Red Hat build of OpenJDK 21

Release notes for Red Hat build of OpenJDK 21.0.5

Last Updated: 2024-10-18

Red Hat build of OpenJDK 21 Release notes for Red Hat build of OpenJDK 21.0.5

Legal Notice

Abstract

The Release notes for Red Hat build of OpenJDK 21.0.5 document provides an overview of new features in Red Hat build of 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). The Red Hat build of OpenJDK is available in four versions: 8u, 11u, 17u, and 21u.

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 **Create** creates and routes the issue to the appropriate documentation team.

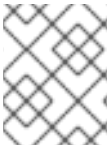
MAKING OPEN SOURCE MORE INCLUSIVE

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 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 21

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 21 changes:

- FIPS support. Red Hat build of OpenJDK 21 automatically detects whether RHEL is in FIPS mode and automatically configures Red Hat build of OpenJDK 21 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 21 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.
- 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 certificates 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 21 release might include new features. Additionally, the latest release might enhance, deprecate, or remove features that originated from earlier Red Hat build of OpenJDK 21 releases.

Red Hat build of OpenJDK enhancements

Red Hat build of OpenJDK 21 provides enhancements to features originally created in earlier releases of Red Hat build of OpenJDK.

Distrust of TLS server certificates issued after 11 November 2024 and anchored by Entrust root CAs

In accordance with similar plans that Google and Mozilla recently announced, Red Hat build of OpenJDK 21.0.5 distrusts TLS certificates that are issued after 11 November 2024 and anchored by Entrust root certificate authorities (CAs). This change in behavior includes any certificates that are branded as AffirmTrust, which are managed by Entrust.

Red Hat build of OpenJDK will continue to trust certificates that are issued on or before 11 November 2024 until these certificates expire.

If a server's certificate chain is anchored by an affected certificate, any attempts to negotiate a TLS session now fail with an exception to indicate that the trust anchor is not trusted. For example:

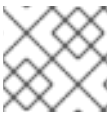
> TLS server certificate issued after 2024-11-11 and anchored by a distrusted legacy Entrust root CA: CN=Entrust.net CertificationAuthority (2048), OU=(c) 1999 Entrust.net Limited,OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.),O=Entrust.net

You can check whether this change affects a certificate in a JDK keystore by using the following **keytool** command:

keytool -v -list -alias *<your_server_alias>* -keystore *<your_keystore_filename>*

If this change affects any certificate in the chain, update this certificate or contact the organization that is responsible for managing the certificate.

If you want to continue using TLS server certificates that are anchored by Entrust root certificates, you can remove **ENTRUST_TLS** from the **jdk.security.caDistrustPolicies** security property either by modifying the **java.security** configuration file or by using the **java.security.properties** system property.



NOTE

Continued use of the distrusted TLS server certificates is at your own risk.

These restrictions apply to the following Entrust root certificates that Red Hat build of OpenJDK includes:

Certificate 1

- Alias name: entrustevca [jdk]
- Distinguished name: CN=Entrust Root Certification Authority, OU=(c) 2006 Entrust, Inc., OU=www.entrust.net/CPS is incorporated by reference, O=Entrust, Inc., C=US

- SHA256: 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:

Certificate 2

- Alias name: entrustrootcaec1 [jdk]
- Distinguished name: CN=Entrust Root Certification Authority - EC1, OU=(c) 2012 Entrust, Inc. - for authorized use only, OU=See www.entrust.net/legal-terms, O=Entrust, Inc., C=US
- SHA256: 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E

Certificate 3

- Alias name: entrustrootcag2 [jdk]
- Distinguished name: CN=Entrust Root Certification Authority - G2, OU=(c) 2009 Entrust, Inc. - for authorized use only, OU=See www.entrust.net/legal-terms, O=Entrust, Inc., C=US
- SHA256: 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7

Certificate 4

- Alias name: entrustrootcag4 [jdk]
- Distinguished name: CN=Entrust Root Certification Authority - G4, OU=(c) 2015 Entrust, Inc. - for authorized use only, OU=See www.entrust.net/legal-termsO=Entrust, Inc., C=US
- SHA256: DB:35:17:D1:F6:73:2A:2D:5A:B9:7C:53:3E:C7:07:79:EE:32:70:A6:2F:B4:AC:42:38:37:24:60:E6:F(

Certificate 5

- Alias name: entrust2048ca [jdk]
- Distinguished name: CN=Entrust.net Certification Authority (2048), OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net
- SHA256: 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1

Certificate 6

- Alias name: affirmtrustcommercialca [jdk]
- Distinguished name: CN=AffirmTrust Commercial, O=AffirmTrust, C=US
- SHA256: 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:(

Certificate 7

- Alias name: affirmtrustnetworkingca [jdk]
- Distinguished name: CN=AffirmTrust Networking, O=AffirmTrust, C=US

- SHA256: 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:FC

Certificate 8

- Alias name: affirmtrustpremiumca [jdk]
- Distinguished name: CN=AffirmTrust Premium, O=AffirmTrust, C=US
- SHA256: 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9

Certificate 9

- Alias name: affirmtrustpremiumeccca [jdk]
- Distinguished name: CN=AffirmTrust Premium ECCO=AffirmTrust, C=US
- SHA256: BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:

See JDK-8337664 (JDK Bug System) and JDK-8341059 (JDK Bug System).

KEM.getInstance() method checks if a third-party security provider is signed

The JDK's cryptographic framework authenticates third-party security provider implementations by determining the provider's codebase and by verifying the provider's signature.

In earlier releases, the JDK did not authenticate key encapsulation mechanism (KEM) implementations.

Red Hat build of OpenJDK 21.0.5 authenticates KEM implementations in a manner that is consistent with other JDK service types, such as Cipher and Mac providers.

See JDK-8322971 (JDK Bug System).

Reduced verbose locale output in -XshowSettings launcher option

In earlier releases, the **-XshowSettings** launcher option printed a long list of available locales, which obscured other settings.

In Red Hat build of OpenJDK 21.0.5, the **-XshowSettings** launcher option no longer prints the list of available locales by default. If you want to view all settings that relate to the available locales, you can use the **-XshowSettings:locale** option.

See JDK-8310201 (JDK Bug System).

Additional timestamp and thread options for java.security.debug system property

Red Hat build of OpenJDK 21.0.5 adds the following options to the **java.security.debug** property, which can be applied to any specified component:

- The **+timestamp** option prints a timestamp with each debug statement.
- The **+thread** option prints thread and caller information for each debug statement.

For example, **-Djava.security.debug=all+timestamp+thread** enables debug information for all components with both timestamps and thread information. Alternatively, **-Djava.security.debug=properties+timestamp** enables debug information only for security properties

and includes a timestamp. You can use **-Djava.security.debug=help** to display a complete list of supported components and options.

See JDK-8051959 (JDK Bug System).

SSL.com root certificates added

In Red Hat build of OpenJDK 21.0.5, the **cacerts** truststore includes two SSL.com TLS root certificates:

Certificate 1

- Name: SSL.com
- Alias name: ssltlsrootecc2022
- Distinguished name: CN=SSL.com TLS ECC Root CA 2022, O=SSL Corporation, C=US

Certificate 2

- Name: SSL.com
- Alias name: ssltlsrootrsa2022
- Distinguished name: CN=SSL.com TLS RSA Root CA 2022, O=SSL Corporation, C=US

See JDK-8341057 (JDK Bug System).

HTTP client enhancements

Red Hat build of OpenJDK 21.0.5 limits the maximum header field size that the HTTP client accepts within the JDK for all supported versions of the HTTP protocol. The header field size is computed as the sum of the size of the uncompressed header name, the size of the uncompressed header value, and an overhead of 32 bytes for each field section line. If a peer sends a field section that exceeds this limit, a **java.net.ProtocolException** is raised.

Red Hat build of OpenJDK 21.0.5 introduces a **jdk.http.maxHeaderSize** system property that you can use to change the maximum header field size (in bytes). Alternatively, you can disable the maximum header field size by setting the **jdk.http.maxHeaderSize** property to zero or a negative value. The **jdk.http.maxHeaderSize** property is set to 393,216 bytes (that is, 384KB) by default.

See JDK-8328286 (JDK Bug System).

ClassLoadingMXBean and MemoryMXBean APIs have isVerbose() methods consistent with their setVerbose() methods

The **setVerbose(boolean enabled)** method for the **ClassLoadingMXBean** API displays the following behavior:

- If **enabled** is **true**, the **setVerbose** method sets **class+load*** logging on standard output (stdout) at the **Info** level.
- If **enabled** is **false**, the **setVerbose** method disables **class+load*** logging on stdout.

In earlier releases, the **isVerbose()** method for the **ClassLoadingMXBean** API checked if **class+load** logging was enabled at the **Info** level on any type of log output, not just stdout. In this situation, if you enabled logging to a file by using the **java -Xlog** option, the **isVerbose()** method returned **true** even if **setVerbose(false)** was called, which resulted in counterintuitive behavior. The **isVerbose()** method for the **MemoryMXBean** API also displayed similar counterintuitive behavior.

From Red Hat build of OpenJDK 21.0.5 onward, the **ClassLoadingMXBean.isVerbose()** and **MemoryMXBean.isVerbose()** methods display the following behavior:

- **ClassLoadingMXBean::isVerbose()** returns **true** only if **class+load*** logging is enabled at the **Info** level (or higher) specifically on standard output (stdout).
- **MemoryMXBean::isVerbose()** returns **true** only if garbage collector logging is enabled at the **Info** level (or higher) on stdout.

See JDK-8338139 (JDK Bug System).

CHAPTER 4. ADVISORIES RELATED TO THIS RELEASE

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

- RHSA-2024:8127
- RHSA-2024:8128
- RHSA-2024:8129

Revised on 2024-10-18 15:09:13 UTC