Red Hat build of OpenJDK 21

Release notes for Eclipse Temurin 21.0.4

Last Updated: 2024-08-02

Legal Notice

Abstract

The release notes for Eclipse Temurin 21.0.4 provide an overview of new features in OpenJDK 21 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). Eclipse Temurin is available in four LTS versions: OpenJDK 8u, OpenJDK 11u, OpenJDK 17u, and OpenJDK 21u.

Binary files for Eclipse Temurin are available for macOS, Microsoft Windows, and multiple Linux x86 Operating Systems including Red Hat Enterprise Linux and Ubuntu.

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 **Create** 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 ECLIPSE TEMURIN

Red Hat will support select major versions of Eclipse Temurin in its products. For consistency, these versions remain similar to Oracle JDK versions that Oracle designates as long-term support (LTS).

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 that the latest OpenJDK 21 release of Eclipse Temurin includes, see OpenJDK 21.0.4 Released.

New features and enhancements

Review the following release notes to understand new features and feature enhancements included with the Eclipse Temurin 21.0.4 release:

-XshowSettings launcher option includes a security category

In OpenJDK 21.0.4, the **-XshowSettings** launcher option includes a security category, which allows the following arguments to be passed:

Argument	Details
-XshowSettings:security or **-XshowSettings:security:all**	Show all security settings and continue.
-XshowSettings:security:properties	Show security properties and continue.
-XshowSettings:security:providers	Show static security provider settings and continue.
-XshowSettings:security:tls	Show TLS-related security settings and continue.

If third-party security providers are included in the application class path or module path, and configured in the **java.security** file, the output includes these third-party security providers.

See JDK-8281658 (JDK Bug System).

GlobalSign R46 and E46 root certificates added

In OpenJDK 21.0.4, the **cacerts** truststore includes two GlobalSign TLS root certificates:

Certificate 1

- Name: GlobalSign
- Alias name: globalsignr46
- Distinguished name: CN=GlobalSign Root R46, O=GlobalSign nv-sa, C=BE

Certificate 2

- Name: GlobalSign
- Alias name: globalsigne46
- Distinguished name: CN=GlobalSign Root E46, O=GlobalSign nv-sa, C=BE

See JDK-8316138 (JDK Bug System).

Fallback option for POST-only OCSP requests

JDK-8175903, which was introduced in OpenJDK 17, added support for using the HTTP **GET** method for Online Certificate Status Protocol (OCSP) requests. This feature was enabled unconditionally for small requests.

The Internet Engineering Task Force (IETF) RFC 5019 and RFC 6960 explicitly allow and recommend the use of HTTP **GET** requests. However, some OCSP responders do not work well with these types of requests.

OpenJDK 21.0.4 introduces a JDK system property, **com.sun.security.ocsp.useget**. By default, this property is set to **true**, which retains the current behavior of using **GET** requests for small requests. If this property is set to **false**, only HTTP **POST** requests are used, regardless of size.

NOTE

This fallback option for **POST**-only OCSP requests is a non-standard feature, which might be removed in a future release if the use of HTTP **GET** requests with OCSP responders no longer causes any issues.

See JDK-8328638 (JDK Bug System).

RPATH preferred over RUNPATH for $ORIGIN runtime search paths in internal JDK binaries

Native executables and libraries in the JDK use embedded runtime search paths (rpaths) to locate required internal JDK native libraries. On Linux systems, binaries can specify these search paths by using either **DT_RPATH** or **DT_RUNPATH**:

- If a binary specifies search paths by using **DT_RPATH**, these paths are searched *before* any paths that are specified in the **LD_LIBRARY_PATH** environment variable.
- If a binary specifies search paths by using **DT_RUNPATH**, these paths are searched only *after* paths that are specified in **LD_LIBRARY_PATH**. This means that the use of **DT_RUNPATH** can allow JDK internal libraries to be overridden by any libraries of the same name that are specified in **LD_LIBRARY_PATH**.

In earlier releases, the type of runtime search path used was based on the default search path for the dynamic linker. In OpenJDK 21.0.4, to ensure that **DT_RPATH** is used, the **--disable-new-dtags** option is explicitly passed to the linker.

See JDK-8326891 (JDK Bug System).

Jpackage tool resolves symbolic links before passing file paths to dpkg

On Debian and Ubuntu systems, the **jpackage** tool uses the **dpkg -S** command to check which package provides a specific file. However, on newer Debian and Ubuntu systems, the **dpkg -S** command does not resolve symbolic links.

In OpenJDK 21.0.4, **jpackage** resolves symbolic links before passing the real file path to **dpkg**.

See JDK-8295111 (JDK Bug System).

G1 garbage collector ignores existing eden regions for heap resizing during the Remark phase

To comply with **-XX:MinHeapFreeRatio** and **-XX:MaxHeapFreeRatio** settings, the Garbage-First (G1) garbage collector adjusts the Java heap size during the **Remark** phase of garbage collection, keeping the number of free regions within these limits.

In earlier releases, eden regions were considered to be occupied or full for the purpose of calculating Java heap size. This meant the heap size was dependent on the eden occupancy at the time the **Remark** phase was run. However, after the next garbage collection, these eden regions were empty.

In OpenJDK 21.0.4, eden regions are considered to be empty or free during the **Remark** phase calculation. This enhancement means that the G1 garbage collector now expands the Java heap less aggressively and more deterministically, because the number of free regions does not vary as much. This enhancement also aligns Java heap sizing with the full garbage collection heap sizing. However, this might potentially lead to more frequent garbage collection.

See JDK-8314573 (JDK Bug System).

Fix for long garbage collection pauses due to imbalanced iteration during the Code Root Scan phase

The **Code Root Scan** phase of garbage collection finds references to Java objects within compiled code. To speed up this process, a cache is maintained within each region of the compiled code that contains references into the Java heap.

On the assumption that the set of references was small, previous releases used a single thread per region to iterate through these references. This single-threaded approach introduced a scalability bottleneck, where performance could be reduced if a specific region contained a large number of references.

In OpenJDK 21.0.4, multiple threads are used, which helps to remove any scalability bottleneck.

See JDK-8315503 (JDK Bug System).

Fix for long garbage collection pauses in Stop-the-World collectors

In early releases of OpenJDK 21, applications could experience long pause times during stop-the-world (STW) garbage collection. This issue affected various phases of the collection pauses, including **Class Unloading**, **Root Scanning**, and **CodeCache Unloading**, in all types of STW garbage collectors, such as the Serial, Parallel, and G1 collectors.

OpenJDK 21.0.4 resolves these performance issues in STW collectors.

See JDK-8333832 (JDK Bug System).

Change in behavior for AWT headless mode detection on Windows

In earlier releases, unless the **java.awt.headless** system property was set to **true**, a call to **java.awt.GraphicsEnvironment.isHeadless()** returned **false** on Windows Server platforms.

From OpenJDK 21.0.4 onward, unless the **java.awt.headless** property is explicitly set to **false** and if no valid monitor is detected on the current system at runtime, a call to **java.awt.GraphicsEnvironment.isHeadless()** returns **true** on Windows Server platforms. A valid monitor might not be detected, for example, if a session was initiated by a service or by PowerShell remoting.

This change in behavior means that applications running under these conditions, which previously expected to run in a headful context, might now encounter unexpected **HeadlessException** errors being thrown by Abstract Window Toolkit (AWT) operations.

You can reinstate the old behavior by setting the **java.awt.headless** property to **false**. However, if applications are running in headful mode and a valid display is not available, these applications are likely to continue experiencing unexpected issues.

See JDK-8185862 (JDK Bug System).

Revised on 2024-08-02 13:44:35 UTC