



- 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



## JEP 380: Unix-Domain Socket Channels

<i>Owner</i>	Michael McMahon
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	16
<i>Component</i>	core-libs / java.nio
<i>Discussion</i>	nio dash dev at openjdk dot java dot net
<i>Effort</i>	S
<i>Duration</i>	M
<i>Reviewed by</i>	Alan Bateman, Brian Goetz, Chris Hegarty
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2020/02/06 09:45
<i>Updated</i>	2021/06/29 09:00
<i>Issue</i>	<a href="#">8238588</a>

### Summary

Add Unix-domain (AF\_UNIX) socket support to the [socket channel](#) and [server-socket channel](#) APIs in the `java.nio.channels` package. Extend the [inherited channel mechanism](#) to support Unix-domain socket channels and server socket channels.

### Goals

Unix-domain sockets are used for inter-process communication (IPC) on the same host. They are similar to TCP/IP sockets in most respects, except that they are addressed by filesystem path names rather than Internet Protocol (IP) addresses and port numbers. The goal of this JEP is to support all of the features of Unix-domain sockets that are common across the major Unix platforms and Windows. Unix-domain socket channels will behave the same as existing TCP/IP channels in terms of read/write behavior, connection setup, acceptance of incoming connections by servers, multiplexing with other non-blocking selectable channels in a selector, and support of relevant socket options.

### Non-Goals

It is not a goal to support features that are not common across the major Unix platforms and Windows. This includes Linux-specific features such as the abstract filesystem-independent namespace. It also includes features that are generally supported on Unix but unsupported on Windows, such as socket pairs. Support for these features can be revisited in the future if needed and, in the case of the missing Windows features, if the Windows platform evolves to support them. An exception to this non-goal is support for peer credentials, which can be implemented as a JDK specific socket option on platforms that support it. Other socket options may be investigated as follow up work, possibly also as JDK specific options, after this JEP is completed.

### Motivation

For local, inter-process communication, Unix-domain sockets are both more secure and more efficient than TCP/IP loopback connections.

- Unix-domain sockets are strictly for communication between processes on the same system. Applications that are not intended to accept remote connections can improve security by using Unix-domain sockets.
- Unix-domain sockets are further protected by operating-system enforced, filesystem-based access controls.
- Unix-domain sockets have faster setup times and higher data throughput than TCP/IP loopback connections.
- Unix-domain sockets may be a better solution than TCP/IP sockets for container environments, where communication between containers on the same system is required. This can be achieved using sockets located in shared volumes.

Unix-domain sockets have long been a feature of most Unix platforms, and are now supported in Windows 10 and Windows Server 2019.

### Description

To support Unix-domain socket channels we will add the following API elements:

- A new socket address class, `java.net.UnixDomainSocketAddress`;
- A UNIX constant value in the existing `java.net.StandardProtocolFamily` enum;
- New open factory methods on `SocketChannel` and `ServerSocketChannel` that specify the protocol family
- Updates to the `SocketChannel` and `ServerSocketChannel` specifications to specify how channels to Unix domain sockets behave.

### Alternatives

An application could access AF\_UNIX address structures and socket system calls directly, either via the Java Native Interface (JNI) or Project Panama. Socket objects of this type would, however, not be compatible with the existing `SocketChannel`

API and therefore could not be multiplexed with other selectable channels using the Selector API.

Testing

Automatic unit tests will exercise the API and implementation. These tests will run on all supported Unix platforms and on multiple versions of Windows, including some that support Unix-domain sockets and some that do not.

Risks and Assumptions

Existing code that uses the `SocketChannel` and `ServerSocketChannel` classes often assumes that instances of `SocketAddress` returned by those APIs can be blindly cast to `InetSocketAddress`. This cast will fail with Unix-domain socket channels.