



- Installing
- Contributing
- Sponsoring
- Developers' Guide
- Vulnerabilities
- JDK GA/EA Builds
- Mailing lists
- Wiki · IRC
- Bylaws · Census
- Legal
- Workshop
- JEP Process
- Source code
 - Mercurial
 - GitHub
- Tools
 - Git
 - jtreg harness
- Groups
 - (overview)
 - Adoption
 - Build
 - Client Libraries
 - Compatibility & Specification
 - Review
 - Compiler
 - Conformance
 - Core Libraries
 - Governing Board
 - HotSpot
 - IDE Tooling & Support
 - Internationalization
 - JMX
 - Members
 - Networking
 - Porters
 - Quality
 - Security
 - Serviceability
 - Vulnerability
 - Web
- Projects
 - (overview, archive)
 - Amber
 - Audio Engine
 - CRaC
 - Caciocavallo
 - Closures
 - Code Tools
 - Coin
 - Common VM
 - Interface
 - Compiler Grammar
 - Detroit
 - Developers' Guide
 - Device I/O
 - Duke
 - Font Scaler
 - Galahad
 - Graal
 - Graphics Rasterizer
 - IcedTea
 - JDK 7
 - JDK 8
 - JDK 8 Updates
 - JDK 9
 - JDK (... , 21, 22)
 - JDK Updates
 - JavaDoc.Next
 - Jigsaw
 - Kona
 - Kulla
 - Lambda
 - Lanai
 - Leyden
 - Lilliput
 - Locale Enhancement
 - Loom
 - Memory Model
 - Update
 - Metropolis
 - Mission Control
 - Modules
 - Multi-Language VM
 - Nashorn
 - New I/O
 - OpenJFX
 - Panama
 - Penrose
 - 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
 - Tsan
 - Type Annotations
 - Valhalla
 - Verona
 - VisualVM
 - Wakefield
 - Zero
 - ZGC



Project Panama: Interconnecting JVM and native code

We are improving and enriching the connections between the Java virtual machine and well-defined but “foreign” (non-Java) APIs, including many interfaces commonly used by C programmers.

To this end, Project Panama will include most or all of these components:

- native function calling from JVM
- native data access from JVM or inside JVM heap
- new data layouts in JVM heap
- native metadata definition for JVM
- header file API extraction tools (jextract)
- native library management APIs
- native-oriented interpreter and runtime “hooks”
- class and method resolution “hooks”
- native-oriented JIT optimizations
- tooling or wrapper interposition for safety
- exploratory work with difficult-to-integrate native libraries

Community

This Project is sponsored by the [Hotspot Group](#).

- Mailing lists & News
 - [panama-dev](#) — Foreign Function & Memory API, Vector API
 - [jextract-dev](#) — jextract tool
 - Stay tuned on latest Panama development at [Inside.java](#)
- Design documents
 - [Panama foreign memory access](#)
 - [Panama foreign function support](#)
 - Panama jextract usage examples
 - early problem space overview: [the isthmus in the VM](#)
- JEPs
 - Foreign Function & Memory API: JEP-424
 - Vector API: [JEP-426](#)
- Talks
 - [Project Panama: say goodbye to JNI!](#)
 - [The Vector API in JDK 17](#)
 - [ByteBuffers are dead, long live ByteBuffers!](#)
 - [Deconstructing Panama](#)
 - [Vector API](#)
 - [Panama: a foreign policy for Java](#)

Repository organization

Project Panama is designed to incubate a series of components for eventual inclusion in the JDK, via curated merge. Project Panama features are being actively developed in the following repositories:

- [Panama foreign support](#), which adds support for foreign memory access, as well as for foreign function calls;
- [Panama vector support](#), which adds vectorization support in Java through JVM intrinsics; and
- [jextract](#), a tool which mechanically generate Java bindings from native library headers.

The legacy Panama repository is also available [here](#), although we do not expect to carry out further work there; as such this repository should not be used (and in the future we might make this more explicit by marking the legacy repository as *read-only*).

