



Red Hat build of OpenJDK 11

Release notes for Eclipse Temurin 11.0.17

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Review the release notes to understand new features and feature enhancements that have been included with latest build of OpenJDK 11 that is provided by Eclipse Temurin.

Table of Contents

| | |
|--------------------------------------------------------------------------------|----------|
| PREFACE | 3 |
| MAKING OPEN SOURCE MORE INCLUSIVE | 4 |
| CHAPTER 1. SUPPORT POLICY FOR ECLIPSE TEMURIN | 5 |
| CHAPTER 2. ECLIPSE TEMURIN FEATURES | 6 |
| New features and enhancements | 6 |
| Disabled cpu.shares parameter | 6 |
| jdk.httpserver.maxConnections system property | 6 |
| Monitor deserialization of objects with JFR | 6 |
| SHA-1 Signed JARs | 7 |
| System properties for controlling the keep-alive behavior of HTTPURLConnection | 8 |
| Updated the default PKCS #12 MAC algorithm | 9 |
| Deprecated and removed features | 9 |
| Deprecated Kerberos encryption types | 9 |

PREFACE

Open Java Development Kit (OpenJDK) is a free and open-source implementation of the Java Platform, Standard Edition (Java SE). Eclipse Temurin is available in three LTS versions: OpenJDK 8u, OpenJDK 11u, and OpenJDK 17u.

Packages for Eclipse Temurin are made available on Microsoft Windows and on multiple Linux x86 Operating Systems including Red Hat Enterprise Linux and Ubuntu.

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 ECLIPSE TEMURIN

Red Hat will support select major versions of Eclipse Temurin 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 Eclipse Temurin will be supported for a minimum of six years from the time that version is first introduced. For more information, see the [Eclipse Temurin Life Cycle and Support Policy](#).



NOTE

RHEL 6 reached the end of life in November 2020. Because of this, Eclipse Temurin does not support RHEL 6 as a supported configuration.

CHAPTER 2. ECLIPSE TEMURIN FEATURES

Eclipse Temurin does not contain structural changes from the upstream distribution of OpenJDK.

For the list of changes and security fixes included in the latest OpenJDK 11.0.17 release of Eclipse Temurin, see [OpenJDK 11.0.17 Released](#).

New features and enhancements

Review the following release notes to understand new features and feature enhancements included with the Eclipse Temurin 11.0.17 release:

Disabled `cpu.shares` parameter

Before the OpenJDK 11.0.17 release, 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 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 OpenJDK release.

See [JDK-8281181](#) (JDK Bug System).

`jdk.httpservlet.maxConnections` system property

OpenJDK 11.0.17 adds a new system property, `jdk.httpservlet.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.httpservlet.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, 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 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 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 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 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 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 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 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 OpenJDK's HTTP protocol handler closes idle connections after a specified number of seconds.

Before the 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 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 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 OpenJDK **keep-alive** cache on the client uses a timeout of 5 seconds for all connections.

The 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 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 OpenJDK version earlier than OpenJDK 11.0.12, you would receive a **java.security.NoSuchAlgorithmException** message.

To use the previous keystore with an OpenJDK version that is earlier than 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 OpenJDK 11.0.17 release:

Deprecated Kerberos encryption types

OpenJDK 11.0.17 deprecates **des3-hmac-sha1** and **rc4-hmac** Kerberos encryption types. By default, 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).

Revised on 2023-11-02 19:12:42 UTC