Open**JDK**

Contributing Sponsoring Developers' Guide **Vulnerabilities** JDK GA/EA Builds Mailing lists Wiki · IRC

Bylaws · Census Legal Workshop

JEP Process Source code Mercurial GitHub

Tools jtreg harness

Groups (overview) Adoption Build Client Libraries Compatibility & Specification Review Compiler Conformance Core Libraries

Governing Board HotSpot IDE Tooling & Support Internationalization **IMX** Members Networking Porters Quality

Security Serviceability Vulnerability Web **Projects** (overview, archive) Amber Babylon CRaC Caciocavallo

Closures Code Tools Coin Common VM Interface Compiler Grammar Detroit Developers' Guide Device I/O Duke Galahad Graal IcedTea JDK 7 JDK 8 JDK 8 Updates IDK 9 JDK (..., 21, 22, 23) JDK Updates JavaDoc.Next

Kona Kulla Lambda Lanai Leyden Lilliput Locale Enhancement Loom Memory Model Update Metropolis Mission Control Multi-Language VM Nashorn New I/O OpenJFX Panama Penrose

Jigsaw

Port: AArch32 Port: AArch64 Port: BSD Port: Haiku Port: Mac OS X Port: MIPS Port: Mobile Port: PowerPC/AIX Port: RISC-V Port: s390x Portola SCTP Shenandoah Skara Sumatra **Tiered Attribution** Type Annotations Valhalla Verona

ORACLE

VisualVM Wakefield Zero

ZGC

JEP 449: Deprecate the Windows 32-bit x86 Port for Removal

Author George Adams & Bruno Borges Owner George Adams *Type* Feature Scope Implementation Status Closed / Delivered Release 21 Component hotspot/other Discussion jdk dash dev at openjdk dot org Effort S Duration S Relates to JEP 479: Remove the Windows 32-bit x86 Port Reviewed by Jesper Wilhelmsson Endorsed by Dalibor Topic, Vladimir Kozlov Created 2023/02/24 15:30 *Updated* 2024/05/07 16:40 *Issue* 8303167

Summary

Deprecate the Windows 32-bit x86 port, with the intent to remove it in a future release.

Goals

- Update the build system to issue an error message when an attempt is made to configure a build for Windows 32-bit x86 (x86-32). The error message will be suppressible via a new configuration option.
- Mark the port, and related port-specific features, as deprecated for removal in the relevant documentation.

Non-Goals

- It is not a goal to change the status of the affected port in any prior release. The earliest release to which this JEP could be targeted is JDK 21.
- It is not a goal to deprecate any other 32-bit port.

Motivation

- Allow contributors in the OpenJDK Community to accelerate the development of new features and enhancements that will move the platform forward.
- The implementation of JEP 436 (Virtual Threads) for Windows x86-32 falls back to the use of kernel threads and therefore does not bring the expected benefits of Project Loom.
- Windows 10, the last Windows operating system to support 32-bit operation, will reach End of Life in October 2025.

Description

An attempt to configure a Windows x86-32 build will produce the following output:

```
$ bash ./configure
checking compilation type... native
configure: error: The Windows 32-bit x86 port is deprecated and may be removed in a future release. \
Use --enable-deprecated-ports=yes to suppress this error.
configure exiting with result code 1
```

The new build configuration option --enable-deprecated-ports=yes will suppress

```
the error and continue:
   $ bash ./configure --enable-deprecated-ports=yes
```

checking compilation type... native configure: WARNING: The Windows 32-bit x86 port is deprecated and may be removed in a future release.

Build performance summary: * Cores to use: 32 * Memory limit: 96601 MB The following warnings were produced. Repeated here for convenience:

WARNING: The Windows 32-bit x86 port is deprecated and may be removed in a future release. \$

Alternatives

The alternative would be to continue to support Windows x86-32. This would require active maintainers who can provide a sustainable and performant implementation of Virtual Threads, as well as future JEPs, to ensure that the JDK on Windows x86-32 continues to meet the expectations of Java developers.

Risks and Assumptions

32-bit JVMs are still used on Windows due to integration with 32-bit native libraries (DLLs). Their users cannot migrate directly to 64-bit JVMs because a 64-bit process on Windows cannot load 32-bit DLLs. While Windows x64 is capable of running 32bit applications by emulating an 32-bit environment through WOW64, applications will suffer dramatic performance degradation despite the assumed memory footprint benefits.

We therefore assume that

- Users can continue to run existing builds of the Windows 32-bit JVM to integrate with native 32-bit libraries and, if necessary, expose 32-bit functionality via remote APIs to be consumed by applications running on a 64-bit JVM within the same environment; and
- Legacy systems are unlikely to migrate to versions of the JDK following the release of Java 21.