

# 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
<b>core-libs</b>		
protected java.lang.ClassLoader::defineClass method	java.lang.invoke.MethodHandles.Lookup::defineClass @since 9	Frameworks may use java.lang.invoke.MethodHandles::privateLookupIn to obtain a Lookup object with the permission to access the private members a target class in a different module if the framework is granted with deep reflection access to the target class.
sun.io	<a href="#">java.nio.charsets</a> @since 1.4	
sun.misc.BASE64Decoder, sun.misc.BASE64Encoder, com.sun.org.apache.xml.internal.security.utils.Base64	<a href="#">java.util.Base64</a> @since 8	See <a href="http://openjdk.java.net/jeps/135">http://openjdk.java.net/jeps/135</a>
sun.misc.ClassLoaderUtil	java.net.URLClassLoader.close() @since 7	
sun.misc.Cleaner	java.lang.ref.PhantomReference @since 1.2	JDK-6417205 may help with the resource issues that can arise when mapped byte buffers are not unmapped in a timely manner. Libraries accessing sun.misc.Cleaner have to be fixed as direct byte buffer no longer uses sun.misc.Cleaner class; instead jdk.internal.misc.Cleaner.  See <a href="#">JDK-6685587</a> and <a href="#">JDK-4724038</a>
sun.misc.Service	java.util.ServiceLoader @since 1.6	
sun.misc.Timer	java.util.Timer @since 1.3	

sun.misc.Unsafe	<p>java.lang.invoke.VarHandle since 9</p> <p>java.lang.invoke.MethodHandles.Lookup::defineClass @since 9</p> <p>java.lang.invoke.MethodHandles.Lookup::defineHiddenClass @since 15</p> <p>java.lang.invoke.MethodHandles.Lookup::ensureInitialized @since 15</p>	<p>sun.misc.Unsafe consists of a number of use cases. The following features are identified to provide support in the future releases:</p> <ul style="list-style-type: none"> <li>• <a href="#">JEP 193: Enhanced Volatile</a></li> <li>• <a href="#">JEP 187: Serialization 2.0</a></li> <li>• <a href="#">JEP 189: Shenandoah:Low-Pause GC</a></li> <li>• <a href="#">Arrays 2.0</a></li> <li>• <a href="#">Project Panama</a></li> <li>• <a href="#">JEP 191: FFI</a></li> <li>• <a href="#">JEP 370: Foreign-Memory Access API (Incubator) &amp; JEP 383: Foreign-Memory Access API (Second Incubator)</a></li> <li>• <a href="#">JEP 371: Hidden Classes</a></li> </ul> <p>See also</p> <ul style="list-style-type: none"> <li>• <a href="#">JDK-8044082</a> Efficient array comparison intrinsics</li> <li>• <a href="#">JDK-8033148</a> Lexicographic comparators for arrays</li> </ul>
sun.reflect.Reflection.getCallerClass	java.lang.StackWalker::getCallerClass @since 9	See <a href="#">JDK-8043814</a> (Stack Walking API)
sun.util.calendar.ZoneInfo	java.util.TimeZone or java.time API @since 8	
<b>security-libs</b>		
sun.security.action.*	java.security.PrivilegedAction to call System.getProperty or other action @since 1.1	<code>AccessController.doPrivileged((PrivilegedAction&lt;String&gt; () -&gt; System.getProperty(key));</code>
sun.security.krb5.*	<p>Some provided in <a href="#">com.sun.security.jgss</a></p> <p>javax.security.auth.kerberos.EncryptionKey @since 1.9</p> <p>javax.security.auth.kerberos.KerberosCredMessage @since 1.9</p> <p>javax.security.auth.kerberos.KerberosTicket.getSessionKey() @since 1.9</p>	<p>If internal classes are used to get the session key of Krb5Context, we now have ExtendedGSSContext for this purpose.</p> <p><a href="#">JDK-8043071</a> resolved in JDK 9 b25</p>
sun.security.util.SecurityConstants	java.lang.RuntimePermission, java.net.NetPermission, or specific Permission class @since 1.1	
sun.security.util.HostnameChecker	<p><a href="#">javax.net.ssl.SSLParameters.setEndpointIdentificationAlgorithm("HTTPS" or "LDAPS")</a> can be used to enable hostname checking during handshaking</p> <p><a href="#">javax.net.ssl.HttpURLConnection.setHostnameVerifier()</a> can be customized hostname verifier rules for URL operations.</p>	See also <a href="#">JDK-7192189</a> RFE to support the new endpoint identification.
sun.security.x509.*	<a href="#">javax.security.auth.x500.X500Principal</a> @since 1.4	<a href="#">JDK-8056174</a> defines <code>jdk.security.jarsigner.JarSigner</code> API in JDK 9. This API can also be used to generate self-signed certificates.
com.sun.org.apache.xml.internal.security	<a href="#">javax.xml.crypto</a> @since 1.6	
com.sun.net.ssl.**	<a href="#">javax.net.ssl</a> @since 1.4	

security provider implementation class such as <ul style="list-style-type: none"> <li>com.sun.net.ssl.internal.ssl.Provider</li> <li>sun.security.provider.Sun</li> <li>com.sun.crypto.provider.SunJCE</li> </ul>	java.security.Security.getProvider (NAME) @since 1.3  where NAME is the security provider name such as "SUN", "SunJCE".	In general, you should avoid depending on a specific provider as it may not be available on other Java implementations. See <a href="#">Oracle security providers documentation</a> for more rationale.
sun.security.provider.PolicyFile() or sun.security.provider.PolicyFile(URL)	java.security.Policy.getInstance ("JavaPolicy", new java.security.URIPParameter(uri)); @since 1.6	
<b>client-libs</b>		
java.awt.peer and java.awt.dnd.peer	<p>Instead of doing:</p> <pre>if (c.getPeer() != null) { .. }</pre> <p>could be replaced with:</p> <pre>if (c.isDisplayable()) { ... }</pre> <p>To test if a component has a LightweightPeer, use:</p> <pre>public boolean isLightweight(); @since 1.2</pre> <p>To obtain the color model of the component comes from the peer, instead of doing:</p> <pre>getPanel().getPeer().getColorModel()</pre> <p>could be replaced with:</p> <pre>public ColorModel getColorModel();</pre>	<p>java.awt.peer.* and java.awt.dnd.peer.* types are encapsulated.</p> <p>API reference to java.awt.peer.* and java.awt.dnd.peer.* types are removed in JDK 9. See <a href="#">JDK-8037739</a> and <a href="#">awt-dev discussion</a></p>
com.sun.image.codec.jpeg.**  sun.awt.image.codec	<a href="#">javax.imageio</a> @since 1.4	See <a href="#">JDK-6527962</a>
com.apple.eawt	java.awt.Desktop @since 9	See <a href="http://openjdk.java.net/jeps/272">http://openjdk.java.net/jeps/272</a>
<b>JDBC</b>		
com.sun.rowset.**	<a href="#">javax.sql.rowset.RowSetProvider</a> @since 7	
<b>JAXP</b>		
org.w3c.dom.{html, css, stylesheets}	org.w3c.dom.{html, css, stylesheets} APIs are JDK supported APIs @since 9.	<a href="#">JDK-8042244</a> resolved in JDK 9 b62
org.w3c.dom.xpath	org.w3c.dom.xpath API is now JDK supported API @since 9	<a href="#">JDK-8042244</a> resolved in JDK 9 b62 <a href="#">JDK-8054196</a> for XPath support any API resolved in JDK 9 b49
com.sun.org.apache.xml.internal.resolver.**	<a href="#">javax.xml.catalog</a> @since 9	See <a href="#">JDK-8023732</a> (XML Catalog API)

org.relaxng.datatype	org.relaxng.** will be repackaged in JDK 9. Users should include the org.relaxng.** types in the classpath.	See <a href="#">JDK-8061466</a>
<b>Others</b>		
com.sun.tools.javac.**	<a href="#">javax.tools</a> , <a href="#">javax.lang.model</a> @since 1.6 <a href="#">com.sun.source.*</a> @since 1.6	com.sun.tools.javac.Main is a supported API.
jdk.nashorn.internal.ir.**	<a href="#">JEP 236</a> Parser API for Nashorn	<a href="#">JDK-8048176</a> (Nashorn Parser API) resolved in JDK 9 b55