Red Hat build of OpenJDK 17

Release notes for Red Hat build of OpenJDK 17.0.5

Red Hat build of OpenJDK 17 Release notes for Red Hat build of OpenJDK 17.0.5

Legal Notice

Abstract

The Release notes for Red Hat build of OpenJDK 17.0.5 document provides an overview of new features in Red Hat build of OpenJDK 17, and 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 three versions: 8u, 11u, and 17u.

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 Ecosystem 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 versions remain similar to Oracle JDK versions that are designated as long-term support (LTS).

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 .
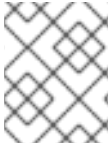
NOTE

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 17

Red Hat build of OpenJDK in Red Hat Enterprise Linux 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 Red Hat Enterprise Linux updates as closely as possible.

The following list details the most notable Red Hat build of OpenJDK 17 changes:

- FIPS support. Red Hat build of OpenJDK 17 automatically detects whether RHEL is in FIPS mode and automatically configures Red Hat build of OpenJDK 17 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 17 obtains the list of enabled cryptographic algorithms and key size constraints from the RHEL system configuration. 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. This change does not apply to Red Hat build of OpenJDK builds for Microsoft Windows.

- The **src.zip** file includes the source for all of 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

- See, Improve system FIPS detection (RHEL Planning Jira)

- See, Using system-wide cryptographic policies (RHEL documentation)

CHAPTER 3. RED HAT BUILD OF OPENJDK FEATURES

The latest Red Hat build of OpenJDK 17 release might include new features. Additionally, the latest release might enhance, deprecate, or remove features that originated from previous Red Hat build of OpenJDK 17 releases.
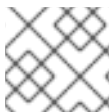
> **NOTE**
>
> For all the other changes and security fixes, see OpenJDK 17.0.5 Released.

Red Hat build of OpenJDK enhancements

Red Hat build of OpenJDK 17 provides enhancements to features originally created in previous releases of Red Hat build of OpenJDK.

Disabled `cpu.shares` parameter

Before the Red Hat build of OpenJDK 17.0.5 release, Red Hat build of OpenJDK used an incorrect interpretation of the **cpu.shares** parameter, which belongs to Linux control groups, also known as **cgroups**. The parameter might cause a Java Virtual machine (JVM) to use fewer CPUs than available, which can impact the JVM's CPU resources and performance when it operates inside a container.

The Red Hat build of OpenJDK 17.0.5 release configures a JVM to no longer use the **cpu.shares** parameter when determining the number of threads for a thread pool. If you want to revert this configuration, pass the **-XX:+UseContainerCpuShares** argument on JVM startup.

> **NOTE**
>
> The **-XX:+UseContainerCpuShares** argument is a deprecated feature and might be removed in a future Red Hat build of OpenJDK release.

See JDK-8281181 (JDK Bug System).

SHA-1 Signed JARs

With the Red Hat build of OpenJDK 17.0.5 release, JARs signed with **SHA-1** algorithms are restricted by default and treated as if they were unsigned. These restrictions apply to the following algorithms:

- Algorithms used to digest, sign, and optionally timestamp the JAR.

- Signature and digest algorithms of the certificates in the certificate chain of the code signer and the Timestamp Authority, and any Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP) responses that are used to verify if those certificates have been revoked.

Additionally, the restrictions apply to signed Java Cryptography Extension (JCE) providers.

To reduce the compatibility risk for JARs that have been previously timestamped, the restriction does not apply to any JAR signed with **SHA-1** algorithms and timestamped prior to **January 01, 2019**. This exception might be removed in a future Red Hat build of OpenJDK release.

To determine if your JAR file is impacted by the restriction, you can issue the following command in your CLI:

```
$ jarsigner -verify -verbose -certs
```

From the output of the previous command, search for instance of **SHA1** , **SHA-1**, or **disabled**. Additionally, search for any warning messages that indicate that the JAR will be treated as unsigned. For example:

> Signed by "CN="Signer""
> Digest algorithm: SHA-1 (disabled)
> Signature algorithm: SHA1withRSA (disabled), 2048-bit key
>
> WARNING: The jar will be treated as unsigned, because it is signed with a weak algorithm that is now disabled by the security property:
>
> jdk.jar.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, DSA keySize < 1024, SHA1 denyAfter 2019-01-01

Consider replacing or re-signing any JARs affected by the new restrictions with stronger algorithms.

If your JAR file is impacted by this restriction, you can remove the algorithm and re-sign the file with a stronger algorithm, such as **SHA-256**. If you want to remove the restriction on **SHA-1** signed JARs for Red Hat build of OpenJDK 17.0.5, and you accept the security risks, you can complete the following actions:

1. Modify the **java.security** configuration file. Alternatively, you can preserve this file and instead create another file with the required configurations.

2. Remove the **SHA1 usage SignedJAR & denyAfter 2019 01 011** entry from the **jdk.certpath.disabledAlgorithms** security property.

3. Remove the **SHA1 denyAfter 2019-01-01** entry from the **jdk.jar.disabledAlgorithms** security property.

> **NOTE**
>
> The value of **jdk.certpath.disabledAlgorithms** in the **java.security** file might be overridden by the system security policy on RHEL 8 and 9. The values used by the system security policy can be seen in the file **/etc/crypto-policies/back-ends/java.config** and disabled by either setting **security.useSystemPropertiesFile** to false in the java.security file or passing **-Djava.security.disableSystemPropertiesFile=true** to the JVM. These values are not modified by this release, so the values remain the same for previous releases of Red Hat build of OpenJDK.
>
> For an example of configuring the **java.security** file, see Overriding **java.security** properties for JBoss EAP for OpenShift (Red Hat Customer Portal).

See JDK-8269039 (JDK Bug System).

SunMSCAPI provider supports new Microsoft Windows keystore types

The **SunMSCAPI** provider supports the following Microsoft Windows keystore types where you must append your local namespace to *Windows-*:

- **Windows-<local_computer_name>**

- **Windows-<root_local_computer_name>**

- **Windows-<current_username>**

- **Windows-<root_username>**

By specifying any of these types, you can provide access to your local computer's location for the Microsoft Windows keystore. Thereby providing the keystore access to certificates that are stored on your local system.

See JDK-6782021 (JDK Bug System).

System properties for controlling the keep-alive behavior of HTTPURLConnection

The Red Hat build of OpenJDK 17.0.5 release includes the following new system properties that you can use to control the **keep-alive** behavior of **HTTPURLConnection**:

- **http.keepAlive.time.server**, which controls connections to servers.

- **http.keepAlive.time.proxy**, which controls connections to proxies.

Before the Red Hat build of OpenJDK 17.0.5 release, a server or a proxy with an unspecified **keep-alive** time might cause an idle connection to remain open for a period defined by a hard-coded default value.

With Red Hat build of OpenJDK 17.0.5, you can use system properties to change the default value for the **keep-alive** time. The **keep-alive** properties control this behavior by changing the HTTP **keep-alive** time of either a server or proxy, so that Red Hat build of OpenJDK's HTTP protocol handler closes idle connections after a specified number of seconds.

Before the Red Hat build of OpenJDK 17.0.5 release, the following use cases would lead to specific **keep-alive** behaviors for **HTTPURLConnection**:

- If the server specifies the **Connection:keep-alive** header and the server's response contains **Keep-alive:timeout=N** then the Red Hat build of OpenJDK **keep-alive** cache on the client uses a timeout of **N** seconds, where **N** is an integer value.

- If the server specifies the **Connection:keep-alive** header, but the server's response does not contain an entry for **Keep-alive:timeout=N** then the Red Hat build of OpenJDK **keep-alive** cache on the client uses a timeout of **60** seconds for a proxy and **5** seconds for a server.

- If the server does not specify the **Connection:keep-alive** header, the Red Hat build of OpenJDK **keep-alive** cache on the client uses a timeout of 5 seconds for all connections.

The Red Hat build of OpenJDK 17.0.5 release maintains the previously described behavior, but you can now specify the timeouts in the second and third listed use cases by using the **http.keepAlive.time.server** and **http.keepAlive.time.proxy** properties, rather than having to rely on the default settings.

NOTE

If you set the **keep-alive** property and the server specifies a **keep-alive** time for the **Keep-Alive** response header, the HTTP protocol handler uses the time specified by the server. This situation is identical for a proxy.

See JDK-8278067 (JDK Bug System).

CHAPTER 4. ADVISORIES RELATED TO THIS RELEASE

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

- RHSA-2022:6999

- RHSA-2022:7000

- RHSA-2022:7001

- RHSA-2022:7051

- RHSA-2022:7054

Revised on 2024-05-03 15:36:36 UTC