK-804:
sun.security.util.SecurityConstants	java.lang.RuntimePermission, java.net.NetPermission, or specific Permission class @since 1.1	
sun.security.util.HostnameChecker	javax.net.ssl.SSLParameters.setEndpointIdentificationAlgorithm("HTTPS" or "LDAPS") can be used to enabled hostname checking during handshaking javax.net.ssl.HttpURLConnection.setHostnameVerifier() can be customized hostname verifier rules for URL operations.	See also the new €
sun.security.x509.*	javax.security.auth.x500.X500Principal @since 1.4	JDK-8051 jdk.secur JDK 9. T generate
com.sun.org.apache.xml.internal.security	javax.xml.crypto @since 1.6	
com.sun.net.ssl.**	javax.net.ssl @since 1.4	
security provider implementation class such as <ul style="list-style-type: none"> com.sun.net.ssl.internal.ssl.Provider sun.security.provider.Sun com.sun.crypto.provider.SunJCE 	java.security.Security.getProvider(NAME) @since 1.3 where NAME is the security provider name such as "SUN", "SunJCE".	In genera on a spec available impleme providers rationale.
sun.security.provider.PolicyFile() or sun.security.provider.PolicyFile(URL)	java.security.Policy.getInstance("JavaPolicy", new java.security.URIPParameter(uri)); @since 1.6	
client-libs		
java.awt.peer and java.awt.dnd.peer	Instead of doing: if (c.getPeer() != null) { .. } could be replaced with: if (c.isDisplayable()) { ... } To test if a component has a LightweightPeer, use: public boolean isLightweight() ; @since 1.2 To obtain the color model of the component comes from the peer, instead of doing: getPanel().getPeer().getColorModel() could be replaced with: public ColorModel getColorModel() ;	java.awt.) types are API refer java.awt.) in JDK 9. ev discus

com.sun.image.codec.jpeg.** sun.awt.image.codec	javax.imageio @since 1.4	See JDK
com.apple.eawt	java.awt.Desktop @since 9	See http://
JDBC		
com.sun.rowset.**	javax.sql.rowset.RowSetProvider @since 7	
JAXP		
org.w3c.dom.{html, css, stylesheets}	org.w3c.dom.{html, css, stylesheets} APIs are JDK supported APIs @since 9.	JDK-804:
org.w3c.dom.xpath	org.w3c.dom.xpath API is now JDK supported API @since 9	JDK-804: JDK-805- resolved
com.sun.org.apache.xml.internal.resolver.**	javax.xml.catalog @since 9	See JDK
org.relaxng.datatype	org.relaxng.** will be repackaged in JDK 9. Users should include the org.relaxng.** types in the classpath.	See JDK
Others		
com.sun.tools.javac.**	javax.tools, javax.lang.model @since 1.6 com.sun.source.* @since 1.6	com.sun. API.
jdk.nashorn.internal.ir.**	JEP 236 Parser API for Nashorn	JDK-804: resolved