Zulu Release Notes

Zulu 15.28 (15.0.1) for Arm v8 64-bit

Release Date: October 20, 2020

Document Version: 1.4

Last Modified: October 30, 2020

Revision History

Revision	Date	Description
1.0	October 20, 2020	Initial version of the document.
1.1	October 21, 2020	Updated the description of the new property "jdk.jndi.ldap.mechsAllowedToSendCredentials."
1.2	October 23, 2020	Corrected the release version in What's New section.
1.3	October 23, 2020	Updated In-Depth Non-CVE Security Fixes and Other OpenJDK Bug Fixes tables.
1.4	October 30, 2020	Updated the description of CA builds. Increased the font size in the CVE table.

Table of Contents

Zulu Overview

Azul Systems ® Zulu® is an implementation of the Java Standard Edition (SE) based on OpenJDK and optimized for embedded devices. Zulu includes Java Development Kit (JDK) that provides a collection of tools for application development.

What's New

October 20, 2020

Zulu 15.28.13 release

Summary

Azul Zulu distribution types:

SA are tested, certified, and commercially supported Azul Zulu builds of OpenJDK whereby Azul ensures that software that uses the Accessible APIs of the product is not required to carry a specific license and that such use does not contaminate the code or intellectual property of such software with any license requirements.

CA are Azul Zulu builds of OpenJDK that are free to download and use.

The details of the released Zulu versions are specified in the following table:

Java SE Version	Java Update Type*	Zulu Version	JDK Version	Based on**	
				Zulu Version	JDK Version
15	PSU	15.28	15.0.1+8	15.27	15 GA

***Java Update Type:**

- **CPU** (Critical Patch Updates) contain fixes to security vulnerabilities and critical bug fixes. Zulu CPU releases are generally based on prior-cycle PSU releases, with only security fixes applied. They provide a low-risk vehicle for the potentially urgent deployment of security fixes when issues of sufficient severity arise. CPU releases are available in SA distributions.
- **PSU** (Patch Set Updates) incorporates all of fixes in the corresponding CPU, as well as additional non-security bug fixes. Zulu PSU releases incorporate both security fixes and other accumulated changes that align the release contents with the associated OpenJDK project quarterly release. PSU releases are available in SA and CA distributions.

****Based on:** Zulu CPU releases are based on prior-cycle PSU releases. Zulu PSU releases are based on the current-cycle CPU releases.

IANA time zone data version

This release of Zulu comes with IANA time zone data version 2020a. For more information, see Timezone Data Versions in the JRE Software. Please note that Zulu with IANA time zone data version 2020b version or later will be delivered shortly.

New Features

Shenandoah garbage collector included

Now both Zulu 11 and Zulu 15 PSU releases include Shenandoah as a product feature. Please note, it is disabled by default in runtime. To enable Shenandoah in runtime, use `-XX:+UseShenandoahGC` flag. Please note that the product may not be mature enough for production use and should be pre-tested with your application.

Additions and Changes in Behavior

JDK-8237990: New system and environment runtime property for LDAP context

Introduced in all Zulu versions.

The `jdk.jndi.ldap.mechsAllowedToSendCredentials` property sets the list of authentication mechanisms that are allowed to send credentials over a non-encrypted connection in the LDAP context. Possible values are:

- `all` allows all mechanisms
- ` ` (empty value) allows none
- `mech1,mech2,...,mechN` allows the given authentication mechanisms

If the property is not set, all mechanisms are allowed. Note that "none" and "anonymous" authentication mechanisms are always allowed irrespective of the property value.

In previous versions, user credentials could be sent over an unencrypted connection.

JDK-8245417: New runtime properties to control TLS workflow

Introduced in all Zulu versions.

- The `jdk.tls.maxHandshakeMessageSize` property controls the maximum size limit for the handshake message. The default property value is 32768.
- The `jdk.tls.maxCertificateChainLength` property controls the maximum length for the certificate chain. The default property value is 10.

Note that when the corresponding data size exceeds the allowed value, an exception is thrown from various parts of the JDK. For example: `The size of the handshake message <actual size> exceeds the maximum allowed size <maxHandshakeMessageSize>`.

In previous versions, the maximum handshake messages size was limited to 2^24 bytes, and there was no limit on the length of certificate chains.

JDK-8236862: New runtime property to control the number of interfaces allowed for Proxies

Introduced in all Zulu versions.

The `jdk.serialProxyInterfaceLimit` property sets the implementation limit on the number of interfaces allowed for Proxies. The property range is from 0 to 65535; the default value is 65535.

In previous versions, the maximum number of interfaces was 65535 and you couldn't set a lower value.

JDK-8233624: New mapping rules from a Java native method name to a C native library implementation function name

Introduced in all Zulu versions.

The JNI methods mapping rules have been changed, for details on new rules, see Troubleshooting tips.

Note, if a Java class or package name for some reason begins with a digit "0", "1", "2", or "3", no native library search is performed, and the attempt to link the native method invocation throws `UnsatisfiedLinkError`.

To resolve compatibility issues, you can either set the `XX:+UseLegacyJNINameEscaping` runtime flag to skip the extra mapping check or notify your library provider to update the JNI names according to the new mapping rules and to reissue and redeploy the libraries.

Features and Supported Platforms

Supported Platforms

Zulu is delivered as binary builds of OpenJDK on Linux Kernel 3.10.x or higher. The following platforms are supported:

- Arm v8 CPU with 64-bit support.
- Linux Arm 64-bit EABI.

Supported Functionality

Hotspot Compilers

Zulu for Arm v8 64-bit architecture supports both server (C2) and client (C1) compilers in addition to the optimized template interpreter. The following command-line options can be used to switch between the implementations:

- `-Xint` – Runs the application in interpreted-only mode.
- `-Xcomp` – Enforces compilation of methods on first invocation.
- `-Xbatch` – Disables background compilation so that compilation of all methods proceeds as a foreground task until completed.
- `-XX:[+/-]TieredCompilation` - Enables or disables the tiered compilation (enabled by default). When the tiered compilation is disabled, only the server compiler is used.
- `-XX:TieredStopAtLevel=X` -Limits the compilation level (0 - interpreted, 1 - client compiler is used only, 4 - full tiered compilation to up C2).

Refer to the extended list of the Advanced JIT Compiler Options, for more information about fine-tuning the compilation behavior. For example, you can tune

the compilation thresholds in order to balance the startup time and execution performance.

Getting Started with Zulu

To start working with the Zulu perform the following steps:

Extract the Installation Archive

- Download one of the installation archives and save it in a reasonable location on your system.
- Extract the downloaded archive:

```
$ tar -xzf zulu15.28.13-ca-jdk15.0.1-linux_
aarch64.tar.gz
```

- Copy the extracted directory into the following location:

```
/usr/lib/jvm
```

Add Debug Symbols

If you do not need debug symbols, skip this step.

- Download one of the archives with the debug symbols:

```
zulu15.28.13-ca-dbg-jdk15.0.1-linux_aarch64.zip
```

- Copy the archive to

```
/usr/lib/jvm/zulu15.28.13-ca-jdk15.0.1-linux_aarch64
```

- Extract the archive. For example,

```
$ unzip zulu15.28.13-ca-dbg-jdk15.0.1-linux_
aarch64.zip
```

Verify Java Version

- Run a simple Java command:

```
$ java -version
```

If needed, provide the fully qualified path. For example:

```
$ /usr/lib/jvm/zulu15.28.13-ca-jdk15.0.1-linux_
aarch64/bin/java -version
```

- Inspect the system response. Correct sample output should look as follows:

```
openjdk version "15.0.1" 2020-10-20
```

```
OpenJDK Runtime Environment Zulu15.28+13-CA (build
15.0.1+8)
```

```
OpenJDK 64-Bit Server VM Zulu15.28+13-CA (build
15.0.1+8, mixed mode)
```

Zulu Resolved Issues

This section summarizes JDK Common Vulnerabilities and Exposure (CVE) fixes reflecting October, 2020 OpenJDK changes implemented for the following Zulu levels:

- Zulu 15
- Zulu 13
- Zulu 11
- Zulu 8

JDK Common Vulnerabilities and Exposure (CVE) Fixes

October, 2020 CVE Fixes

CVSS VERSION 3.0 RISK															
CVE #	Component	Protocol	Remote Exploit without Auth.	Base Score	Attack Vector	Attack Complex	Privs Req'd	User Interact	Scope	Confid-entiality	Inte-grity	Avail-ability	Supported Zulu Versions Affected	Modules Changed to Address CVE	Notes
CVE-2020-14803	Libraries	Multiple	Yes	5.3	Network	Low	None	None	Unchanged	Low	None	None	15, 13, 11	15, 13, 11: java.base	Note 1
CVE-2020-14792	Hotspot	Multiple	Yes	4.2	Network	High	None	Required	Unchanged	Low	Low	None	15, 13, 11, 8, 7, 6	15, 13, 11: hotspot 8, 7, 6: HOTSPOT	Note 2

October, 2020 CVE Fixes

CVSS VERSION 3.0 RISK															
CVE #	Component	Protocol	Remote Exploit without Auth.	Base Score	Attack Vector	Attack Complex	Privs Req'd	User Interact	Scope	Confid-entiality	Inte-grity	Avail-ability	Supported Zulu Versions Affected	Modules Changed to Address CVE	Notes
CVE-2020-14797	Libraries	Multiple	Yes	3.7	Network	High	None	None	Unchanged	None	Low	None	15, 13, 11, 8, 7	15, 13, 11: java.base 8, 7: JDK	Note 2
CVE-2020-14782	Libraries	Multiple	Yes	3.7	Network	High	None	None	Unchanged	None	Low	None	15, 13, 11, 8, 7	15, 13, 11: java.base 8, 7: JDK	Note 2
CVE-2020-14781	JNDI	Multiple	Yes	3.7	Network	High	None	None	Unchanged	Low	None	None	15, 13, 11, 8, 7, 6	15, 13, 11: java.naming 8, 7, 6: JDK	Note 2
CVE-2020-14779	Serialization	Multiple	Yes	3.7	Network	High	None	None	Unchanged	None	None	Low	15, 13, 11, 8, 7, 6	15, 13, 11: java.base 8, 7, 6: JDK JDK	Note 2

October, 2020 CVE Fixes

CVSS VERSION 3.0 RISK															
CVE #	Component	Protocol	Remote Exploit without Auth.	Base Score	Attack Vector	Attack Complex	Privs Req'd	User Interact	Scope	Confid-entiality	Inte-grity	Avail-ability	Supported Zulu Versions Affected	Modules Changed to Address CVE	Notes
CVE-2020-14798	Libraries	Multiple	Yes	3.1	Network	High	None	Required	Unchanged	None	Low	None	15, 13, 11, 8, 7	15, 13, 11: java.base 8, 7: JDK	Note 1
CVE-2020-14796	Libraries	Multiple	Yes	3.1	Network	High	None	Required	Unchanged	Low	None	None	15, 13, 11, 8, 7	15, 13, 11: java.base 8, 7: JDK	Note 1

Base and Impact Metric:

Metrics	Values
Attack Vector	Network (N), Adjacent (A), Local (L), and Physical (P)
Attack Complexity	Low (L) and High (H)
Privileges Required	None (N), Low (L), and High (H)
User Interaction	None (N) and Required (R)
Scope	Unchanged (U) and Changed (C)
Confidentiality Impact	High (H), Low (L), and None (N)

Integrity Impact	High (H), Low (L), and None (N)
Availability Impact	High (H), Low (L), and None (N)

Notes:

ID	Notes
1	This vulnerability applies to Java deployments that load and run untrusted code (e.g., code that comes from the internet) and rely on the Java sandbox for security. This vulnerability does not apply to Java deployments, typically in servers, that load and run only trusted code (e.g., code installed by an administrator).
2	This vulnerability applies to client and server deployment of Java. This vulnerability can be exploited through untrusted code executed under Java sandbox restrictions. It can also be exploited by supplying data to APIs in the specified Component without using untrusted code executed under Java sandbox restrictions, such as through a web service.

In-Depth Non-CVE Security Fixes

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8253019	Enhanced JPEG decoding	CPU
JDK-8249927	Specify limits of jdk.serialProxyInterfaceLimit	CPU
JDK-8248574	Improve jpeg processing	CPU
JDK-8248177	Improve XML support	CPU
JDK-8248171	Better query support	CPU
JDK-8245417	Improve certificate chain handling	CPU
JDK-8245412	Better class definitions	CPU

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8245407	Enhance zoning of times	CPU
JDK-8244955	Additional Fix for JDK-8240124	CPU
JDK-8244479	Further constrain certificates	CPU
JDK-8243302	Advanced class supports	CPU
JDK-8240124	Better VM Interning	CPU
JDK-8236196	Improve string pooling	CPU
JDK-8233624	Enhance JNI Inkage	CPU

Other OpenJDK Bug Fixes

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8253019	Enhanced JPEG decoding	PSU
JDK-8251910	Shenandoah: Handshake threads between weak-roots and reset phases	PSU
JDK-8251859	sun/security/validator/PKIXValAndRevCheckTests.java fails with "RuntimeException: Received unexpected exception"	PSU
JDK-8251451	Shenandoah: Remark ObjectSynchronizer roots with I-U	PSU
JDK-8251359	Shenandoah: filter null oops before calling enqueue/SATB barrier	PSU

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8250876	Fix issues with cross-compile on macos	PSU
JDK-8250844	Make sure {type,obj}ArrayOopDesc accessors check the bounds	PSU
JDK-8250841	Shenandoah: need to reset/finish dead counters for StringTable/ResolvedMethodTable during STW root processing	PSU
JDK-8250665	Wrong translation for the month of May in ar_JO, ar_LB and ar_SY	PSU
JDK-8250612	jvmciCompilerToVM.cpp declares jio_printf with "void" return type, should be "int"	PSU
JDK-8250582	Revert Principal Name type to NT-UNKNOWN when requesting TGS Kerberos tickets	PSU
JDK-8250548	libgraal can deadlock in -Xcomp mode	PSU
JDK-8249953	Shenandoah: gc/shenandoah/mxbeans tests should account for corner cases	PSU
JDK-8249927	Specify limits of jdk.serialProxyInterfaceLimit	PSU
JDK-8249801	Shenandoah: Clear soft-refs on requested GC cycle	PSU
JDK-8249672	Include microcode revision in features_string on x86	PSU
JDK-8249649	Shenandoah: provide per-cycle pacing stats	PSU
JDK-8249230	Shenandoah: assertion failure with -XX:-ResizeTLAB	PSU
JDK-8248987	AOT's Linker.java seems to eagerly fail-fast on Windows.	PSU
JDK-8248745	Add jarsigner and keytool tests for restricted algorithms	PSU
JDK-8248652	Shenandoah: SATB buffer handling may assume no forwarded objects	PSU

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8248634	Shenandoah: incorrect include in shenandoahInitLogger.cpp	PSU
JDK-8248632	Shenandoah: build fails without both JVMTI and JFR	PSU
JDK-8248574	Improve jpeg processing	PSU
JDK-8248495	[macos] zerovm is broken due to libffi headers location	PSU
JDK-8248467	C2: compiler/intrinsics/object/TestClone fails with -XX:+VerifyGraphEdges	PSU
JDK-8248041	Shenandoah: pre-Full GC root updates may miss some roots	PSU
JDK-8247860	Shenandoah: add update watermark line in rich assert failure message	PSU
JDK-8247845	Shenandoah: refactor TLAB/GCLAB retirement code	PSU
JDK-8247757	Shenandoah: split heavy tests by heuristics to improve parallelism	PSU
JDK-8247754	Shenandoah: mxbeans tests can be shorter	PSU
JDK-8247751	Shenandoah: options tests should run with smaller heaps	PSU
JDK-8247736	Shenandoah: assert(_nm->is_alive()) failed: only alive nmethods here	PSU
JDK-8247593	Shenandoah: should not block pacing reporters	PSU
JDK-8247367	Shenandoah: pacer should wait on lock instead of exponential backoff	PSU
JDK-8245417	Improve certificate chain handling	PSU
JDK-8245412	Better class definitions	PSU
JDK-8245407	Enhance zoning of times	PSU

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8244955	Additional Fix for JDK-8240124	PSU
JDK-8244479	Further constrain certificates	PSU
JDK-8244136	Improved Buffer supports	PSU
JDK-8243470	[macos] bring back O2 opt level for unsafe.cpp	PSU
JDK-8243321	Add Entrust root CA - G4 to Oracle Root CA program	PSU
JDK-8243320	Add SSL root certificates to Oracle Root CA program	PSU
JDK-8243302	Advanced class supports	PSU
JDK-8242695	Enhanced Buffer Support	PSU
JDK-8242685	Better Path Validation	PSU
JDK-8242680	Improved URI support	PSU
JDK-8241574	Shenandoah: remove ShenandoahAssertToSpaceClosure	PSU
JDK-8241114	Better range handling	PSU
JDK-8241065	Shenandoah: remove leftover code after JDK-8231086	PSU
JDK-8241007	Shenandoah: remove ShenandoahCriticalControlThreadPriority support	PSU
JDK-8240124	Better VM Interning	PSU
JDK-8237995	Enhance certificate processing	PSU
JDK-8237990	Enhanced LDAP contexts	PSU

OpenJDK Patch ID	Synopsis	CPU/PSU
JDK-8236862	Enhance support of Proxy class	PSU
JDK-8236196	Improve string pooling	PSU
JDK-8233624	Enhance JNI Inkage	PSU

License Changes

The TPL document was not changed in this release.

Refer to the `TPL` file for the complete list of third-party software packages licensed for this release.

Legal Notice

Published October 20, 2020

Products and specifications discussed in this document may reflect future versions and are subject to change without notice. Azul Systems assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.