# Java™ 2 Platform, Standard Edition v1.3 on the Solaris™ Operating Environment

*A Technical White Paper*

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Please
Recycle

# Contents

CHAPTER 1

# Overview

This document is written primarily for Solaris™ Operating Environment developers and deployers who are already using Java™ technology to develop products and technologies. It may also be useful to developers and deployers interested in using Java 2 Platform, Standard Edition (J2SE™) software with the Solaris Operating Environment. This document provides a high-level overview of J2SE v1.3 software, including technical descriptions of its features, and the benefits of upgrading to J2SE v1.3 software on the Solaris platform from previous versions of Java technology. By upgrading to J2SE v1.3 technology on the Solaris 8 platform, developers and deployers will experience an easier transition to J2SE v1.4 software, which supports 64-bit CPUs as well as a number of other new features.

J2SE v1.3 software for the Solaris Operating Environment provides a comprehensive development, test, and deployment platform that delivers significantly improved features, quality, performance, and scalability. Tightly integrated with the latest technologies in Solaris software, J2SE software offers a solid foundation for enterprise-grade Java technology application, development, and deployment.

The Solaris Operating Environment leads the industry as the premier, mission-critical platform for deploying Java technology-based applications. Solaris 8 software improves the server-side performance, stability, and scalability that data centers rely on, while retaining the agility today's businesses need to succeed. Java technology is tightly integrated into the operating environment, and can leverage its multithreaded capabilities to deliver the scalability, performance, and stability needed for enterprise applications.

This feature-rich release includes many enhancements. Created to address the diverse needs of both client and server environments, it includes a new Java virtual machine (JVM™) implementation, tuned libraries throughout the platform, enhanced security, and improved enterprise services for interacting with corporate resources, including directory and database interfaces. Additional functionality is available with the addition of Java Optional Package software, including Java Media Framework, XML, Java 3D™, and others. J2SE v1.3 software also provides a stronger technology foundation with improved Java HotSpot™ technology. A new Java HotSpot virtual machine (VM) enables more aggressive utilization and optimization of Solaris enterprise capabilities.

J2SE technology provides an exceptional cross-platform development and deployment environment for Web-enabling businesses. In particular, J2SE v1.3 software for the Solaris platform offers:

- Comprehensive enterprise services. Many enterprise service APIs are included in the base package — they are not optional as in prior releases. Solaris 8 software offers copackaged products that link to these service APIs. Complete feature stacks in database, directory, and security are provided.
- Better performance, scalability, reliability, and overall quality. This latest release has been strengthened with more rigorous quality assurance and beta testing to deliver greater code integrity. Performance and scalability levels have also been raised substantially over previous releases.
- New technology foundation. This release contains improved Java HotSpot technology, with new client and server virtual machines. Many improvements have been made to the overall virtual machine, and a new client compiler enhances performance for client applications. Building on this new technology foundation, it is expected that future J2SE releases on the Solaris platform will deliver even greater capabilities for the enterprise.

CHAPTER 2

# J2SE Technology on the Solaris Platform Features

The Solaris 8 software bundle contains a number of integrated feature stacks that provide an optimized interface to J2SE v1.3 services. Users and developers can take advantage of key enterprise services more quickly, rather than putting the pieces together on their own. In addition, many features have been added to the Java technology core. These enhance the overall capabilities of Java 2 Platform, Enterprise Edition (J2EE™) software, and make it more available. Key feature stacks include (with varied license terms):

- Java Naming and Directory Interface™ (JNDI) extension to bundled directory server for complete directory stack
- RSA / X.509 to bundled certificate server for complete security stack
- JDBC™ technology to bundled Oracle8i for complete database stack
- J2SE technology now supports Motif version 2.1
- Thread Model

## Java Naming and Directory Interface™ (JNDI)

JNDI provides Java platform-based applications with a unified interface to multiple naming and directory services in the enterprise. It is designed to be independent of any specific naming or directory service implementation. A variety of services — new, emerging, and already deployed — can be accessed in a common way. JNDI includes support for event notification and LDAP v3 extensions and controls, as well as class libraries and service providers for LDAP, the Common Object Request Broker Architecture (CORBA) Object Services (COS) Naming Service, and the Java Remote Method Invocation (RMI) Registry.

- LDAP – This protocol is an Internet standard for accessing directory services. The JNDI/LDAP service provider offers access to servers implementing LDAP protocols. The LDAP service provider implements the basic features for LDAP access. Additional functionality, such as support for a number of popular LDAP controls, SASL authentication, and the storing and reading of RMI and CORBA objects, can be added to the basic provider by installing a booster pack, available for download at the JNDI Web site.
- COS – This name server stores CORBA object references. It can be accessed from CORBA applications that use the COS Naming package (`org.omg.CORBA.CosNaming`). The JNDI/COS naming service provider implements the `javax.naming.Context` interface on top of the COS Naming package. This allows applications to use JNDI to access the COS Name Server. JNDI may also be used to access other naming and directory services in addition to the COS Name Server, thereby offering the CORBA application a single interface for accessing several naming and directory services.
- The RMI registry service provider allows JNDI applications to access remote objects registered with the RMI registry. Given the location of a registry, the provider will create a naming context with bindings for the objects registered there. Such a context may be bound into another JNDI-accessible namespace (such as LDAP, for example). References to individual remote objects may likewise be bound into another namespace. A key benefit of binding registry contexts into other namespaces is location-independent access to remote objects: the RMI clients do not need to know the registry's host name or port number. RMI servers may take advantage of this to advertise their services to potential clients. In addition, remote objects can be linked into the same enterprise directory that is used to access information about people, organizations, and network resources.

# Security

- J2SE v1.3 software for the Solaris Operating Environment has been enhanced with a number of improvements for increasing the security of applications and services. This includes better support for RSA signatures and X.509 certificates in Java applications that are based on technology, as well as those that are not.
- Digital certificates support includes a broad range of enhancements relating to digital certificates that have been made to the J2SE v1.3 environment, improving the ability to work in heterogeneous situations. There is also increased compliance with open standards relating to security protocols.
- The Java 2 SDK, Standard Edition, v1.3 now fully interoperates with VeriSign's code signing certificates. Keytool is now able to import VeriSign certificates.
- `Jarsigner` can now verify Netscape™ signed jar files.
- A new class `java.security.spec.RSAKeyGenParameterSpec` makes it possible to specify both the size of the public modulus before it is generated, as well as the value of the public exponent when generating an RSA keypair.

- The handling of X509 certificates has been updated to include support for multiple Attribute/Value Assertions (AVAs) within a Relative Distinguished Name (RDN).
- A new signature verification method has been added to `java.security.Signature`, and this new method is FIPS 140-1 compliant.
- Enhancements have been made to support all X.520 attributes that are either mandated or recommended by PKIX RFC 2459.

## Remote Method Invocation Over Internet Inter-ORB Protocol (RMI-IIOP)

CORBA is a standards-based, distributed computing environment that has been widely adopted in backend systems. Java applets and applications can interoperate in this environment, reducing application development time and enabling the reuse of existing software.

CORBA technology in J2SE v1.3 software provides the support for the Java Interface Definition Language (IDL) and RMI over IIOP. Java IDL is the Java implementation of CORBA as specified by the Object Management Group (OMG). Java IDL provides a framework for rapid development of distributed applications.

The RMI over IIOP extends the RMI programming model by utilizing the Internet Inter-ORB Protocol (IIOP) as the communications protocol for interoperability. Using this technology, application programmers can write distributed applications completely in the Java language using the RMI programming model, and then use the IIOP protocol for interoperating with the application components written with other CORBA-compliant languages, such as C++, COBOL, and Smalltalk.

RMI-IIOP combines the best features of RMI with the best features of CORBA. It provides program authors with a powerful environment for distributed programming using the Java platform.

## Motif Support

In the Solaris 7 and Solaris 8 Operating Environments, J2SE v1.3 software uses Motif 2.1. The Solaris 2.6 Operating Environment still uses Motif 1.2, an earlier version. Because Motif 2.1 is the supported version of Motif going forward, switching to it enables the Java platform to take advantage of future features and bug fixes in Motif.

Motif 2.1 is compatible in both functionality and performance to Motif 1.2, except that it offers two additional features that may be of use for the Java 2 platform:

- Motif 2.1 is multithread safe. Applications implemented partly in native code that use Motif directly through the `jawt` interface or the `DrawingSurface` interface can benefit from this feature.
- Motif 2.1 includes new Complex Text Layout (CTL) software for advanced locale support.

## Thread Model

J2SE v1.3 technology offers Solaris software-specific thread synchronization. This makes it easier to write multithreaded programs with fine-grained locking. The Java HotSpot technology incorporates a unique synchronization implementation that substantially boosts performance. This mechanism offers ultra-fast, constant-time performance for all uncontested synchronizations, which dynamically comprise the great majority of synchronizations. The Java HotSpot synchronization implementation fully leverages the multiprocessing capability of the Solaris software, and exhibits excellent multiprocessor performance characteristics.

CHAPTER 3

# J2SE Software Features

J2SE v1.3 technology delivers significant performance gains and improved Web deployment for enterprise, client-side Java applets and applications. It includes a new client Java virtual machine (JVM), tuned libraries throughout the platform, and enhancements to the Java Plug-In software for improved Web browser deployments.

## Performance Enhancements

J2SE v1.3 software takes full advantage of the strengths and benefits of Solaris software. Many enhancements have been made to improve the performance of the Java 2 SDK and Java runtime environment (JRE). J2SE software delivers excellent performance and scalability to meet the demanding requirements of enterprise application environments. A new JVM, using Java HotSpot technology, incorporates compilers tuned for both client and server environments, providing a foundation for enhanced performance.

The Java HotSpot technology is tuned to the Solaris Operating Environment, offering fast thread synchronization and locking. In addition, J2SE v1.3 technology leverages the Solaris multithreaded design to scale to dozens of CPUs. While there are enhancements in virtually every major area in this release of J2SE technology, the greatest improvements come from the following areas:

- Java HotSpot client and server virtual machines
- Improvements to startup time and memory footprint
- Improvements to program execution speed

# Performance Improvement Metrics

J2SE v1.3 software performs a wide array of computation-intensive tasks with much greater speed than previous JRE implementations. Internal testing shows that this version performs more than 45 percent faster. Figure 1 shows the composite score for several benchmarks that exercise different areas of the JVM.

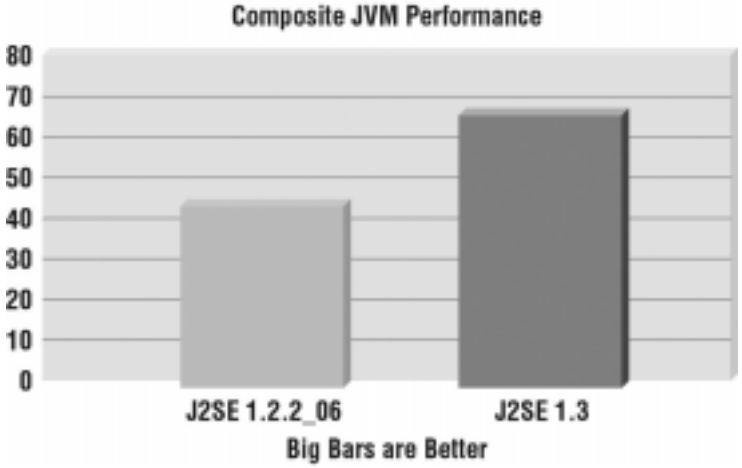


Figure 1: J2SE v1.3 software offers significantly faster performance over J2SE 1.22_06, the last production release of the J2SE platform.

# Java HotSpot Virtual Machine

- The Java HotSpot performance engine is the technology base for all future Java virtual machines running Solaris software. The Java HotSpot product line delivers the highest possible performance for Java applications. With this release of J2SE technology, there are now two virtual machines — a server-side offering and a version for client-side performance. These two solutions share the Java HotSpot runtime environment, but have different compilers suited to the distinctly different performance characteristics of clients and servers.

- The Java HotSpot Client VM serves as a replacement for both the “classic” virtual machine and the just-in-time (JIT) compilers used by previous versions of the Java 2 SDK. The Client VM offers improved runtime performance for applications and applets. The Java HotSpot Client VM has been specially tuned to reduce application startup time and memory footprint, making it particularly well suited for client environments.

  The Client VM doesn't try to execute many of the more complex optimizations performed by the compiler in the Server VM, and for that reason it requires less time to analyze and compile a piece of code. This means that the Client VM starts up faster, and requires a smaller memory footprint.
- The Java HotSpot Server VM is similar to the Client VM, except that it has been specially tuned to maximize peak operating speed. It is intended for executing long-running server applications. For these, achieving the fastest possible operating speed is generally more important than having the fastest possible startup time or a smaller footprint.
- The Server VM contains an advanced adaptive compiler that supports many of the same types of optimizations performed by C++ compilers, as well as some optimizations that can't be done, such as aggressive inlining.

Both solutions deliver extremely reliable, secure, and maintainable environments, to meet the demands of today's enterprise customers. These two systems are different binaries, and are essentially two different compilers interfacing to the same runtime system.

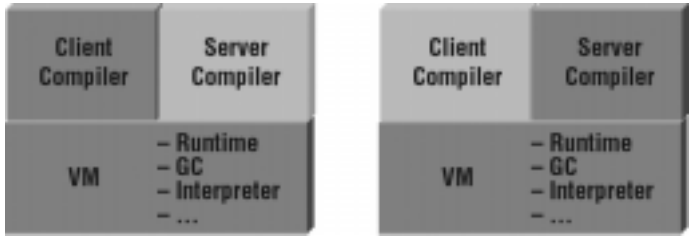


Figure 2: The Java HotSpot Client VM (above left) and the Java HotSpot Server VM (above right) use different compilers, but otherwise interface to the same virtual machine with the same garbage collection (GC) routine, interpreter, thread, and lock subsystems.

The Client VM is optimal for applications that need fast startup times, small footprints, or high GUI performance, while the server system is typically recommended for applications where sustained performance is most important. Other differences include the compilation policy used, heap defaults, and inlining policy.

J2SE v1.3 software for the Solaris platform contains both of the above systems in the distribution. The programmer chooses a system by specifying `-server` or `-client`.

Features of Java HotSpot technology that are common to both VM implementations include:

- Adaptive compiler – Applications are launched using a standard interpreter, but the code is analyzed as it runs to detect performance bottlenecks, or "hot spots." The Java HotSpot VM compiles those performance-critical portions of the code for a boost in performance, while avoiding unnecessary compilation of seldom-used code; typically, this is most of the program code. The VM also uses the adaptive compiler to decide — on the fly — how best to optimize compiled code with techniques such as inlining. The runtime analysis performed by the compiler eliminates guesswork when determining which optimizations will yield the largest performance benefit.
- Improved memory allocation and garbage collection – The Java HotSpot VM allocates memory for objects more quickly than previous virtual machine implementations. The VM also has a state-of-the-art garbage collector that is faster and more efficient than previous versions. Java HotSpot technology provides an exact, generational, incremental garbage collector. This means smaller memory usage and better performance due to increases in the speed and efficiency of garbage collection, and fewer user-perceivable pauses during garbage collection.
- Thread synchronization – The Java programming language allows for use of multiple, concurrent paths of program execution — threads. The Java HotSpot VM provides a leaner, speedier thread-handling capability that is designed to scale readily for use in large, shared-memory multiprocessor servers.

## Improvements to Startup Time and Memory Footprint

This release of J2SE technology features changes that help to reduce startup time and decrease the amount of memory used by the Java 2 SDK and JRE.

- Swing Class Loading – In version 1.2, Swing classes were loaded at startup regardless of whether the classes were needed immediately. In version 1.3, loading of some Swing classes is delayed until they are actually needed by the runtime environment. This change reduces startup time. In version 1.2, class JTable loaded the internationalization code at startup, even if it was not needed immediately. In version 1.3, the loading of international date and number formatting code by JTable is deferred until the internationalization code is needed by the runtime environment.

- JAR File Memory Usage – The way JAR files are indexed in memory has been improved to reduce the memory cost of loading a JAR file. The class libraries in `rt.jar`, for example, occupied about 420 Kbytes of memory when loaded in version 1.2, but require only about 100 Kbytes in version 1.3.
- Reduce Memory Footprint Associated with Strings – In 1.2, String loaded the much larger `java.io.ObjectStreamClass` for this purpose. In version 1.3 of the SDK, `java.lang.String` has been modified to load the relatively small `java.io.ObjectStreamField` class for handling serialization requirements.
- The size of internal VM data structures used to represent classes, methods, and fields by the virtual machine has been reduced.

## Improvements to Program Execution Speed

A number of changes in this release improve the program execution speed and general run-time performance of the Java 2 SDK and JRE. These include:

- Improved Performance for `readUTF` and `writeUTF` – The implementations of the `readUTF` and `writeUTF` methods of `DataInputStream` and `DataOutputStream` have been modified to enhance performance.
- Improved Performance of `JScrollPane` Painting – The implementation of `JScrollPane` no longer causes excessive painting and screen flashing when double buffering is not used.
- Improved JTable Performance – Previously, the operations of adding columns or recalculating column widths did not scale well as the number of columns became large. The implementation has been changed to improve scalability.
- Paint Coalescing – Through an improved paint coalescing algorithm, areas that require repainting are coalesced into a single, non-rectangular repaint area as they are posted to the event queue. This has the effect of batching repaints, so multiple repaints can be executed simultaneously in a more efficient manner than in previous releases.
- Internal Frame Blitting – By using `Graphics.copyArea` calls, Swing's `DefaultDesktopManager` now requires far less redrawing when frames are dragged.
- Performance Enhancements in `BigInteger` – Class `java.math.BigInteger` has been reimplemented in 100% Pure Java™ code, instead of the Plumb C library used in previous implementations. The new version performs all standard operations much faster — as much as five times or more — depending on the operation being performed.
- Provision for Better Performing Numeric Operations – The Java 2 platform now contains two classes that provide APIs for performing general numeric operations. These are:

- `java.lang.StrictMath` – Class `java.lang.StrictMath` is a renaming of Class `java.lang.Math` from previous versions of the Java platform, but otherwise retains the previous specification of Math. In particular, it is defined to return bit-for-bit reproducible results in all implementations.
- java`.lang.Math` – Class `java.lang.Math` provides the same set of APIs as does Class `StrictMath`. However, unlike some of the numeric functions of Class `StrictMath`, all implementations of the equivalent functions of Class `Math` are not defined to return the same results, bit-for-bit, but may vary within specified constraints. This relaxation enhances performance where strict reproducibility is not required.

## Web Deployment

J2SE v1.3 software contains a number of improvements that make it easier to deploy applets to the Web. Java Plug-In software offers a consistent source code base — and consistent applet behavior — across Netscape browsers on Solaris, Microsoft Windows, and Macintosh clients, as well as Internet Explorer for Microsoft Windows clients. Java applets may now remain stored on local disk, significantly improving performance on subsequent visits to a Web page.

## Java Plug-In Software

Java Plug-In software enables developers to write Java 2 SDK v 1.3 software-based applets and immediately deploy them to existing Netscape Navigator™ or Microsoft Internet Explorer users. Java Plug-In software enables enterprise customers to run applets on their intranet Web pages using JRE instead of the Web browser's default virtual machine. This provides a Java technology-compatible environment for today's widely adopted Web browsers — ensuring consistency and reliability when running applets.

## Applet Caching

This new feature stores deployed applets on the local drive until they are updated, dramatically improving applet performance while reducing network traffic.

Applet caching ensures that applets are not downloaded unnecessarily by a browser every time a user references them. Java Plug-In supported caching in previous versions, but it used the same cache used by all other Web files. This was acceptable for casual applet usage, but larger applets can often get flushed from the cache to make room for other files, as the browser has no specific knowledge that an applet file might be needed in the future. The result was that the Java Plug-In caching strategy failed in large business applets in version 1.2.

This has been remedied in the latest J2SE release. J2SE v1.3 software introduces an alternative form of applet caching that allows an applet deployer to decide if an applet should be "sticky," that is, if it should stay on the disk in a secondary cache that the browser cannot overwrite. Once a sticky applet resides in this secondary cache, it is downloaded only when it is updated on the server. Otherwise, the applet is always available locally for quick loading.

Other improvements in this area include:

- Fast Loading of Applet Support Classes – Jar files containing support classes for applets can now be placed in the JRE Plug-In `lib/applet/` directory. This reduces startup time for large applets by allowing applet classes to be preloaded from the local file system by the applet class loader, providing the same protections as if they had been downloaded over the Internet.
- Automatic Installation of Extensions – Installs the right version of the extension on the end-user machine, without requiring it to be downloaded repeatedly each time an applet is loaded. This enables faster startup of applets.
- Jar Indexing – Enables developers and deployers to break functionality into different Jar files and ensure that only those that are needed for a given run of the applet are actually downloaded.

CHAPTER 4

# Additional J2SE Software Enhancements

## Java Sound

The Java 2 SDK includes the Java Sound API which defines a set of standard classes and interfaces for low-level audio support. The Java Sound API enables Java programs to capture, process, and play audio and Musical Instrument Digital Interface (MIDI) data. The Java Sound API is supported by an efficient sound engine that guarantees high-quality audio mixing and MIDI synthesis capabilities for the platform. Additionally, the API defines a set of service provider interfaces which developers and deployers may use to extend the capabilities of the current implementation. Users can install modules that provide support for additional file formats, codecs, and devices.

The Solaris software implementation of this API supports the following features:

- Support for audio in applications as well as applets
- API for capturing, processing, and playing back audio and MIDI data
- Support for audio file formats AIFF, AU, and WAV
- Support for music file formats MIDI Type 0, MIDI Type 1, and Rich Music Format (RMF)
- Support for sound formats 8-bit and 16-bit audio data, in mono and stereo, with sample rates from eight kHz to 48 kHz
- Support for linear, a-law, and mu-law encoded data in any of the supported audio file formats
- MIDI wavetable synthesis and sequencing in software, and access to hardware MIDI devices
- An all-software mixer that can mix and render up to 64 total channels of digital audio and synthesized MIDI music

Note that the above list describes the reference implementation, not the API itself. The API permits flexible configuration of the audio and MIDI system, including ways for applications to ask the system exactly what resources are installed and available.

# Localization

- J2SE technology complements the international capabilities of the Solaris platform, offering localized support for a number of countries. The following table indicates which locales are supported by J2SE v1.3 software. Support is indicated with an X.

| **Locale** | **Solaris Software** | | |
|---|---|---|---|
| | **2.6** | **7** | **8** |
| **German** | | | |
| de_DE.UTF-8 | | X | X[2] |
| de.ISO8859-15 | X[1] | X[1] | X |
| de | X | X | X |
| **Spanish** | | | |
| es_ES.UTF-8 | | X | X[2] |
| es.ISO8859-15 | X[1] | X[1] | X |
| es | X | X | X |
| **French** | | | |
| fr_FR.UTF-8 | | X | X[2] |
| fr.ISO8859-15 | X[1] | X[1] | X |
| fr | X | X | X |
| **Italian** | | | |
| it_IT.UTF-8 | | X | X[2] |
| it.ISO8859-15 | X[1] | X[1] | X |
| it | X | X | X |
| **Swedish** | | | |
| sv_SV.UTF-8 | | X | X[2] |
| sv.ISO8859-15 | X[1] | X[1] | X |

| | | | |
|---|---|---|---|
| sv | X | X | X |
| **Korean** | | | |
| ko.UTF-8 | | | X[2] |
| ko | X | X | X |
| **Japanese** | | | |
| ja_JP.UTF-8 | | X | X[2] |
| ja | X | X | |
| ja_JP.PCK | X | X | X |
| **Simplified Chinese** | | | |
| zh.UTF-8 | | | X[2] |
| zh | X | X | X |
| zh.GBK | | X | X |
| **Traditional Chinese** | | | |
| zh_TW.UTF-8 | | | X[2] |
| zh_TW | X | X | X |
| zh_TW.BIG5 | | X | X |

**TABLE 1**

Notes:
1. The Euro currency symbol does not display on Intel Architecture in ISO8859-15 locales using Solaris 2.6 and Solaris 7 software.
2. An error message appears unless the following optional font packages are installed: SUNWi2of, SUNWi4of, SUNWi5of, SUNWi7of, SUNWi9of.

# Ease of Development

A number of new features have been added to make application development easier and more productive, including:

- The robot API for automated testing and accessibility support helps automate testing of JFC, Swing, and AWT-based Java applets and applications. Automation increases the speed of testing and enables programmers to more quickly update and iterate software during the development process. In addition, the `Robot` API can be used by Java applets and applications that need automation, for example, computer-based training and accessibility-enabled applications.
- Java Platform Debugger Architecture (JPDA) gives third-party development tool providers access to internal JVM information. Also, Java developer tools that integrate JPDA debugging functions can be used to create software with fewer defects.

- The new `javac` has been completely rewritten for J2SE v1.3 software, and is significantly faster — saving time when compiling software that is written in the Java language into bytecode.

## Compatibility with Previous Versions of J2SE Technology

A great deal of attention has been placed on compatibility with previous releases. J2SE v1.3 software is highly compatible with previous releases of the Java technology deployment platform, and J2SE v1.2x users should be able to upgrade with minimum effort. Details on compatibility with previous versions can be found in the *J2SE Developer's Guide for Solaris*, available at http://java.sun.com/j2se/1.3/compatibility.html#incompatibilities1.3.

## System Configuration Requirements

J2SE v1.3 software is available on the Solaris Operating Environment for both SPARC™ and Intel Architecture platforms. It runs with the Solaris 2.6, 7, and 8 Operating Environments, and requires approximately 60 Mbytes of free disk space for installation.

CHAPTER 5

# Future Directions

J2SE v1.4 software is being designed through the Java Community Process℠ (JCP), and the specifications are available on the JCP at http://java.sun.com. In addition to improved reliability and enhanced scalability, J2SE v1.4 will feature:

- Improved scalability
  - Full 64-bit support for the JVM
  - SPARC v9 — UltraSPARC™ III tuning
  - IPv6
  - VM resource sharing
  - Scalable I/O API for sockets and files
- Reliability, availability, scalability (RAS) enhancements
  - APIs for logging and failover
  - Improved remote monitoring capabilities
  - Low-resource and fatal-error handling
- Performance
  - Faster compiler
  - Improved garbage collection performance
  - UltraSPARC III tuning
- Deployment
  - Java Web Start
  - Installer API and a preference API
- Development
  - Integration of Forte™ for Java™, Community Edition IDE into J2SE technology
  - JPDA class file redefinition
  - Application control of thread stack size
- Improved XML support, including parsing, databinding, and XSLT support

- Improved security features, including JSSE, JAAS, JCE, CertPath, JGSS, and Kerberos integration into the core
- Improved international support, including Unicode and additional locales
- Improved name service support

For complete details on future versions of J2SE software, please see the Java Community Process link on the Java Web site (http://java.sun.com). The next version of J2SE technology will be a synchronized release across the Solaris Operating Environment for SPARC and Intel Architecture platforms, as well as Linux and Microsoft Windows.

CHAPTER

# Summary

Delivering the best from both worlds, Java technology on the Solaris Operating Environment provides the agility and predictability required to make today's enterprises more competitive, dynamic, and profitable. As the premier, enterprise-ready Java deployment platform, it helps increase competitive advantage by delivering a flexible application platform, scalable server platform, and reliable dot-com environment.

Because J2SE v1.3 software is designed to be forward-compatible to J2SE v1.4, users who upgrade to version 1.3 should have a smooth transition to version 1.4 when it becomes available. J2SE v1.3 software on the Solaris 8 platform is solid, feature-rich, and highly scalable. It is designed to serve as the Java platform of choice for enterprise application development and deployment.

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

http://www.sun.com/solaris

February 2001