jDeploy 5.4 Release Notes

Table of Contents

Overview

jDeploy 5.4 adds support for JetBrains Runtime (JBR) as a JRE provider, increases the default file descriptor limit on macOS, and improves JavaFX version configuration.

What's New in 5.4

JetBrains Runtime (JBR) Support

Applications can now use JetBrains Runtime as their JRE provider, enabling access to JCEF (Java Chromium Embedded Framework) and enhanced rendering capabilities.

Configuration

Specify JBR in package.json:

```
{
  "jdeploy": {
    "javaVersion": "21",
    "jdkProvider": "jbr",
    "jbrVariant": "jcef"
  }
}
```

Available Variants

Variant	Description
`standard`	JBR without JCEF
`jcef`	JBR with JCEF (recommended for GUI applications)
`sdk`	JBR SDK without JCEF (for development)
`sdk_jcef`	JBR SDK with JCEF (for development with embedded browser)

Environment Variable Configuration

```
export JDEPLOY_JDK_PROVIDER=jbr
export JDEPLOY_JBR_VARIANT=jcef
```

Platform Support

JBR is available on all platforms:

- macOS (x64 and ARM64)
- Windows (x64 and ARM64)
- Linux (x64 and ARM64)

JavaFX with JBR

JBR does not include JavaFX. When using JBR with JavaFX applications, JavaFX is automatically downloaded via Maven:

```
{
  "jdeploy": {
    "javaVersion": "21",
    "jdkProvider": "jbr",
    "jbrVariant": "jcef",
    "javafx": true
  }
}
```

Use Cases

- **Applications requiring embedded browsers**: JCEF provides Chromium-based browser embedding
- **Enhanced rendering**: JBR includes improved HiDPI support and rendering optimizations
- **IDE-like applications**: Same runtime used by IntelliJ IDEA and other JetBrains IDEs
- **Advanced GUI features**: Better support for complex graphical applications

Increased Default File Limit on macOS

The default file descriptor limit on macOS has been increased from 10,240 (the previous Java default) to the kernel maximum (typically around 138,000) to prevent "Too many open files" errors in applications with high file I/O or network connection requirements.

What Changed

The launcher now automatically raises the file descriptor soft limit to the kernel hard limit on macOS before starting the JVM. This applies to all applications without requiring configuration.

Override Behavior

Applications can still specify custom limits using the `-Djdeploy.file.limit` property:

```
{
  "jdeploy": {
    "args": [
      "-Djdeploy.file.limit=16384"
    ]
  }
}
```

Platform-Specific

This default only applies to macOS. Linux and Windows are unaffected.

Improved JavaFX Version Configuration

The `javafxVersion` property now supports more flexible version specifications, enabling the use of early access and custom JavaFX versions.

Enhanced Version Support

Applications can now specify:

- **Specific versions**: `"25.0.1"`
- **Early access versions**: `"25-ea+29"`
- **Version ranges**: `"[25.0.0,26.0.0)"`

Example Configuration

```
{
  "jdeploy": {
    "javaVersion": "21",
    "javafx": true,
    "javafxVersion": "25-ea+29"
  }
}
```

JRE Selection

When `javafxVersion` is specified, the launcher ensures only JREs without bundled JavaFX are selected, preventing conflicts between Maven-downloaded JavaFX and JRE-bundled versions.

Installer Window Icon

The installer window now uses the application's custom icon instead of the default Java icon.

What Changed

Previously, the installer window displayed the generic Java icon in the window title bar (Windows/Linux) and taskbar. Now it displays the application's custom icon throughout the installation process, providing a more branded experience.

Affected Platforms

- **Windows**: Icon appears in window title bar and taskbar
- **Linux**: Icon appears in window title bar and taskbar
- **macOS**: Uses app bundle icon (no change)

Bug Fixes

Architecture-Specific Package Installation

Fixed conflicts when installing both x64 and ARM64 versions of applications on systems that support both architectures (macOS with Rosetta, Windows on ARM).

Problem: Previously, both architectures shared the same package cache directory, causing the first installed version to be used for both architectures.

Solution: Package caches are now architecture-specific:

- `~/.jdeploy/packages-x64/` for x64 binaries
- `~/.jdeploy/packages-arm64/` for ARM64 binaries
- `~/.jdeploy/gh-packages-x64/` for GitHub packages (x64)
- `~/.jdeploy/gh-packages-arm64/` for GitHub packages (ARM64)

Each architecture now maintains its own package cache, ensuring correct native resources and binaries are used.

macOS x64 Splash Screen Crash

Fixed a regression causing crashes during splash screen display on macOS Ventura and earlier when running x64 builds.

Problem: Window operation ordering in the native webview code caused SIGSEGV crashes on older macOS versions when displaying the splash screen.

Solution: Restored the original window operation sequence that is compatible with all macOS versions.

Affected Versions: macOS Ventura (13.x) and earlier on Intel (x64) processors. macOS Sequoia and ARM64 builds were unaffected.

Technical Details

JBR Implementation

- Zero-dependency GitHub API integration for JBR release discovery
- Cache-redirector.jetbrains.com download source for global distribution
- Automatic home directory detection for both standard and macOS bundle structures
- Platform-specific build tags for cross-platform compatibility

File Limit Changes

- Implemented using `syscall.Setrlimit` on Unix-like systems
- Automatically raises to kernel hard limit on macOS
- Non-fatal failures with warning messages
- No-op on Windows (uses different handle limit mechanisms)

Architecture-Specific Packages

- Runtime architecture detection using `runtime.GOARCH`
- Automatic suffix application (`-x64`, `-arm64`)
- Backward compatible (existing installs unaffected)
- Applies to both npm and GitHub package sources

macOS Splash Screen Fix

- Window operation reordering for compatibility with older macOS versions
- Maintains splash screen visibility across all macOS versions
- No functional changes to splash screen behavior

Migration Guide

Upgrading from 5.3

jDeploy 5.4 is backward compatible with 5.3. No changes are required for existing applications.

Using JetBrains Runtime

1. Update package.json with JBR configuration
2. Choose appropriate variant (typically `jcef` for GUI applications)
3. Test on target platforms
4. For JavaFX applications, ensure `javafx: true` is set

Custom JavaFX Versions

1. Specify `javafxVersion` in package.json
2. Use format matching Maven Central artifacts
3. Test with specified version
4. Document version requirements

Architecture-Specific Installations

No migration required. New installs will automatically use architecture-specific directories. Existing installations in non-suffixed directories will remain separate from new installs.

Best Practices

JBR Usage

- Use `jcef` variant for applications with embedded browser needs
- Use `standard` variant for applications without JCEF to save space (~300-400MB vs ~200MB)
- Use SDK variants only when development tools are needed
- Test on all target platforms, as JBR behavior may differ from Zulu/Adoptium

File Limits

- The new default (kernel hard limit) is sufficient for most applications on macOS
- Override only if you need a specific limit different from the kernel maximum
- The kernel hard limit is typically around 138,000 on modern macOS systems
- Document file descriptor requirements in application documentation

JavaFX Versions

- Use specific versions for production deployments
- Test early access versions thoroughly before production use
- Document JavaFX version dependencies
- Consider compatibility with target Java versions

Known Issues

- JBR download requires internet connectivity (same as other JRE providers)
- JBR GitHub API is subject to rate limiting (60 requests/hour unauthenticated)
- Architecture-specific packages consume additional disk space during migration period

Resources

- **Official Website**: https://www.jdeploy.com/
- **Download**: https://github.com/shannah/jdeploy-desktop-gui/releases/latest
- **Documentation**: https://www.jdeploy.com/docs/manual/

- **GitHub Repository**: https://github.com/shannah/jdeploy
- **Support**: GitHub Issues and Discussions
- **JetBrains Runtime**: https://github.com/JetBrains/JetBrainsRuntime