Red Hat build of OpenJDK 11

Release notes for Red Hat build of OpenJDK 11.0.9

Last Updated: 2024-05-09

Legal Notice

Abstract

This document provides an overview of new features in Red Hat build of OpenJDK 11, as well as a list of potential known issues and possible workarounds.

Table of Contents

PREFACE

Open Java Development Kit (OpenJDK) is a free and open source implementation of the Java Platform, Standard Edition (Java SE). The Red Hat build of OpenJDK is available in two versions, Red Hat build of OpenJDK 8u and Red Hat build of OpenJDK 11u.

Packages for the Red Hat build of OpenJDK are made available on Red Hat Enterprise Linux and Microsoft Windows and shipped as a JDK and JRE in the Red Hat Container Catalog.

PROVIDING FEEDBACK ON RED HAT BUILD OF OPENJDK DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to create a ticket.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. SUPPORT POLICY FOR RED HAT BUILD OF OPENJDK

Red Hat will support select major versions of Red Hat build of OpenJDK in its products. For consistency, these are the same versions that Oracle designates as long-term support (LTS) for the Oracle JDK.

A major version of Red Hat build of OpenJDK will be supported for a minimum of six years from the time that version is first introduced. For more information, see the OpenJDK Life Cycle and Support Policy .

NOTE

RHEL 6 reached the end of life in November 2020. Because of this, Red Hat build of OpenJDK is not supporting RHEL 6 as a supported configuration.

CHAPTER 2. DIFFERENCES FROM UPSTREAM OPENJDK 11

Red Hat build of OpenJDK in Red Hat Enterprise Linux (RHEL) contains a number of structural changes from the upstream distribution of OpenJDK. The Microsoft Windows version of Red Hat build of OpenJDK attempts to follow RHEL updates as closely as possible.

The following list details the most notable Red Hat build of OpenJDK 11 changes:

- FIPS support. Red Hat build of OpenJDK 11 automatically detects whether RHEL is in FIPS mode and automatically configures Red Hat build of OpenJDK 11 to operate in that mode. This change does not apply to Red Hat build of OpenJDK builds for Microsoft Windows.
- Cryptographic policy support. Red Hat build of OpenJDK 11 obtains the list of enabled cryptographic algorithms and key size constraints from RHEL. These configuration components are used by the Transport Layer Security (TLS) encryption protocol, the certificate path validation, and any signed JARs. You can set different security profiles to balance safety and compatibility. This change does not apply to Red Hat build of OpenJDK builds for Microsoft Windows.
- Red Hat build of OpenJDK on RHEL dynamically links against native libraries such as **zlib** for archive format support and **libjpeg-turbo**, **libpng**, and **giflib** for image support. RHEL also dynamically links against **Harfbuzz** and **Freetype** for font rendering and management.
- The **src.zip** file includes the source for all the JAR libraries shipped with Red Hat build of OpenJDK.
- Red Hat build of OpenJDK on RHEL uses system-wide timezone data files as a source for timezone information.
- Red Hat build of OpenJDK on RHEL uses system-wide CA certificates.
- Red Hat build of OpenJDK on Microsoft Windows includes the latest available timezone data from RHEL.
- Red Hat build of OpenJDK on Microsoft Windows uses the latest available CA certificate from RHEL.

Additional resources

- For more information about detecting if a system is in FIPS mode, see the Improve system FIPS detection example on the Red Hat RHEL Planning Jira.
- For more information about cryptographic policies, see Using system-wide cryptographic policies.

CHAPTER 3. RED HAT BUILD OF OPENJDK FEATURES

3.1. NEW FEATURES AND ENHANCEMENTS

This section describes the new features introduced in this release. It also contains information about changes in the existing features.

> **NOTE**
>
> For all the other changes and security fixes, see https://mail.openjdk.java.net/pipermail/jdk-updates-dev/2020-October/004007.html.

3.1.1. Modified the MS950 charset encoder's conversion table

Some of the one-way byte-to-char mappings have been aligned with the preferred mappings provided by the Unicode Consortium.

For more information, see JDK-8240196.

3.1.2. Allow SunPKCS11 initialization with NSS when external FIPS modules are present in the Security Modules Database

The SunPKCS11 security provider can now be initialized with NSS when FIPS-enabled external modules are configured in the Security Modules Database (NSSDB). Prior to this change, the SunPKCS11 provider would throw a RuntimeException with the message: "FIPS flag set for non-internal module" when such a library was configured for NSS in non-FIPS mode.

This change allows the Red Hat build of OpenJDK to work properly with recent NSS releases in GNU/Linux operating systems when the system-wide FIPS policy is turned on.

For more information, see JDK-8240191.

3.1.3. Localized time zone name inconsistency between English and other locales

English time zone names provided by the CLDR locale provider are now correctly synthesized following the CLDR spec, rather than substituted from the COMPAT provider.

For example, SHORT style names are no longer synthesized abbreviations of LONG style names, but instead produce GMT offset formats.

For more information, see JDK-8238914.

3.1.4. OperatingSystemMXBean methods inside a container return container specific data

When executing in a container, or other virtualized operating environment, the following **OperatingSystemMXBean** methods return container-specific information, if available. Otherwise, it returns the following host-specific data:

- getFreePhysicalMemorySize()
- getTotalPhysicalMemorySize()

- getFreeSwapSpaceSize()
- getTotalSwapSpaceSize()
- getSystemCpuLoad()

For more information, see JDK-8236876.

3.1.5. Added entrust root certification authority - G4 certificate

The entrust root certificate has been added to the cacerts truststore:

- Alias Name: entrustrootcag4
  Distinguished Name: CN=Entrust Root Certification Authority - G4, OU="(c) 2015 Entrust, Inc. - for authorized use only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US

For more information, see JDK-8250756.

3.1.6. Added 3 SSL Corporation root CA certificates

The following root certificates have been added to the cacerts truststore for the SSL Corporation:

- Alias Name: sslrootrsaca
  Distinguished Name: CN=SSL.com Root Certification Authority RSA, O=SSL Corporation, L=Houston, ST=Texas, C=US
- Alias Name: sslrootevrsaca
  Distinguished Name: CN=SSL.com EV Root Certification Authority RSA R2, O=SSL Corporation, L=Houston, ST=Texas, C=US
- Alias Name: sslrooteccca
  Distinguished Name: CN=SSL.com Root Certification Authority ECC, O=SSL Corporation, L=Houston, ST=Texas, C=US

For more information, see JDK-8250860.

3.1.7. Tools updated to warn users if weak algorithms are used before restricting them

The **keytool** and **jarsigner** tools have been updated to warn users about weak cryptographic algorithms being used before they are disabled. The tools will issue warnings for the SHA-1 hash algorithm and 1024-bit RSA/DSA keys.

For more information, see JDK-8244286.

3.1.8. New system properties to configure the TLS signature scheme

Two new system properties have been added to customize the TLS signature schemes in Red Hat build of OpenJDK. **jdk.tls.client.SignatureSchemes** has been added for the TLS client side, and **jdk.tls.server.SignatureSchemes** has been added for the server side.

Each system property contains a comma-separated list of supported signature scheme names specifying the signature schemes that could be used for the TLS connections.

The names are described in the "Signature Schemes" section of the **Java Security Standard Algorithm Names Specification**.

For more information, see JDK-8242147.

3.1.9. Support for canonicalize in krb5.conf

The 'canonicalize' flag in the krb5.conf file is now supported by the JDK Kerberos implementation. When set to **true**, RFC 6806 name canonicalization is requested by clients in TGT requests to KDC services (AS protocol). Otherwise, by default, it is not requested.

The new default behavior is different from previous releases where name canonicalization was always requested by clients in TGT requests to KDC services (provided that support for RFC 6806[1] was not explicitly disabled with the **sun.security.krb5.disableReferrals** system or security properties).

For more information, see JDK-8242059.

3.2. DEPRECATED FEATURES

3.2.1. Weak named curves in TLS, CertPath, and Signed JAR disabled by default

Weak named curves are disabled by default by adding them to the following **disabledAlgorithms** security properties:

- jdk.tls.disabledAlgorithms
- jdk.certpath.disabledAlgorithms
- jdk.jar.disabledAlgorithms

Red Hat has always removed many of the curves provided by upstream, so the only curve disabled in this release is:

- secp256k1

The following curves are still enabled:

- secp256r1
- secp384r1
- secp521r1
- X25519
- X448

When large numbers of weak named curves need to be disabled, adding individual named curves to each **disabledAlgorithms** property would be overwhelming. To relieve this, a new security property, **jdk.disabled.namedCurves**, is implemented that can list the named curves common to all of the **disabledAlgorithms** properties. To use the new property in the **disabledAlgorithms** properties, precede the full property name with the keyword **include**. Users can still add individual named curves to **disabledAlgorithms** properties separate from this new property. No other properties can be included in the **disabledAlgorithms** properties.

To restore the named curves, remove the **include jdk.disabled.namedCurves** either from specific or from all **disabledAlgorithms** security properties. To restore one or more curves, remove the specific named curve(s) from the **jdk.disabled.namedCurves** property.

For more information, see JDK-8236730.

3.2.2. US/Pacific-New zone name removed as part of tzdata2020b

The following JDK's update to tzdata2020b, the long-obsolete files pacificnew and systemv have been removed. As a result, the "US/Pacific-New" zone name declared in the pacificnew data file is no longer available for use.

Information regarding the update can be viewed at https://mm.icann.org/pipermail/tz-announce/2020-October/000059.html

For more information, see JDK-8254177.

CHAPTER 4. ADVISORIES RELATED TO THIS RELEASE

The following advisories have been issued to bugfixes and CVE fixes included in this release.

- RHSA-2020:4316
- RHSA-2020:4305
- RHSA-2020:4306
- RHSA-2020:4307

Revised on 2024-05-09 16:47:57 UTC