Red Hat build of OpenJDK 11

Release notes for Red Hat build of OpenJDK 11.0.17

Last Updated: 2024-05-09

Red Hat build of OpenJDK 11 Release notes for Red Hat build of OpenJDK 11.0.17

Legal Notice

Abstract

The Release notes for Red Hat build of OpenJDK 11.0.17 document provides an overview of new features in Red Hat build of OpenJDK 11 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.

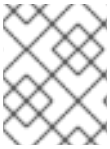
MAKING OPEN SOURCE MORE INCLUSIVE

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 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:

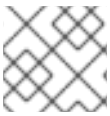
- FIPS support. Red Hat build of OpenJDK 11 automatically detects whether RHEL is in FIPS mode and automatically configures Red Hat build of OpenJDK 11 to operate in that mode. This change does not apply to Red Hat build of OpenJDK builds for Microsoft Windows.
- Cryptographic policy support. Red Hat 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

The latest Red Hat build of OpenJDK 11 release might include new features. Additionally, the latest release might enhance, deprecate, or remove features that originated from previous Red Hat build of OpenJDK 11 releases.



NOTE

For all the other changes and security fixes, see OpenJDK 11.0.17 Released.

Red Hat build of OpenJDK new features and enhancements

Review the following release notes to understand new features and feature enhancements that are included with the Red Hat build of OpenJDK 11.0.17 release:

Disabled cpu.shares parameter

Before the Red Hat build of OpenJDK 11.0.17 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 11.0.17 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).

jdk.httpserver.maxConnections system property

Red Hat build of OpenJDK 11.0.17 adds a new system property, **jdk.httpserver.maxConnections**, that fixes a security issue where no connection limits were specified for the **HttpServer** service, which can cause accepted connections and established connections to remain open indefinitely.

You can use the **jdk.httpserver.maxConnections** system property to change the **HttpServer** service, behavior in the following ways:

- Set a value of **0** or a negative value, such as **-1**, to specify no connection limit for the service.
- Set a positive value, such as **1**, to cause the service to check any accepted connection against the current count of established connections. If the established connection for the service is reached, the service immediately closes the accepted connection.

See JDK-8286918 (JDK Bug System).

Monitor deserialization of objects with JFR

You can now monitor deserialization of objects with the JDK Flight Recorder (JFR). By default, Red Hat build of OpenJDK 11.0.17 disables the **jdk.deserialization** event setting for JFR. You can enable this feature by updating the **event-name** element in your JFR configuration. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration version="2.0" description="test">
```

```
<event name="jdk.Deserialization">
<setting name="enabled">true</setting>
<setting name="stackTrace">false</setting>
</event>
</configuration>
```

After you enable JFR and you configure JFR to monitor deserialization events, JFR creates an event whenever a monitored application attempts to deserialize an object. The serialization filter mechanism of JFR can then determine whether to accept or reject a deserialized object from the monitored application.

See JDK-8261160 (JDK Bug System).

SHA-1 Signed JARs

With the Red Hat build of OpenJDK 11.0.17 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
```

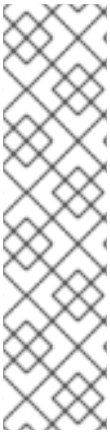
Consider replacing or re-signing any JARs affected by the new restrictions with stronger algorithms.

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 11.0.17, 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).

System properties for controlling the keep-alive behavior of HTTPURLConnection

The Red Hat build of OpenJDK 11.0.17 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 11.0.17 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.

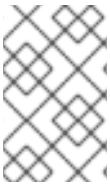
With Red Hat build of OpenJDK 11.0.17, you can use system properties to change the default value for the **keep-alive** time. The **keep-alive** properties control this behavior by changing the HTTP **keep-alive** time of either a server or proxy, so that Red Hat build of OpenJDK's HTTP protocol handler closes idle connections after a specified number of seconds.

Before the Red Hat build of OpenJDK 11.0.17 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 11.0.17 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).

Updated the default PKCS #12 MAC algorithm

The Red Hat build of OpenJDK 11.0.17 updates the default Message Authentication Code (MAC) algorithm for the PKCS #12 keystore to use the **SHA-256** cryptographic hash function rather than the **SHA-1** function. The **SHA-256** function provides a stronger way to secure data.

You can view this update in the **keystore.pkcs12.macAlgorithm** and the **keystore.pkcs12.macIterationCount** system properties.

If you create a keystore with this updated MAC algorithm, and you attempt to use the keystore with an Red Hat build of OpenJDK version earlier than Red Hat build of OpenJDK 11.0.12, you would receive a **java.security.NoSuchAlgorithmException** message.

To use the previous keystore with an Red Hat build of OpenJDK version that is earlier than Red Hat build of OpenJDK 11.0.12, set the **keystore.pkcs12.legacy** system property to **true** to revert the MAC algorithm.

See JDK-8267880 (JDK Bug System).

Deprecated and removed features

Review the following release notes to understand pre-existing features that have been either deprecated or removed in the Red Hat build of OpenJDK 11.0.17 release:

Deprecated Kerberos encryption types

Red Hat build of OpenJDK 11.0.17 deprecates **des3-hmac-sha1** and **rc4-hmac** Kerberos encryption types. By default, Red Hat build of OpenJDK 11.0.17 disables these encryption types, but you can enable them by completing the following action:

- In the **krb5.conf** configuration file, set the **allow_weak_crypto** tab to **true**. This configuration also enables other encryption types, such as **des-cbc-crc** and **des-cbc-md5**.

WARNING

Before you apply this configuration, consider the risks of enabling all of these weak Kerberos encryption types, such as introducing weak encryption algorithms to your Kerberos's authentication mechanism.

You can disable a subset of weak encryption types by explicitly listing an encryption type in one of the following **krb5.conf** configuration file's settings:

- **default_tkt_enctypes**
- **default_tgs_enctypes**
- **permitted_enctypes**

See JDK-8139348 (JDK Bug System).

CHAPTER 4. ADVISORIES RELATED TO THIS RELEASE

The following advisories are issued to bug fixes and CVE fixes included in this release:

- RHSA-2022:7008
- RHSA-2022:7009
- RHSA-2022:7010
- RHSA-2022:7011
- RHSA-2022:7012
- RHSA-2022:7013
- RHSA-2022:7052
- RHSA-2022:7054

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