

Java™ 2 Platform, Micro Edition (J2ME™) Web Services

A Technical White Paper
July 2004



© 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Java, Java Community Process, JCP, J2EE, J2ME, J2SE, and The Network Is The Computer are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Table of Contents

Introduction J2ME Web Services	1
The Promise of Web Services	2
An Overview of Web Services	3
Web Services Solutions	3
Using Web Services	4
Scope of This Paper	5
The Web Services Client Model	5
Devices as Consumers of Web Services	6
The J2ME Web Services Specification	7
The J2ME Client Platform	7
The JAXP Optional Package	7
The JAXP Parser	8
The JAX-RPC Optional Package	8
Contents of the JAX-RPC Optional Package	8
The Service Provider Interface	8
JAX-RPC and the Stub Generator	9
Web Services in the Wireless Marketplace	10
Glossary	11

Chapter 1

Introduction

To understand Web services, it helps to think back to the childhood swimming game Marco Polo. In that game, a child is blindfolded and tries to find other children, steered only by their voices. The blindfolded child does not know the location of the other children. Yet, Marco finds Polo. The mechanism used to locate the children works.

Though deceptively simple, the game of Marco Polo provides an easy analogy for the complex field of Web services. Services somewhere on the network are found by clients who seek them; clients seek services, but without knowing where they are. Like the game of Marco Polo, the client must have a mechanism for finding a specific service. The mechanism for doing this is the Web services paradigm.

Chapter 2

The Promise of Web Services

In simplest terms, a Web service is an application. That is, a Web service provides a defined set of functionality to achieve a specific end, for example, a mapping service that takes two addresses and tells you how to travel between them. Before the adoption of the Internet, a client could only use an application if it was installed on a local machine. Today, with the emergence of the Java™ 2 platform and Internet standards such as the HyperText Transfer Protocol (HTTP), eXtensible Markup Language (XML), Remote Procedure Call (RPC), Web Services Description Language (WSDL), and Simple Object Access Protocol (SOAP), it is possible to access an application on any computer, anywhere in the world.

The promise of Web services is to put any application — regardless of the platform on which it is developed or the architecture on which it is deployed — within the reach of any client. It seeks to create an environment where any networked client, running on any type of device, can find a service on the network and use it as though it was a local service.

Today, building and deploying applications on the Internet is a multibillion dollar industry. In this dynamic environment, Web services give companies an unprecedented opportunity to make use of the functionality and services provided by others. From airline Web sites that offer car rental and hotel services with up-to-the-minute rates, to the doctor who has access to online medical services from his or her cell phone, Web services are becoming a part of daily life.

This paper provides a brief overview of the field of Web services and then looks at how the Java 2 Platform, Micro Edition (J2ME™) Web Services Specification fits into the picture.

Chapter 3

An Overview of Web Services

To fulfill the promise of Web services requires cooperation. Companies like Sun Microsystems and others work closely with bodies such as the World Wide Web Consortium (W3C) and the Java Community ProcessSM (JCPSM) program to develop new standards and protocols. Many difficult technical problems — such as connectivity, discovery, data description, and security — must be solved.

For example, what does it take for a client developer in California to find and use a remote Web service developed by a programmer in the Czech Republic? For this to happen, solutions to three different but interrelated problems must be in place:

1. How must an application be written and deployed so that it is accessible to clients on the network?
2. How does a client developer, who wants to use a certain type of Web service, find that service?
3. Once a service is located, how does a client developer access its functionality?

The solution to these problems provides a context for understanding the field of Web services.

Web Services Solutions

The first problem described — writing and deploying accessible applications — is best addressed by using the tools and techniques of the Java programming language. The compact nature of Java technology code, the inherent security of the Java Virtual Machine, and the standardized deployment methodologies of the Java 2 Platform, Enterprise Edition (J2EE™ platform) make the Java language particularly well-suited for Web services development and deployment.

The second problem — finding a remote service — is handled through the use of a network *registry*. A registry is a centralized location where developers post information about a service so that other developers can use it. Although there is more than one network registry available, the most widely used is the Universal Description, Discovery, and Integration (UDDI) registry.

The third problem — connecting with and using a network service — lies at the heart of the Web services paradigm. To address this requires a number of well-defined steps, and makes use of the following standard protocols:

- **eXtensible Markup Language (XML):** A portable document markup language used extensively in the Web services paradigm. It provides a transportable means of describing system and configuration information that can be passed between applications. Both SOAP and WSDL files are written using XML.

- **Simple Object Access Protocol (SOAP):** An XML-based protocol that provides an envelope for exchanging object data on the Internet. It uses standard formatting conventions and encoding rules, binds to HTTP as its transport mechanism, and relies upon RPC.
- **Web Services Description Language (WSDL):** An XML-based protocol used to describe a remote Web service and facilitate application-to-application communication. A WSDL file describes what a service does, how to invoke its operations, where it can be found on the Net, and its interface.
- **Remote Procedure Call (RPC):** An Internet protocol that allows data exchange between different types of systems. Used with SOAP, it allows one application to call methods on another application — without needing to know the underlying network implementation.

The relationship between these four protocols, how a developer uses them to connect to a remote service, and how they tie in with UDDI are explained in the following sections.

Using Web Services

Web services are designed to be platform and language independent. For client developers, this means they can make calls to the business logic of an Internet application and have the processing returned in a usable way, regardless of the way the client or Web service is implemented. The goal of Web services is to ensure seamless interaction between the requesting client and the responding Web service.

To properly use Web services, the following steps must take place (as shown in Figure 1):

1. The service developer must write an application that conforms to standard programming expectations (e.g., API interfaces are properly defined, methods are public, etc.).
2. The service developer must deploy this application in a manner that makes it available to other applications, for example, the application may be packaged as a J2EE WAR file and deployed to the J2EE Web container.
3. The service developer must define an abstract service description in the form of an XML-based WSDL file, describing the interfaces for the application. The WSDL file defines the endpoint needed for a client to make calls to the service.
4. The service developer must register (or “publish”) the service description to the UDDI registry. Registering a service requires the following:
 - a. Posting information about the company or developer who created the service
 - b. Categorizing the service so that it can be looked up; for example, if a service provides real-time pricing for corn futures, it might be categorized as an “e-commerce” or “commodities” service
 - c. Making the URL of the service description WSDL file available to client developers
5. A client developer goes to the UDDI registry and looks for a Web service that provides the functionality needed by his/her client application. Once a Web service is found, the client developer obtains the WSDL file for it.
6. The client developer uses the Web service’s WSDL file as input to a stub generator program. The stub generator outputs the code needed by the client application to make calls to the remote service.
7. The client developer incorporates the stub generator output code into his/her client application and makes calls to it. The stub generator code then handles the SOAP message packaging, HTTP binding, and RPC connectivity needed to connect to the remote Web service. When the Web service is done processing the request, the stub generator handles the response to the client application.

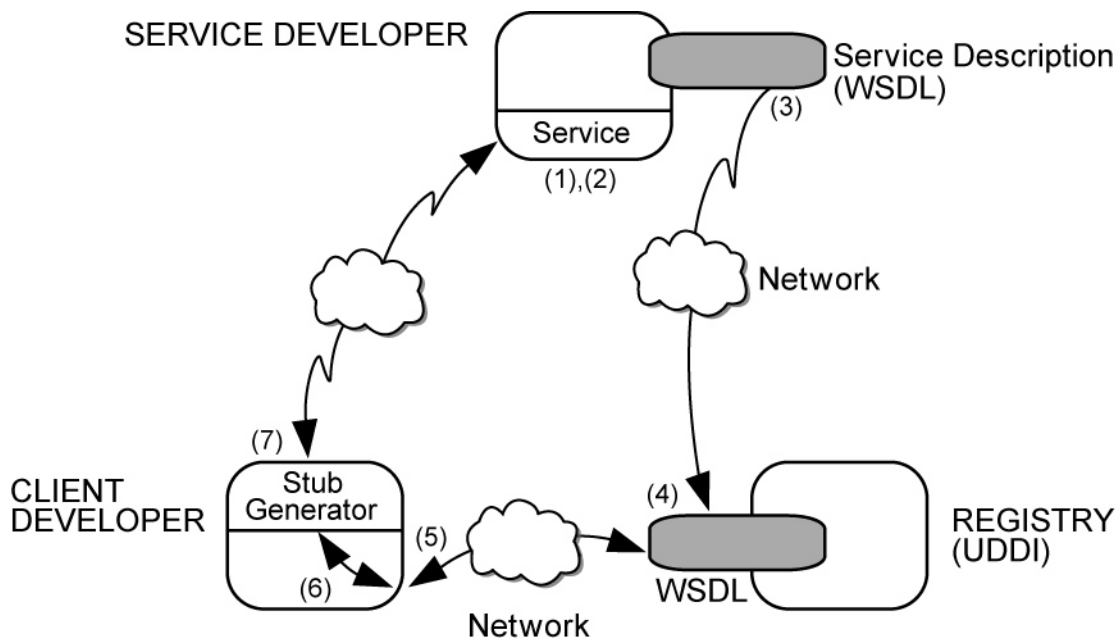


Figure 1. Steps in the Web Services Paradigm

Scope of This Paper

For the purpose of describing the J2ME Web Services Specification, further discussion of steps 1, 2, 3, 4, and 5 in Figure 1 are largely outside the scope of this paper. For more information on writing and deploying Java applications, visit java.sun.com. There are also a number of excellent books available that provide information on using the UDDI registry as well as the details of the SOAP, WSDL, XML, and RPC protocols.

The Web Services Client Model

As described in “Using Web Services,” once a client developer has obtained the WSDL file for a remote Web service, it becomes possible for the client to make calls to that service.

Note – The following steps expand on the actions described in Figure 1, Step 7.

The following is a typical interaction between a Web services client and a remote Web service (as illustrated in Figure 2):

1. A local application makes a procedure call to a local client component that has been developed using the WSDL file obtained from a remote Web service.
2. The local client component sets up an HTTP communication channel with the remote Web service, using the service endpoint defined in the WSDL file. A procedure call is sent to the remote service over the HTTP connection, using SOAP and RPC.

3. The remote Web service accepts the procedure call, performs some processing, and returns a response to the local client component via the HTTP communication channel, also using SOAP and RPC.
4. The local client component returns the response to the local application. The transaction appears to the local application as if it had occurred entirely on the client device.

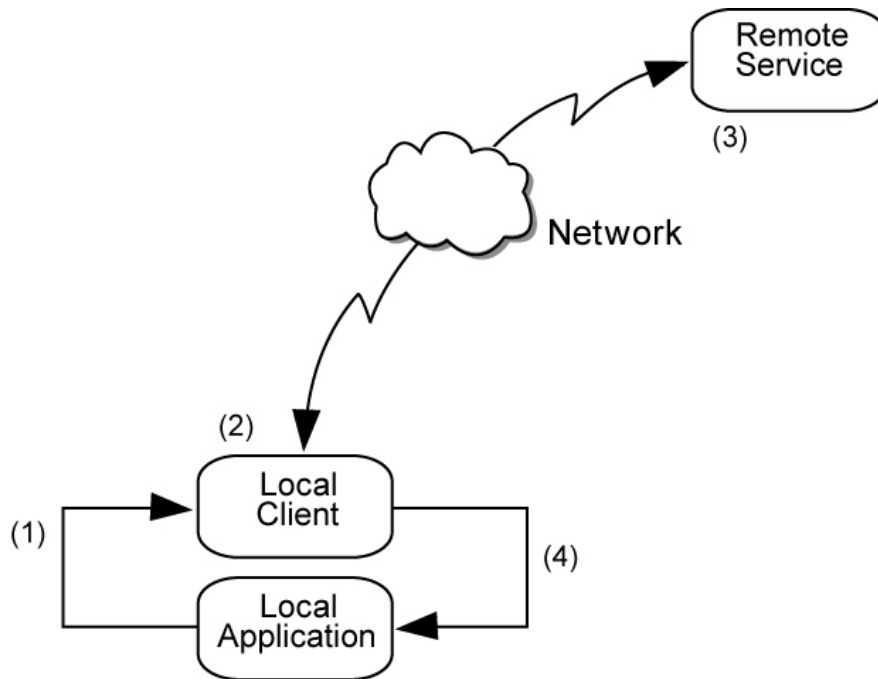


Figure 2. Client Device to Remote Service Transaction

Devices as Consumers of Web Services

The portable nature of Web services, and their ability to provide distributed functionality across a broad spectrum of the Internet, makes them well-suited for the quickly growing mobile marketplace. The increasing power and memory capacity of mobile devices make them ideal for the compact capabilities provided by Web services. Through the use of technologies such as SOAP, WSDL, XML, and RPC, mobile client developers can create applications that are as mobile and highly available as the cell phones and PDAs carried by their customers.

In this rapidly expanding mobile marketplace, client developers gain significant benefits by using the J2ME Web Services Specification.

Chapter 4

The J2ME Web Services Specification

The J2ME Web Services Specification is specifically designed to facilitate development of small, fast, and robust Java platform applications in the wireless marketplace. The J2ME Web Services Specification provides two XML-based optional packages that can be used to develop Java Web services for small, reduced-memory, handheld devices such as cell phones, PDAs, and pagers. These packages are:

- Java API for XML Processing (JAXP)
- Java API for XML-based RPC (JAX-RPC)

The J2ME Client Platform

The J2ME platform is a set of standard Java APIs defined through the Java Community Process (JCP). It provides a flexible user interface, robust security, and built-in network protocols. The J2ME platform can be tailored for a specific class of device by matching it with a specific configuration and profile, and can be extended for a specific market by adding additional optional packages.

The J2ME Web Services Specification supports the following:

- **Connected Limited Device Configuration (CLDC), v1.x.** CLDC is designed for small devices with 16-bit or 32-bit CPUs and 128 KB to 512 KB of memory.
- **Mobile Information Device Profile (MIDP), v1.x and 2.x.** MIDP is specifically designed for cell phones and entry-level PDAs, and provides the user interface, network connectivity, local data storage, and application management needed by these devices.

With CLDC and MIDP, the J2ME platform provides a complete Java runtime environment for wireless, handheld, and mobile devices. Installing the J2ME Web Services JAXP and JAX-RPC optional packages further extends this Java runtime environment by adding additional XML processing capabilities to the J2ME platform.

The JAXP Optional Package

The purpose of the J2ME Web Services JAXP optional package is to provide XML parsing support to the J2ME platform. The J2ME JAXP optional package is a subset of the Java 2 Platform, Standard Edition (J2SE™ platform) JAXP 1.2 API. The J2ME JAXP package has been reduced in size to fit the constraints of the J2ME platform and to run effectively in a reduced memory footprint.

Today, more and more developers are using XML as the standard means of passing data between client devices and back-end servers, or between client device and client device. Hence, the ability to parse XML data on the J2ME platform enables the development of XML-based, networked Web services in the client device market.

The JAXP Parser

The parser provided in the J2ME Web Services JAXP optional package is a non-validating parser whose primary purpose is to parse incoming XML documents and make the data contained in them available to the local device and the applications that run on it. The J2ME Web Services JAXP parser is designed to parse an XML document as an input stream rather than as a document tree. The behavior of the J2ME JAXP Parser is consistent with the XML 1.0 Specification. It supports the W3C XML namespace standard and a subset of the Simple API for XML Parsing (SAX) 2.0 interfaces.

The JAX-RPC Optional Package

The Java API for XML-based RPC (JAX-RPC) is an implementation of Remote Procedure Call (RPC) technology in the Java programming language, and is part of the Java 2 Platform, Enterprise Edition (J2EE platform). The JAX-RPC optional package subset, provided with the J2ME Web Services Specification, is a scaled-down version of JAX-RPC specifically for the J2ME platform tailored to run effectively in the reduced memory environment found in wireless and handheld devices.

The J2ME JAX-RPC subset enables developers to create portable clients that can easily exchange data between clients and back-end servers. Using XML and SOAP, JAX-RPC allows developers to dispatch RPC calls to Web services running on a different machine, a different network, or in a different language. Using RPC, developers can call methods on remote objects as easily as they can call them on local objects.

Contents of the JAX-RPC Optional Package

The J2ME Web Services Specification JAX-RPC optional package contains three parts:

- **JAX-RPC Subset APIs:** The Java APIs used to develop a client implementation of a Web service
- **Java Remote Method Invocation (RMI) APIs:** Java classes included with the J2ME Web Services Specification to satisfy dependencies of JAX-RPC on CLDC-based platforms
- **Java Web Services-specific Service Provider Interface (SPI):** Enables stubs to be portable and compatible across varying implementations of the Java Web Services Specification

The Service Provider Interface

The J2ME Web Services Specification provides only a single implementation of many that are possible; other implementers writing to the J2ME Web Services Specification may produce a somewhat different implementation based upon the same guidelines and APIs.

To ensure that any implementation of the J2ME Web Services Specification will work with any other implementation, an additional API is provided so that all generated stubs are platform independent. This API is the Service Provider Interface (SPI), and it formally defines the interface between the J2ME Web Services Specification and the stub. By formally defining this interface, it ensures that any stub generated by any J2ME Web Services implementation will work with any other implementation that has correctly implemented the SPI.

The API defined by the J2ME Web Services Specification SPI is not found in the J2EE implementation of JAX-RPC.

JAX-RPC and the Stub Generator

The Stub Generator is a JAX-RPC tool provided by the J2ME Web Services Specification. It is used to produce a stub, or *client-side proxy*, that can be called by an application to place calls to a remote Web service. The Stub Generator takes a WSDL file as input and outputs the necessary Java code needed to place calls to the remote Web service.

The purpose of the Stub Generator is to create the SOAP messaging structures, network connections, and underlying RPC coding necessary to place remote calls over the Web. This saves the developer from having to do the rigorous underlying code work.

Note – The following steps expand on the actions described in Figure 1, Step 6.

The interaction between the local application, the stub, the Service Provider Interface, and the Java Web Services implementation takes place according to the following steps (as illustrated in Figure 3):

1. The local application makes programmatic calls to the stub.
2. The stub makes calls to the J2ME Web Services implementation via the formally defined SPI.
3. The J2ME Web Services implementation uses the information received from the stub via the SPI to open a network connection to the remote Web service (as illustrated in Figure 2), perform the requested operation, and return the result to the stub via the SPI.
4. The stub receives the result of the programmatic call and returns it to the local application (as illustrated in Figure 2).

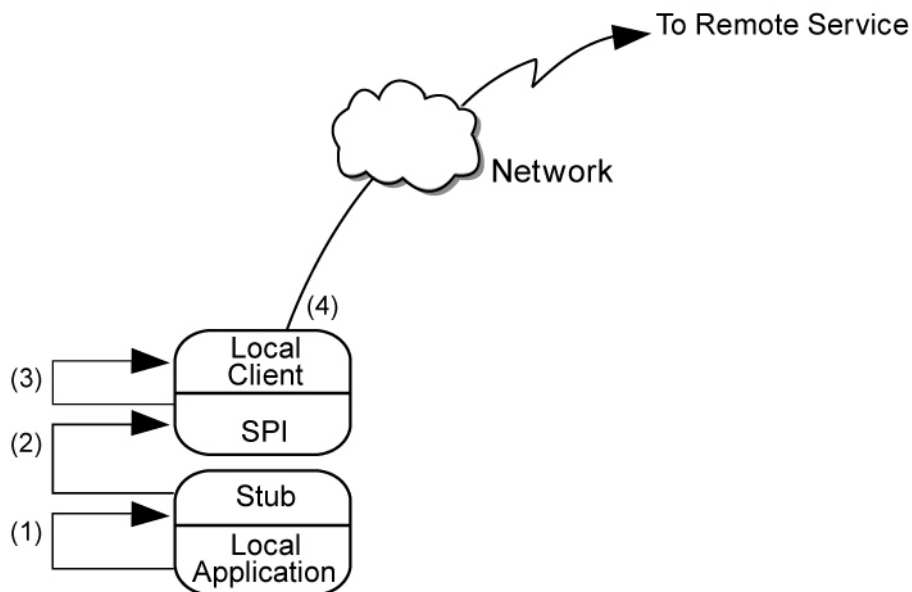


Figure 3. The Stub and the Service Provider Interface

Chapter 5

Web Services in the Wireless Marketplace

Today's Web services marketplace is exploding with new opportunities; each day developers bring ever more powerful devices and new applications into play. According to recent projections, worldwide revenue from mobile data services is expected to top \$47 billion in 2005, with exponential growth in years beyond. Cell phones and handheld PDAs are becoming ubiquitous as consumers around the world move everyday activities onto their mobile devices. Simple applications like calculators and calendars have given way to streaming video on some 3G devices, making it possible for busy travelers to watch real-time news clips or short business presentations from the backseat of a taxi in Thailand or the lobby of a London hotel.

This explosion in the development of mobile Web services is facilitated by the strong presence of the J2ME environment, which provides developers with a rich set of APIs, support for Internet protocols, and the inherent security of the Java platform. Functionality that was once considered impossible even on the desktop — such as three dimensional browsers and CAD-like applications — may one day, in the not too distant future, be running on even the most common of cell phones.

The multibillion dollar business opportunities presented by this explosion in mobile devices — and the compact, portable Web services that make them truly exciting — will continue to grow. Though many technical issues remain to be solved to create a seamless world of secure, end-to-end, mobile Web service access, the momentum for it is there.

In this rapidly evolving, worldwide context for the growth of Web services, the J2ME Web Services Specification is just one component. Yet, it is a powerful one. The Web services solution offered by the J2ME platform and protocols such as HTTP, SOAP, WSDL, and XML, conveyed on an agnostic layer of RPC, is the wave of the future.

Chapter 6

Glossary

CLDC

Connected Limited Device Configuration. In conjunction with MIDP, provides the Java runtime environment for wireless and handheld devices.

HTML

HyperText Markup Language. A structured set of tags and formats used to create and display documents on the Internet.

HTTP

HyperText Transfer Protocol. A primary Internet protocol that facilitates communication between a client and a server.

J2EE

Java 2 Platform, Enterprise Edition. An enhanced version of the Java platform used to develop component-based, multitier enterprise applications. Its features include Web services support and development tools (SDK).

J2ME

Java 2 Platform, Micro Edition. A scaled-down version of the Java platform specifically designed to run in the reduced memory space of a wireless, handheld, or other small device.

J2SE

Java 2 Platform, Standard Edition. The core Java technology platform. It provides a complete environment for developing applications on desktops and servers. It also serves as the foundation for the J2EE platform and Java Web services.

JAXP

Java API for XML Processing. Enables applications to parse and transform XML documents, independent of a particular XML processing implementation. In the J2ME Web Services release, JAXP supports processing of documents using SAX.

JAX-RPC

Java API for XML-based RPC. Enables Java software developers to build Web applications and services incorporating XML-based RPC functionality, according to the SOAP 1.1 specification.

JCP

Java Community Process. The process used by the worldwide community of Java developers for formulating Java technology-based standards and evaluating specifications.

JSR

Java Specification Request. A specification submitted to the Java Community Process for consideration and review.

MIDP

Mobile Information Device Profile. In conjunction with CLDC, provides the Java runtime environment for wireless and handheld devices.

PDA

Personal Digital Assistant. A handheld device that can download and update content over a wireless connection.

RPC

Remote Procedure Call. A protocol that allows a program on one computer to execute a program on another computer somewhere on the Internet, as though it was a local program.

RMI

Remote Method Invocation. A protocol that enables applications written in the Java programming language to interoperate in a transparent, flexible way, even when deployed on very different systems.

SAX

Simple API for XML. A standard interface for event-based XML parsing.

SDK

Software Development Kit. The software development kit provided by J2SE.

SOAP

Simple Object Access Protocol. A lightweight protocol for exchanging information in a decentralized, distributed environment. An XML-based protocol, it consists of three parts: An envelope that defines a framework for describing what is in a message and how to process it; a set of encoding rules for expressing instances of application defined data types; and a convention for representing remote procedure calls and responses.

UDDI

Universal Description, Discovery, and Integration. A cross-industry initiative that creates a platform-independent, open framework for describing services, discovering businesses, integrating business services across the Internet, as well as an operational registry.

WSDL

Web Services Description Language. An XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used.

XML

eXtensible Markup Language. A structured system for organizing and tagging documents, designed especially for Web documents. Enables designers and developers to customize tags, enabling the definition, transmission, validation, and interpretation of data among applications and among organizations.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com



Sun Worldwide Sales Offices: Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +82-2-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333, Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Saudi Arabia +9661 273 4567, Singapore +65-6438-1888, Slovak Republic +421-2-4342-9485, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44 (0)1252 420000, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800, or online at sun.com/store

SUN™ © 2004 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, Java Community Process, JCP, J2EE, J2ME, J2SE, and The Network Is The Computer are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. Information subject to change without notice. 07/04 R1.0