

Java Platform, Standard Edition

What's New in Oracle JDK 9

Release 9

E77563-05

September 2017

Overview of What's New in JDK 9

Java Platform, Standard Edition 9 is a major feature release. The following summarizes features and enhancements in Java SE 9 and in JDK 9, Oracle's implementation of Java SE 9.

A JDK Enhancement Proposal (JEP) is a proposal to design and implement a nontrivial change to the JDK. See [JEP 1: JDK Enhancement-Proposal & Roadmap Process](#). A Java Specification Request (JSR) describes proposed and final specifications for the Java platform. See [JSR Overview](#).

Key Changes in JDK 9

These changes affect more than one technology area.

Feature	Description
Java Platform Module System	<p data-bbox="753 289 1446 369">Introduces a new kind of Java programming component, the module, which is a named, self-describing collection of code and data. This module system:</p> <ul data-bbox="753 373 1446 762" style="list-style-type: none"> <li data-bbox="753 373 1446 516">• Introduces a new optional phase, link time, which is in-between compile time and run time, during which a set of modules can be assembled and optimized into a custom runtime image; see the <code>jlink</code> tool in <i>Java Platform, Standard Edition Tools Reference</i>. <li data-bbox="753 520 1446 600">• Adds options to the tools <code>javac</code>, <code>jlink</code>, and <code>java</code> where you can specify module paths, which locate definitions of modules. <li data-bbox="753 604 1446 667">• Introduces the modular JAR file, which is a JAR file with a <code>module-info.class</code> file in its root directory. <li data-bbox="753 672 1446 762">• Introduces the JMOD format, which is a packaging format similar to JAR except it can include native code and configuration files; see the <code>jmod</code> tool. <p data-bbox="753 766 1446 825">The JDK itself has been divided into a set of modules. This change:</p> <ul data-bbox="753 829 1446 1470" style="list-style-type: none"> <li data-bbox="753 829 1446 1045">• Enables you to combine the JDK's modules into a variety of configurations, including: <ul data-bbox="797 898 1446 1045" style="list-style-type: none"> <li data-bbox="797 898 1446 930">– Configurations corresponding to the JRE and the JDK. <li data-bbox="797 934 1446 987">– Configurations roughly equivalent in content to each of the Compact Profiles defined in Java SE 8. <li data-bbox="797 991 1446 1045">– Custom configurations that contain only a specified set of modules and their required modules. <li data-bbox="753 1050 1446 1140">• Restructures the JDK and JRE runtime images to accommodate modules and improve performance, security, and maintainability. <li data-bbox="753 1144 1446 1224">• Defines a new URI scheme for naming modules, classes, and resources stored in a runtime image without revealing the internal structure or format of the image. <li data-bbox="753 1228 1446 1291">• Removes the endorsed-standards override mechanism and the extension mechanism. <li data-bbox="753 1295 1446 1358">• Removes <code>rt.jar</code> and <code>tools.jar</code> from the Java runtime image. <li data-bbox="753 1362 1446 1470">• Makes most of the JDK's internal APIs inaccessible by default but leaves a few critical, widely used internal APIs accessible until supported replacements exist for all or most of their functionality. <p data-bbox="797 1474 1446 1537">Run the command <code>jdeps -jdkinternals</code> to determine if your code uses internal JDK APIs.</p> <p data-bbox="753 1541 1446 1572">For more information, see the following:</p> <ul data-bbox="753 1577 1446 1736" style="list-style-type: none"> <li data-bbox="753 1577 1446 1608">• Java Platform Module System (JSR 376) <li data-bbox="753 1612 1446 1644">• JEP 261: Module System <li data-bbox="753 1648 1446 1680">• JEP 200: The Modular JDK <li data-bbox="753 1684 1446 1715">• JEP 220: Modular Run-Time Images <li data-bbox="753 1719 1446 1736">• JEP 260: Encapsulate Most Internal APIs

Feature	Description
JEP 223: New Version-String Scheme	<p>Provides a simplified version-string format that helps to clearly distinguish major, minor, security, and patch update releases. The new version-string format is as follows:</p> <pre>\$MAJOR.\$MINOR.\$SECURITY.\$PATCH</pre> <ul style="list-style-type: none"> • <code>\$MAJOR</code> is the version number that is incremented for a major release, for example JDK 9, which contains significant new features as specified by the Java SE platform specification. A major release contains new features and changes to existing features, which are planned and announced well in advance. • <code>\$MINOR</code> is the version number that is incremented for each minor update, such as bug fixes, revisions to standard APIs, or implementation of features outside the scope of the relevant platform specifications. • <code>\$SECURITY</code> is the version number that is incremented for a security-update release, which contains critical fixes, including those necessary to improve security. • <code>\$PATCH</code> is the version number that is incremented for a release containing security and high-priority customer fixes that have been tested together. <p>See New Version String Format in <i>Java Platform, Standard Edition Installation Guide</i>.</p>

What's New for the JDK 9 Installer

JDK 9 includes installer enhancements for Microsoft Windows and macOS platforms.

Installer Enhancements for Microsoft Windows

Feature	Description
Enable or Disable Web Deployment with Installer's UI	Provides the option to enable or disable web deployment in the Welcome page of the installer. To enable web deployment, in the Welcome page, select Custom Setup , click Install , and select the Enable Java content in the Browser check box.

Installer Enhancements for macOS

Feature	Description
CPU Version Availability	Provides notification on next CPU availability after uninstalling the current CPU version.
User Experience	Enhanced user experience while updating the JRE.

What's New for Tools in JDK 9

These are the tools enhancements in JDK 9.

Feature	Description
JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)	<p>Adds Read-Eval-Print Loop (REPL) functionality to the Java platform.</p> <p>The <code>jshell</code> tool provides an interactive command-line interface for evaluating declarations, statements, and expressions of the Java programming language. It facilitates prototyping and exploration of coding options with immediate results and feedback. The immediate feedback combined with the ability to start with expressions is useful for education—whether learning the Java language or just learning a new API or language feature.</p> <p>See <code>jshell</code> in <i>Java Platform, Standard Edition Tools Reference</i>, and Introduction to JShell in <i>Java Platform, Standard Edition Java Shell User's Guide</i>.</p> <p>The JShell API enables applications to leverage REPL functionality. See the <code>jdk.jshell</code> package.</p>
JEP 228: Add More Diagnostic Commands	<p>Defines additional diagnostic commands to improve the ability to diagnose issues with Hotspot and the JDK.</p> <p>See <code>jcmd</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 231: Remove Launch-Time JRE Version Selection	<p>Removes the ability to request a version of the JRE that is not the JRE being launched at launch time.</p> <p>Modern applications are typically deployed through Java Web Start (with a JNLP file), native OS packaging systems, or active installers. These technologies have their own methods to manage the JREs needed by finding or downloading and updating the required JRE as needed. This makes launch-time JRE version selection obsolete.</p>
JEP 238: Multi-Release JAR Files	<p>Extends the JAR file format to enable multiple, Java release-specific versions of class files to coexist in a single archive.</p> <p>A multirelease JAR (MRJAR) contains additional, versioned directories for classes and resources specific to particular Java platform releases. Specify versioned directories with the <code>jar</code> tool's <code>--release</code> option.</p>

Feature	Description
JEP 240: Remove the JVM TI hprof Agent	<p>Removes the <code>hprof</code> agent from the JDK. The <code>hprof</code> agent was written as demonstration code for the JVM Tool Interface and not intended to be a production tool.</p> <p>The useful features of the <code>hprof</code> agent have been superseded by better alternatives.</p>
<div style="border-left: 2px solid #0070C0; padding-left: 10px; margin-left: 40px;"> <p> Note:</p> <p>While the <code>hprof</code> agent has been removed, it is still possible to create heap dumps in the <code>hprof</code> format using <code>jmap</code> or other diagnostic tools. See <i>Diagnostic Tools</i> in <i>Java Platform, Standard Edition Troubleshooting Guide</i>.</p> </div>	
JEP 241: Remove the jhat Tool	<p>Removes the <code>jhat</code> tool from the JDK.</p> <p>The <code>jhat</code> tool was an experimental and unsupported tool added in JDK 6. It is out of date; superior heap visualizers and analyzers have been available for many years.</p>
JEP 245: Validate JVM Command-Line Flag Arguments	<p>Validates arguments to all numerical JVM command-line flags to avoid failures and instead displays an appropriate error message if they are found to be invalid.</p> <p>Range and optional constraint checks have been implemented for arguments that require a user-specified numerical value.</p> <p>See <code>java</code> and <i>Validate Java Virtual Machine Flag Arguments</i> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 247: Compile for Older Platform Versions	<p>Enhances <code>javac</code> so that it can compile Java programs to run on selected earlier versions of the platform.</p> <p>When using the <code>-source</code> or <code>-target</code> options, the compiled program might accidentally use APIs that are not supported on the given target platform. The <code>--release</code> option will prevent accidental use of APIs.</p> <p>See <code>javac</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 282: jlink: The Java Linker	<p>Assembles and optimizes a set of modules and their dependencies into a custom runtime image as defined in JEP 220.</p> <p>The <code>jlink</code> tool defines a plug-in mechanism for transformation and optimization during the assembly process, and for the generation of alternative image formats. It can create a custom runtime optimized for a single program. JEP 261 defines <i>link time</i> as an optional phase between the phases of compile time and run time. Link time requires a linking tool that assembles and optimizes a set of modules and their transitive dependencies to create a runtime image or executable.</p> <p>See <code>jlink</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>

What's New for Security in JDK 9

These are the security enhancements in JDK 9.

Feature	Description
JEP 219: Datagram Transport Layer Security (DTLS)	<p>Enables Java Secure Socket Extension (JSSE) API and the SunJSSE security provider to support DTLS Version 1.0 and DTLS Version 1.2 protocols.</p> <p>See Datagram Transport Layer Security (DTLS) in Java Platform, Standard Edition Security Developer's Guide.</p>
JEP 244: TLS Application-Layer Protocol Negotiation Extension	<p>Enables the client and server in a Transport Layer Security (TLS) connection to negotiate the application protocol to be used. With Application-Layer Protocol Negotiation (ALPN), the client sends the list of supported application protocols as part of the TLS ClientHello message. The server chooses a protocol and returns the selected protocol as part of the TLS ServerHello message. The application protocol negotiation can be accomplished within the TLS handshake, without adding network round-trips.</p> <p>See TLS Handshake and Application Layer Protocol Negotiation in Java Platform, Standard Edition Security Developer's Guide.</p>
JEP 249: OCSP Stapling for TLS	<p>Enables the server in a TLS connection to check for a revoked X.509 certificate revocation. The server does this during TLS handshaking by contacting an Online Certificate Status Protocol (OCSP) responder for the certificate in question. It then attaches or "staples" the revocation information to the certificate that it returns to the client so that the client can take appropriate action.</p> <p>Enables the client to request OCSP stapling from a TLS server. The client checks stapled responses from servers that support the feature.</p> <p>See OCSP Stapling in Java Platform, Standard Edition Security Developer's Guide.</p>
JEP 246: Leverage CPU Instructions for GHASH and RSA	<p>Improves performance ranging from 34x to 150x for <code>AES/GCM/NoPadding</code> using GHASH HotSpot intrinsics. GHASH intrinsics are accelerated by the <code>PCLMULQDQ</code> instruction on Intel x64 CPU and the <code>xmul/xmulhi</code> instructions on SPARC.</p> <p>Improves performance up to 50% for <code>BigInteger squareToLen</code> and <code>BigInteger mulAdd</code> methods using RSA HotSpot intrinsics. RSA intrinsics apply to the <code>java.math.BigInteger</code> class on Intel x64.</p> <p>A new security property <code>jdk.security.provider.preferred</code> is introduced to configure providers that offer significant performance gains for specific algorithms.</p> <p>See Configuring the Preferred Provider for Specific Algorithms in Java Platform, Standard Edition Security Developer's Guide.</p>
JEP 273: DRBG-Based SecureRandom Implementations	<p>Provides the functionality of Deterministic Random Bit Generator (DRBG) mechanisms as specified in NIST SP 800-90Ar1 in the SecureRandom API.</p> <p>The DRBG mechanisms use modern algorithms as strong as SHA-512 and AES-256. Each of these mechanisms can be configured with different security strengths and features to match user requirements.</p> <p>See Generating Random Numbers in Java Platform, Standard Edition Security Developer's Guide.</p>

Feature	Description
JEP 288: Disable SHA-1 Certificates	<p>Improves the security configuration of the JDK by providing a more flexible mechanism to disable X.509 certificate chains with SHA-1-based signatures.</p> <p>Disables SHA-1 in TLS Server certificate chains anchored by roots included by default in the JDK; local or enterprise certificate authorities (CAs) are not affected.</p> <p>The <code>jdk.certpath.disabledAlgorithms</code> security property is enhanced with several new constraints that allow greater control over the types of certificates that can be disabled.</p> <p>See JEP 288.</p>
JEP 229: Create PKCS12 Keystores by Default	<p>Modifies the default keystore type from JKS to PKCS12. PKCS#12 is an extensible, standard, and widely supported format for storing cryptographic keys. PKCS12 keystores improve confidentiality by storing private keys, trusted public key certificates, and secret keys. This feature also opens opportunities for interoperability with other systems such as Mozilla, Microsoft's Internet Explorer, and OpenSSL that support PKCS12.</p> <p>The SunJSSE provider supplies a complete implementation of the PKCS12 <code>java.security.KeyStore</code> format for reading and writing PKCS12 files.</p> <p>See <i>Key Management in Java Platform, Standard Edition Security Developer's Guide</i>.</p> <p>The <code>keytool</code> key and certificate management utility can create PKCS12 keystores.</p> <p>See <i>Creating a Keystore in Java Platform, Standard Edition Security Developer's Guide</i> and <code>keytool</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 287: SHA-3 Hash Algorithms	<p>Supports SHA-3 cryptographic hash functions as specified in NIST FIPS 202.</p> <p>The following additional standard algorithms are supported by the <code>java.security.MessageDigest</code> API: SHA3-224, SHA3-256, SHA3-384, and SHA3-512.</p> <p>The following providers support SHA-3 algorithm enhancements:</p> <ul style="list-style-type: none"> • SUN provider: SHA3-224, SHA3-256, SHA3-384, and SHA3-512 • OracleUcrypto provider: SHA-3 digests supported by Solaris 12.0

What's New for Deployment in JDK 9

These are the deployment enhancements in JDK 9.

Feature	Description
Deprecate the Java Plug-in	<p>Deprecates the Java Plug-in and associated applet technologies in Oracle's JDK 9 builds. While still available in JDK 9, these technologies will be considered for removal from the Oracle JDK and JRE in a future release.</p> <p>Applets and JavaFX applications embedded in a web page require the Java Plug-in to run. Consider rewriting these types of applications as Java Web Start or self-contained applications.</p> <p>See <i>Migrating Java Applets to Java Web Start and JNLP and Self-Contained Application Packaging in Java Platform, Standard Edition Deployment Guide</i>.</p>
Enhanced Java Control Panel	<p>Improves the grouping and presentation of options within the Java Control Panel. Information is easier to locate, a search field is available, and modal dialog boxes are no longer used. Note that the location of some options has changed from previous versions of the Java Control Panel.</p> <p>See <i>Java Control Panel in Java Platform, Standard Edition Deployment Guide</i>.</p>
JEP 275: Modular Java Application Packaging	<p>Integrates features from Project Jigsaw into the Java Packager, including module awareness and custom runtime creation.</p> <p>Leverages the <code>jlink</code> tool to create smaller packages.</p> <p>Creates applications that use the JDK 9 runtime only. Cannot be used to package applications with an earlier release of the JRE.</p> <p>See <i>Customization of the JRE and Packaging for Modular Applications in Java Platform, Standard Edition Deployment Guide</i>.</p>
JEP 289: Deprecate the Applet API	<p>Deprecates the Applet API, which is becoming less useful as web browser vendors remove support for Java browser plug-ins. While still available in JDK 9, the <code>Applet</code> class will be considered for removal in a future release. Consider rewriting applets as Java Web Start or self-contained applications.</p> <p>See <i>Migrating Java Applets to Java Web Start and JNLP and Self-Contained Application Packaging in Java Platform, Standard Edition Deployment Guide</i>.</p>

What's New for the Java Language in JDK 9

A few very small language changes are included in Java SE 9.

Feature	Description
JEP 213: Milling Project Coin	<p>Identifies a few small changes:</p> <ul style="list-style-type: none"> • Allow <code>@SafeVargs</code> on private instance methods. • Allow effectively final variables to be used as resources in the <code>try-with-resources</code> statement. • Allow the diamond with anonymous classes if the argument type of the inferred type is denotable. • Complete the removal, begun in Java SE 8, of the underscore from the set of legal identifier names. • Add support for private interface methods. <p>See Java Language Changes for Java SE 9 in <i>Java Platform, Standard Edition Java Language Updates</i>.</p>

What's New for Javadoc in JDK 9

Javadoc enhancements include the following: a simplified Doclet API, Javadoc search, support for generating HTML5 output, and support for documentation comments in module systems.

Feature	Description
JEP 221: Simplified Doclet API	Replaces the old Doclet API with a new simplified API that leverages other standard, existing APIs. The standard doclet has been rewritten to use the new Doclet API .
<div style="border-left: 2px solid #0070C0; padding-left: 10px; margin: 10px 0;">  Note: The existing API and old standard doclet are available, but have not been updated to support new language features, such as modules. </div>	
JEP 224: HTML5 Javadoc	Supports generating HTML5 output. To get fully compliant HTML5 output, ensure that any HTML content provided in documentation comments are compliant with HTML5.
JEP 225: Javadoc Search	Provides a search box to the generated API documentation. Use this search box to find program elements, tagged words, and phrases within the documentation.
JEP 261: Module System	Supports documentation comments in module declarations. Includes new command-line options to configure the set of modules to be documented and generates a new summary page for any modules being documented.

What's New for the JVM in JDK 9

These are the JVM enhancements in JDK 9.

Feature	Description
JEP 165: Compiler Control	<p>Provides a way to control JVM compilation through compiler directive options. The level of control is runtime-manageable and method-specific. Compiler Control supersedes, and is backward compatible, with <code>CompileCommand</code>.</p> <p>See <code>Compiler Control</code> in <i>Java Platform, Standard Edition Java Virtual Machine Guide</i>.</p>
JEP 197: Segmented Code Cache	<p>Divides the code cache into distinct segments, each of which contains compiled code of a particular type, to improve performance and enable future extensions.</p> <p>See <code>java</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 276: Dynamic Linking of Language-Defined Object Models	<p>Dynamically links high-level object operations at run time, such as <code>read a property</code>, <code>write a property</code>, and <code>invoke a function</code>, to the appropriate target method handles. It links these operations to target method handles based on the actual types of the values passed. These object operations are expressed as <code>invokedynamic</code> sites.</p> <p>While <code>java.lang.invoke</code> provides a low-level API for dynamic linking of <code>invokedynamic</code> call sites, it doesn't provide a way to express higher level operations on objects nor methods that implement them.</p> <p>With the package <code>jdk.dynalink</code>, you can implement programming languages whose expressions contain dynamic types (types that cannot be determined statically) and whose operations on these dynamic types are expressed as <code>invokedynamic</code> call sites (because the language's object model or type system doesn't closely match that of the JVM).</p>

What's New for JVM Tuning in JDK 9

These are the JVM tuning enhancements in JDK 9.

Feature	Description
Improve G1 Usability, Determinism, and Performance	<p>Enhances the Garbage-First (G1) garbage collector to automatically determine several important memory-reclamation settings. Previously these settings had to be set manually to obtain optimal results. In addition, fixes issues with the usability, determinism, and performance of the G1 garbage collector.</p>
JEP 158: Unified JVM Logging	<p>Introduces a common logging system for all components of the JVM.</p> <p>See the <code>-Xloggc java</code> option in <i>Java Platform, Standard Edition Tools Reference</i>.</p>

Feature	Description
JEP 214: Remove GC Combinations Deprecated in JDK 8	<p>Removes garbage collector (GC) combinations that were deprecated in JDK 8.</p> <p>This means that the following GC combinations no longer exist:</p> <ul style="list-style-type: none"> • DefNew + CMS • ParNew + SerialOld • Incremental CMS <p>The "foreground" mode for Concurrent Mark Sweep (CMS) has also been removed. The following command-line flags have been removed:</p> <ul style="list-style-type: none"> • <code>-Xincgc</code> • <code>-XX:+CMSIncrementalMode</code> • <code>-XX:+UseCMSCompactAtFullCollection</code> • <code>-XX:+CMSFullGCsBeforeCompaction</code> • <code>-XX:+UseCMSCollectionPassing</code> <p>The command line flag <code>-XX:+UseParNewGC</code> no longer has an effect. ParNew can only be used with CMS and CMS requires ParNew. Thus, the <code>-XX:+UseParNewGC</code> flag has been deprecated and will likely be removed in a future release.</p>
JEP 248: Make G1 the Default Garbage Collector	<p>Makes Garbage-First (G1) the default garbage collector (GC) on 32- and 64-bit server configurations. Using a low-pause collector such as G1 provides a better overall experience, for most users, than a throughput-oriented collector such as the Parallel GC, which was previously the default.</p> <p>See Garbage-First Garbage Collector in <i>Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide</i></p>
JEP 271: Unified GC Logging	<p>Reimplements Garbage Collection (GC) logging using the unified JVM logging framework introduced in JEP 158. GC logging is re-implemented in a manner consistent with the current GC logging format; however, some differences exist between the new and old formats.</p> <p>See Enable Logging with the JVM Unified Logging Framework in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 291: Deprecate the Concurrent Mark Sweep (CMS) Garbage Collector	<p>Deprecates the Concurrent Mark Sweep (CMS) garbage collector. A warning message is issued when it is requested on the command line, using the <code>-XX:+UseConcMarkSweepGC</code> option. The Garbage-First (G1) garbage collector is intended to be a replacement for most uses of CMS.</p>

What's New for Core Libraries in JDK 9

Feature	Description
JEP 102: Process API Updates	<p>Improves the API for controlling and managing operating system processes. The <code>ProcessHandle</code> class provides the process's native process ID, arguments, command, start time, accumulated CPU time, user, parent process, and descendants. The class can also monitor processes' liveness and destroy processes. With the <code>ProcessHandle.onExit</code> method, the asynchronous mechanisms of the <code>CompletableFuture</code> class can perform an action when the process exits.</p> <p>See Process API in <i>Java Platform, Standard Edition Java Core Libraries Developer's Guide</i>, java.lang.Process, and java.lang.ProcessHandle.</p>
JEP 193: Variable Handles	<p>Defines a standard means to invoke the equivalents of <code>java.util.concurrent.atomic</code> and <code>sun.misc.Unsafe</code> operations upon object fields and array elements.</p> <p>Defines a standard set of fence operations, which consist of <code>VarHandle</code> static methods that enable fine-grained control of memory ordering. This is an alternative to <code>sun.misc.Unsafe</code>, which provides a nonstandard set of fence operations.</p> <p>Defines a standard reachability fence operation to ensure that a referenced object remains strongly reachable.</p>
JEP 254: Compact Strings	<p>Adopts a more space-efficient internal representation for strings. Previously, the <code>String</code> class stored characters in a <code>char</code> array, using two bytes (16 bits) for each character. The new internal representation of the <code>String</code> class is a byte array plus an encoding-flag field.</p> <p>This is purely an implementation change, with no changes to existing public interfaces.</p> <p>See the <code>CompactStrings</code> option of the <code>java</code> command in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 264: Platform Logging API and Service	<p>Defines a minimal logging API that platform classes can use to log messages, together with a service interface for consumers of those messages. A library or application can provide an implementation of this service to route platform log messages to the logging framework of its choice. If no implementation is provided, then a default implementation based on the <code>java.util.logging</code> API is used.</p>
JEP 266: More Concurrency Updates	<p>Adds further concurrency updates to those introduced in JDK 8 in JEP 155: Concurrency Updates, including an interoperable publish-subscribe framework and enhancements to the <code>CompletableFuture</code> API.</p>
JEP 268: XML Catalogs	<p>Adds a standard XML Catalog API that supports the Organization for the Advancement of Structured Information Standards (OASIS) XML Catalogs version 1.1 standard. The API defines catalog and catalog-resolver abstractions that can be used as an intrinsic or external resolver with the JAXP processors that accept resolvers.</p> <p>Existing libraries or applications that use the internal catalog API will need to migrate to the new API to take advantage of the new features.</p> <p>See XML Catalog API in <i>Java Platform, Standard Edition Java Core Libraries Developer's Guide</i>.</p>

Feature	Description
JEP 269: Convenience Factory Methods for Collections	<p>Makes it easier to create instances of collections and maps with small numbers of elements. New static factory methods on the <code>List</code>, <code>Set</code>, and <code>Map</code> interfaces make it simpler to create immutable instances of those collections.</p> <p>For example:</p> <pre>Set<String> alphabet = Set.of("a", "b", "c");</pre> <p>See <i>Creating Immutable Lists, Sets, and Maps in Java Platform, Standard Edition Java Core Libraries Developer's Guide</i>. For API documentation, see Immutable Set Static Factory Methods, Immutable Map Static Factory Methods, and Immutable List Static Factory Methods.</p>
JEP 274: Enhanced Method Handles	<p>Enhances the <code>MethodHandle</code>, <code>MethodHandles</code>, and <code>MethodHandles.Lookup</code> classes of the <code>java.lang.invoke</code> package to ease common use cases and enable better compiler optimizations.</p> <p>Additions include:</p> <ul style="list-style-type: none"> • In the <code>MethodHandles</code> class in the <code>java.lang.invoke</code> package, provide new <code>MethodHandle</code> combinators for loops and <code>try/finally</code> blocks. • Enhance the <code>MethodHandle</code> and <code>MethodHandles</code> classes with new <code>MethodHandle</code> combinators for argument handling. • Implement new lookups for interface methods and, optionally, super constructors in the <code>MethodHandles.Lookup</code> class.
JEP 277: Enhanced Deprecation	<p>Revamps the <code>@Deprecated</code> annotation to provide better information about the status and intended disposition of an API in the specification. Two new elements have been added:</p> <ul style="list-style-type: none"> • <code>@Deprecated(forRemoval=true)</code> indicates that the API will be removed in a future release of the Java SE platform. • <code>@Deprecated(since="version")</code> contains the Java SE version string that indicates when the API element was deprecated, for those deprecated in Java SE 9 and beyond. <p>For example: <code>@Deprecated(since="9", forRemoval=true)</code></p> <p><code>@Deprecated</code> annotations in the core platform have been updated.</p> <p>You can use a new tool, <code>jdprscan</code>, to scan a class library (JAR file) for uses of deprecated JDK API elements.</p> <p>See <i>Enhanced Deprecation in Java Platform, Standard Edition Java Core Libraries Developer's Guide</i>.</p> <p>See <code>jdprscan</code> in <i>Java Platform, Standard Edition Tools Reference</i>.</p>
JEP 285: Spin-Wait Hints	<p>Defines an API that enables Java code to hint that a spin loop is executing. A spin loop repeatedly checks to see if a condition is true, such as when a lock can be acquired, after which some computation can be safely performed followed by the release of the lock. This API is purely a hint, and carries no semantic behavior requirements. See the method Thread.onSpinWait.</p>

Feature	Description
JEP 290: Filter Incoming Serialization Data	<p>Allows incoming streams of object-serialization data to be filtered to improve both security and robustness. Object-serialization clients can validate their input more easily, and exported Remote Method Invocation (RMI) objects can validate invocation arguments more easily as well.</p> <p>Serialization clients implement a filter interface that is set on an <code>ObjectInputStream</code>. For RMI, the object is exported through a <code>RemoteServerRef</code> that sets the filter on the <code>MarshalInputStream</code> to validate the invocation arguments as they are unmarshalled.</p>
JEP 259: Stack-Walking API	<p>Provides a stack-walking API that allows easy filtering and lazy access to the information in stack traces.</p> <p>The API supports both short walks that stop at a frame that matches given criteria, and long walks that traverse the entire stack. Stopping at a frame that matches a given criteria avoids the cost of examining all the frames if the caller is interested only in the top frames on the stack. The API enables access to Class objects when the stack walker is configured to do so. See the class <code>java.lang.StackWalker</code>.</p>
JEP 255: Merge Selected Xerces 2.11.0 Updates into JAXP	<p>Updates the JDK to support the 2.11.0 version of the Xerces parser. There is no change to the public JAXP API.</p> <p>The changes are in the following categories of Xerces 2.11.0: Datatypes, DOM L3 Serializer, XPointer, Catalog Resolver, and XML Schema Validation (including bug fixes, but not the XML Schema 1.1 development code).</p>

What's New for Nashorn in JDK 9

These are the Nashorn enhancements in JDK 9.

Feature	Description
JEP 236: Parser API for Nashorn	<p>Enables applications, in particular IDEs and server-side frameworks, to parse and analyze ECMAScript code.</p> <p>Parse ECMAScript code from a string, URL, or file with methods from the <code>Parser</code> class. These methods return an instance of <code>CompilationUnitTree</code>, which represents ECMAScript code as an abstract syntax tree.</p> <p>The package <code>jdk.nashorn.api.tree</code> contains the Nashorn parser API.</p>
JEP 292: Implement Selected ECMAScript 6 Features in Nashorn	<p>Implements many new features introduced in the 6th edition of ECMA-262, also known as ECMAScript 6, or ES6 for short. Implemented features include the following:</p> <ul style="list-style-type: none"> • Template strings • <code>let</code>, <code>const</code>, and block scope • Iterators and <code>for...of</code> loops • <code>Map</code>, <code>Set</code>, <code>WeakMap</code>, and <code>WeakSet</code> • Symbols • Binary and octal literals

What's New for Client Technologies in JDK 9

These are the client technologies enhancements in JDK 9.

Feature	Description
JEP 251: Multi-Resolution Images	<p>Enables a set of images with different resolutions to be encapsulated into a single multiresolution image. This could be useful for applications to adapt to display devices whose resolutions may vary from approximately 96dpi to 300dpi during run time.</p> <p>The interface <code>java.awt.image.MultiResolutionImage</code> encapsulates a set of images with different resolutions into a single multiresolution image, which enables applications to easily manipulate and display images with resolution variants.</p>
JEP 253: Prepare JavaFX UI Controls and CSS APIs for Modularization	<p>Provides public APIs for JavaFX UI controls and CSS functionality that were previously available only through internal packages but are now inaccessible due to modularization.</p> <p>The new package <code>javafx.scene.control.skin</code> consists of a set of classes that provides a default implementation for the skin (or the look) of each UI control.</p> <p>The new class <code>CssParser</code> is a CSS parser that returns a <code>Stylesheet</code> object, which gives you more control over the CSS styling of your application. It's part of the CSS API (the <code>javafx.css</code> package). The CSS API includes new support classes, including a set of standard converters used by the parser; see the <code>javafx.css.converter</code> package.</p>
JEP 256: BeanInfo Annotations	<p>Replaces the <code>@beaninfo</code> Javadoc tag with the annotation types <code>JavaBean</code>, <code>BeanProperty</code>, and <code>SwingContainer</code>.</p> <p>These annotation types set the corresponding feature attributes during <code>BeanInfo</code> generation at runtime. Thus, you can more easily specify these attributes directly in Bean classes instead of creating a separate <code>BeanInfo</code> class for every Bean class. It also enables the removal of automatically generated classes, which makes it easier to modularize the client library.</p>
JEP 262: TIFF Image I/O	<p>Adds Tag Image File Format (TIFF) reading and writing as standard to the package <code>javax.imageio</code>. The new package <code>javax.imageio.plugins.tiff</code> provides classes that simplify the optional manipulation of TIFF metadata.</p>
JEP 263: HiDPI Graphics on Windows and Linux	<p>Automatically scales and sizes AWT and Swing components for High Dots Per Inch (HiDPI) displays on Windows and Linux.</p> <p>The JDK already supports HiDPI "retina displays" on OS X.</p> <p>Prior to this release, on Windows and Linux, Java applications were sized and rendered based on pixels, even on HiDPI displays that can have pixel densities two to three times as high as traditional displays. This led to GUI components and windows that were too small to read or use.</p>

Feature	Description
JEP 272: Platform-Specific Desktop Features	<p>Adds additional methods to the class <code>java.awt.Desktop</code> that enable you to interact with the desktop, including the following:</p> <ul style="list-style-type: none"> • Show custom About and Preferences windows. • Handle requests to open or print a list of files. • Handle requests to open a URL. • Open the native help viewer application. • Set the default menu bar. • Enable or disable the application to be suddenly terminated. <p>These new methods replace the functionality of the internal APIs contained in the OS X package <code>com.apple.eawt</code>, which are not accessible by default in JDK 9. Note that the package <code>com.apple.eio</code> is no longer accessible.</p>
JEP 283: Enable GTK 3 on Linux	<p>Enables Java graphical applications, whether based on JavaFX, Swing, or Abstract Window Toolkit (AWT), to use either the GTK+ version 2 or version 3 on Linux or Solaris.</p> <p>By default, the JDK on Linux or Solaris uses GTK+ 2 if available; if not, it uses GTK+ 3.</p> <p>To use a specific version of GTK+, set the system property <code>jdk.gtk.version</code>. This system property may have a value of 2, 2.2, or 3. You must set this property before your application loads GTK+, and it must not conflict with a GTK+ version that may have been loaded earlier by another toolkit.</p>

What's New for Internationalization in JDK 9

These are the internationalization enhancements in JDK 9.

Feature	Description
JEP 267: Unicode 8.0	<p>Supports Unicode 8.0. JDK 8 supported Unicode 6.2.</p> <p>The Unicode 6.3, 7.0 and 8.0 standards combined introduced 10,555 characters, 29 scripts, and 42 blocks, all of which are supported in JDK 9.</p>
JEP 252: CLDR Locale Data Enabled by Default	<p>Uses the Common Locale Data Repository's (CLDR) XML-based locale data, first added in JDK 8, as the default locale data in JDK 9. In previous releases, the default was JRE.</p> <p>To enable behavior compatible with JDK 8, set the system property <code>java.locale.providers</code> to a value with <code>COMPAT</code> ahead of <code>CLDR</code>.</p> <p>See CLDR Locale Data Enabled by Default in <i>Java Platform, Standard Edition Internationalization Guide</i>.</p>

Feature	Description
JEP 226: UTF-8 Properties Files	<p>Loads properties files in UTF-8 encoding. In previous releases, ISO-8859-1 encoding was used when loading property resource bundles. UTF-8 is a much more convenient way to represent non-Latin characters.</p> <p>Most existing properties files should not be affected.</p> <p>See UTF-8 Properties Files in <i>Java Platform, Standard Edition Internationalization Guide</i>.</p>

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Java Platform, Standard Edition What's New in Oracle JDK 9, Release 9
E77563-05

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.