

JAIN™ and Java™ in Communications

A white paper describing the key work of the JAIN community, and exploring how it relates to the bigger picture of Java in the communications industry

March 2004

<http://java.sun.com/products/jain>

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun, Sun Microsystems, the Sun logo, Java, JAIN, J2EE, J2SE, J2ME, JavaBeans and EJB are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

Contents

1	INTRODUCTION.....	4
2	WHAT IS THE JAIN INITIATIVE?.....	5
2.1	OBJECTIVES AND SCOPE	5
2.2	BUSINESS DRIVERS AND INDUSTRY GOALS.....	5
3	KEY JAVA APIS FOR THE COMMUNICATIONS DOMAIN.....	7
3.1	JAIN SERVICE LOGIC EXECUTION ENVIRONMENT (JSLEE).....	7
3.2	JAIN SESSION INITIATION PROTOCOL (JSIP).....	7
3.3	JAVA CALL CONTROL (JCC).....	7
3.4	SERVER APIS FOR MOBILE SYSTEMS (SAMS).....	8
3.5	MOBILE DEVICE MANAGEMENT AND MONITORING (DM).....	8
3.6	SIP API FOR J2ME (SIP FOR J2ME).....	8
3.7	LOCATION API FOR J2ME (LOCATION FOR J2ME).....	8
3.8	WIRELESS MESSAGING API (WMA).....	8
3.9	OPERATIONS SUPPORT SYSTEMS THROUGH JAVA (OSS/J) APIS.....	9
4	JAVA IN COMMUNICATIONS NETWORK TOPOLOGY.....	10
5	CONCLUSION.....	13
6	REFERENCES.....	14

1 Introduction

For the purpose of this white paper, Java in communications refers to Java Application Programming Interfaces (APIs) necessary to provide next generation telecommunications services. This includes call control, messaging, presence and location based services that affect devices ranging from the mobile handset to telecommunications grade application servers.

Specifically, the JAIN initiative has defined a set of Java technology APIs that enable the rapid development of Java based next generation communications products and services. The Java APIs defined through the JAIN initiative bring service portability, network independence, and open development to telephony, data and wireless communications networks.

By providing a new level of abstraction and associated Java interfaces for service creation across telephony, data and wireless communications networks, Java technology enables the integration of Internet and Telecommunications protocols. Furthermore, by allowing Java applications to access resources inside the communications network, the opportunity now exists to deliver thousands of services rather than the dozens currently available. Thus, Java technology is changing the telecommunications market from many proprietary closed systems to an open communications network architecture where services can be rapidly created and deployed.

2 What is the JAIN Initiative?

The JAIN initiative represents a community of communications experts defining the necessary Java interfaces as an extension of the core Java platform to migrate proprietary communications networks to open standardized based networks. Development is being carried out under the terms of Sun's Java Specification Participation Agreement (JSPA) and the Java Community Process (JCP™).

2.1 Objectives and Scope

The objective of the JAIN initiative is to create an open value chain from network equipment manufacturers; computer manufacturers; and device manufacturers to 3rd-party service providers; facility-based service providers; and communications service providers to consumers.

By creating an open value chain, this reduces the many proprietary and often specialized container and application interfaces down to a useful, but manageable, set of standard interfaces. This, in turn, has the effect of reducing the cost and complexity of communications development, deployment and maintenance.

The scope of the JAIN initiative is to unify complex wireline, wireless and IP communications interfaces into a set of industry defined Java standards. The container interfaces provide standard and unified ways to cater for the lifecycle of a service, thus accelerating communications service deployment. The application interfaces provide different levels of abstraction and accessibility to cater to many types of developer community, thus proliferating the creation of new and novel communications services.

2.2 Business Drivers and Industry Goals

The JAIN initiative extends Java technology to bring service portability, network independence and open development to wireless, wireline and Internet networks. This will positively alter the current business structure of these networks by providing:

- **Service Portability:** *Write Once, Run Anywhere.* Technology development is currently constrained by proprietary interfaces. This increases development cost, time to market, and maintenance requirements. By using the JCP, proprietary interfaces are standardized into uniform Java interfaces delivering truly portable applications.
- **Network Independence:** *Any Network.* By delivering the facility to allow applications to port, with equal capability, on wireless, wireline and Internet networks, Java technology enables network convergence. Additionally, as demand for services over Internet based networks rises, network independent Java technology makes the migration of services from traditional telephony networks to Internet networks a straight forward task.
- **Open Development:** *By Anyone.* By defining easy-to-use communications network Java APIs, this technology can be used to leverage a massive existing developer community to develop new revenue generating services that may otherwise not be developed.

The goal of the JAIN initiative is to change the communications market from many proprietary closed systems to an open environment, unlocking value similar to the effect triggered by J2EE in the IT industry. By opening the network to Java, network operators can extend their service portfolio, and make next generation communications application development faster, simpler and less expensive.

The removal of proprietary barriers will enable an open market where Network Equipment Providers (NEPs), Independent Software Vendors (ISVs), protocol stack vendors and Service

Providers (SPs) can market a variety of Java technology components. Network operators will then be able to select their components and vendors on the basis of functionality and value. This open value chain will stimulate the re-use of existing components and the development of additional or missing functionality, thereby maximizing efficiency as well as innovation. It also opens the market for innovative new players.

The JAIN initiative currently consists of two areas of Java API specification development:

- The **Container Interfaces** specify service execution environment APIs geared at, but not exclusive to, the communications domain
- The **Application Interfaces** specify service APIs that enable service development across wireline, wireless and IP communications networks

There is a dramatic shift from traditional Java servers in the communications network to container based solutions. The JAIN initiative is focused on container based network servers that adapt application interfaces to the container environment for service execution. Many Java technologies developed within the JCP, can supplement the interfaces defined within the JAIN initiative to provide enhanced communications solutions, for example J2EE, Web Services, security, and network identity.

3 Key Java APIs for the Communications Domain

Java is everywhere in the communications domain. Java APIs are available for mobile handsets, for the access and core networks, for network and web servers, and for operations support and billing systems. In addition to the many JCP-standard Java APIs, there are several proprietary Java APIs that complete the Java in communications picture.

The following sections list the key Java APIs for communications. Some have been standardized by the JAIN initiative, some by the OSS/J initiative, however, all of them are JCP standards. While [J2EE](#), [J2SE](#) and [J2ME](#) are pivotal to Java in communications, they are not discussed as they are already widely deployed and well understood in the communications domain. Some of the following Java APIs are not finalized. For further information on these APIs, please visit the URL associated with them.

3.1 *JAIN Service Logic Execution Environment (JSLEE)*

Of all the Java in communications APIs, JSLEE forms the crucial core component, which is central to many Java in communications network topologies.

The JSLEE specifies the standard Java based service platform for executing event orientated applications. Building upon enterprise technologies similar to that found in J2EE, JSLEE focuses on delivering available, reliable and scalable services that are portable across JSLEE application servers. JSLEE integrates an event model to the component programming model. JSLEE defines standard management interfaces through JMX, a resource adapter infrastructure for network adaptation, generic profile interfaces, persistence management for state redundancy, and concurrency control. Several facilities are part of JSLEE such as timers, alarm features, usage tracking and traces.

3.2 *JAIN Session Initiation Protocol (JSIP)*

JSIP enables voice over IP gateways, client endpoints, PBXs and other communications systems to interface with each other using the SIP protocol. JSIP addresses call setup and tear-down, and presence and instant messaging over IP based networks. JSIP does not include the mechanism for media streaming between the caller and the called party.

Similar to the HTTP protocol, the SIP protocol is a client/server protocol, with requests issued by clients and responses managed by servers. The SIP protocol enables users to participate in multimedia sessions (or calls) and communicate in both unicast and multicast sessions. The JSIP API provides an industry standard interface into proprietary SIP protocol stacks. The JSIP API encompasses functionality for users/agents; proxy servers; registrar servers and redirect servers.

3.3 *Java Call Control (JCC)*

JCC provides applications with a consistent mechanism for interfacing with underlying divergent networks. JCC includes the facilities required for observing, initiating, answering, processing and manipulating calls, where a call is understood to include, but not necessarily limited to, a multimedia, multiparty session. The application need only interface once to a JCC API and subsequent adapters will allow calls and data to pass to underlying integrated independent (wireless, wireline and Internet) networks.

3.4 Server APIs for Mobile Systems (SAMS)

SAMS is an umbrella initiative to provide mobile network service API capabilities, such as messaging, location and presence, for network servers. SAMS APIs provide a high layer of abstraction, independent from underlying network protocols and technologies, which permit them to be simple to use and applicable to multiple network service implementations. While providing such abstracted APIs will prevent some service scenarios, the SAMS APIs are complete enough in order to cover the many of the major ones. The first SAMS API is given below:

- **[SAMS Messaging](#)**

SAMS Messaging provides a standard, portable and protocol agnostic messaging API for SMS and MMS, which enables composing, sending and receiving short messages and multimedia messages. A multimedia message is a message containing text, images and sound. Short messages primarily contain text only, but they can also be used as a bearer for binary data. Since SAMS Messaging is a server API, both message types can be sent and received from J2SE, J2EE and JSLEE platforms.

3.5 Mobile Device Management and Monitoring (DM)

DM supports management of J2ME based mobile devices; and will not depend on specific configurations or profiles. DM functions will include allowing server applications to provision the platform software and libraries of devices on the wireless network; modify default settings and configuration parameters and supporting security policies; provision new application level software onto the device; provide a view of the software and application stack configuration of the remote device; and monitor application usage on mobile devices. DM is currently being defined outside the SAMS umbrella, however, it is a candidate SAMS API.

3.6 SIP API for J2ME (SIP for J2ME)

SIP for J2ME defines a multi purpose SIP connector API for J2ME clients. It enables SIP applications to be executed in memory limited terminals, and is especially targeted at mobile phones. SIP for J2ME is based on the Generic Connection Framework (GCF) defined in the J2ME Connected Limited Device Configuration (CLDC) and uses the existing I/O classes of CLDC. SIP for J2ME can typically be used in conjunction with the Mobile Information Device Profile (MIDP), but it can be used with other profiles.

3.7 Location API for J2ME (Location for J2ME)

Location for J2ME enables developers to write mobile location based applications for resource limited devices, the minimum platform configuration being CLDC. Location for J2ME provides a compact and generic API that produces information about the device's present physical location to Java applications.

3.8 Wireless Messaging API (WMA)

WMA provides standard access to wireless communication resources. This will allow 3rd party developers to build intelligent connected Java applications. The WMA is designed to run on J2ME configurations, such as CLDC and MIDP. WMA includes facilities to send and receive SMS; exchanging data using Unstructured Supplementary Service Data (USSD); and receiving cell broadcast data via the Cell Broadcast Service (CBS).

3.9 Operations Support Systems through Java (OSS/J) APIs

OSS/J is an umbrella initiative to provide network operators with operations support system (OSS) API capabilities, such as trouble ticketing and service activation. OSS/J APIs are designed for the telecommunications domain and developed for the J2EE platform.

- **Quality of Service API**

Quality of Service API provides a standard that allows telecommunications management applications to be developed and integrated with Java enabled Quality of Service (QoS) systems. These management applications can retrieve, calculate and present QoS data, such as support data, operability data, serviceability (ability of a service) data, and security data.

- **Trouble Ticket API**

Trouble Ticket API provides a standard that allows telecommunications management applications to be developed and integrated with Java enabled trouble ticket (trouble report) systems. These management applications can create, query, update and delete trouble tickets.

- **Service Activation API**

Service Activation API provides a standard that allows telecommunications management applications to be developed and integrated with Java enabled service activation systems. Management applications, such as order entry products, order manager products or workflow management products, can, for example, create, amend and cancel orders. These orders initiate the desired changes to the service, including activation, deactivation or configuration changes of the service.

- **IP Billing API**

IP Billing API provides a standard that allows telecommunications management applications to be developed and integrated with Java enabled IP Billing systems. Management applications can be used to retrieve all types of billing records (CDRs, SDRs, IPDRs) and initiate billing for services.

4 Java in Communications Network Topology

Many Java technologies are required to provide a complete communications solution. Core to these solutions are container technologies, such as JSLEE and J2EE, however other Java technologies are needed to provide complete solutions.

The following network topology brings out the essence of a typical and recommended Java in communications solution. It draws on mainstream Java Mobile, Desktop and Enterprise technologies, namely J2ME, J2SE and J2EE, as well as emerging Java Communications and OSS technologies, such as JSLEE, JSIP, SAMS and OSS/J. Such a network topology provides a way to visualise how all these various technologies relate to each other within the communications domain. Network deployment and implementation options are not considered here.

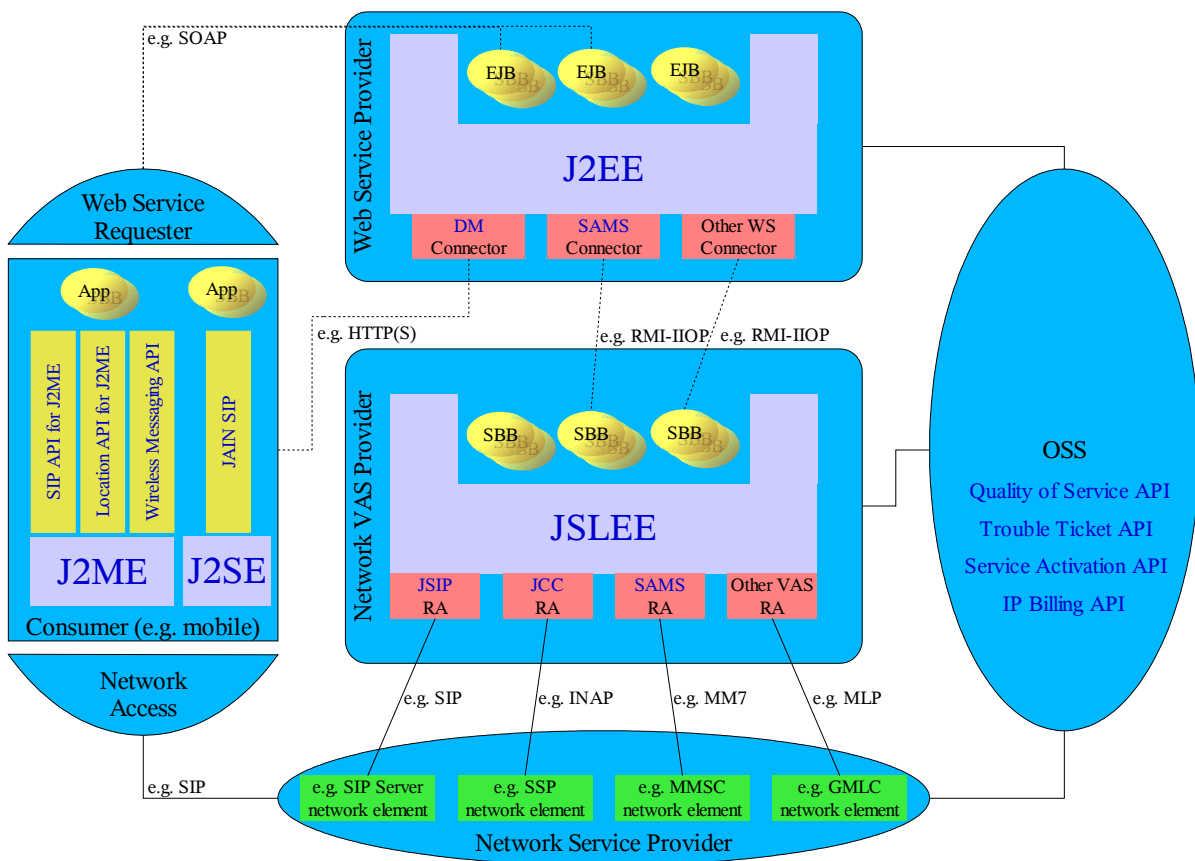


Figure 1. Java in Communications Network Topology

As shown in figure 1, central to the Java in communications network topology is the JSLEE. It is the JSLEE that forms the core of the Network Value-Add Service Provider (NVASP), permitting applications, or Service Building Blocks (SBBs) to access and manipulate network elements (network resources). Access to the network elements is only made possible through the Resource Adaptors (RAs). RAs provide a common architecture for the JSLEE to interface with the various different types of network elements (e.g. Signaling Switching Point (SSP), Multimedia Messaging Service Center (MMSC), and Gateway Mobile Location Center (GMLC)). Key RAs are those for JSIP, JCC and SAMS, where the RAs are derived from their respective service

APIs. Other RAs may be derived from interfaces from other organizations (e.g. 3GPP/3GPP2 OSA) or companies.

The RAs (or network element enablers) communicate with the network elements via network protocols, such as Intelligent Network Application Part (INAP), Multimedia Messaging reference number 7 protocol (MM7) and Mobile Location Protocol (MLP). Many different types of network elements are expected to be deployed within a network offered by the Network Service Provider (NSP), and it is these network elements, the network subelements, the access networks and the operations support systems that make up the fundamental network capabilities offered by the NSP.

The SBBs executing within the JSLEE are software components that can be composed to provide network services ranging in complexity from simple to elaborate. Orchestration of these services can be done entirely within the JSLEE or via an external interface. The benefits of providing orchestration within the JSLEE would include predictable security, reliability and performance for the service. The benefits of providing orchestration outside the JSLEE would include greater integration with IT systems and accessibility for the service.

The most suitable platform for providing greater service accessibility and integration would be J2EE, since it provides Web Services Interoperability (WS-I) Basic Profile 1.0 support and integrates easily with traditional IT systems, thus, a mixture of communications and IT services would be offered by the Web Service Provider (WSP). Within the WSP, the J2EE would permit applications, or Enterprise JavaBeans (EJBs), to orchestrate SBBs possibly into an Enterprise Service (service hosted within the Enterprise) or a Web Service (service exposed through the Internet), both of which would incorporate network capabilities. Access to the SBBs is done via the J2EE connectors, as shown in more detail in figure 2.

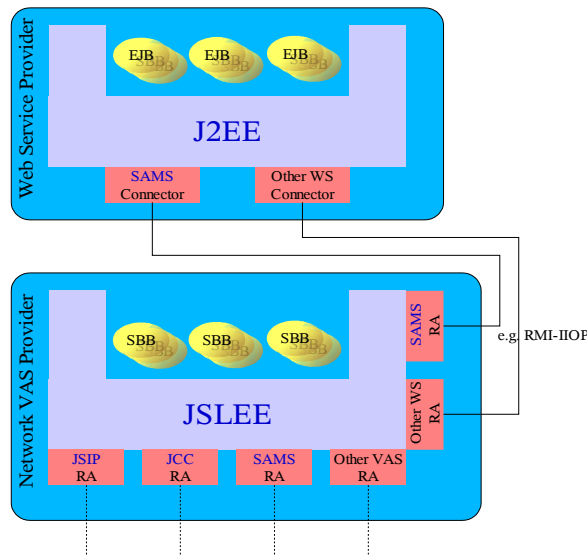


Figure 2. Integrating JSLEE with J2EE

If the WSP and NVASP were not co-located, it is possible that communication between the two entities would be done over the Remote Method Invocation over the Internet Inter-Orb Protocol

(RMI-IIOP), however, if they were co-located, it is possible that a high-speed proprietary protocol might be used.

The Web Services would be offered by the WSP through the Web Services “Publish” mechanism, and accessed by the Web Service Requester (WSR) embedded within the Consumer's device. The choice of Consumer device could vary wildly, however, the two most popular device categories would be mobile devices and desktop devices, which would typically host the J2ME and J2SE platforms, respectively. Applications executing in the Consumer device would access the Web Services through the Web Services “Find” and “Bind” mechanisms, using SOAP, for example. Basic network connectivity and “line” based services, such as sending an SMS, offered by the NSP would be accessed by the Consumer device's Network Access capability via the Consumer device APIs, such as SIP for J2ME, and WMA, using an access protocol, such as SIP. Internet access to Web Portals for functionality such as DM would be done via an Internet protocol, such as HyperText Transfer Protocol (HTTP) or HTTP with security (HTTP(S)).

The operation of the various network elements, network value-add elements and web elements contained within the NSP, NVASP and WSP, respectively, could all be supported by the OSS management applications, which also would run within a J2EE platform.

This network topology describes only part of the Java in communications architecture. Other Java APIs exist for network identity, data presentation, data formatting, service creation, content creation, content delivery, and data storage and access, etc. However, what has been explored is how Java fulfills the roles of service delivery and operations support in next generation networks.

5 Conclusion

Java technology has achieved great progress in providing open markets for the Desktop, Enterprise, Mobile and Card domains. The JAIN and OSS/J initiatives are producing early results that would indicate another success for Java technology, this time in the Communications and OSS domains.

Java APIs for Communications enable an open market for lower cost, rapid development next generation communications products and services. In the fiercely competitive communications market, network operators that embrace these next generation capabilities will succeed by leveraging their ability to create new services to differentiate themselves from the competition, and by lowering their operational costs through cheaper and more integrated platforms.

6 References

- [1] DM – <http://jcp.org/en/jsr/detail?id=233>
- [2] IP Billing API – <http://jcp.org/en/jsr/detail?id=130>
- [3] Java Community Process – <http://jcp.org>
- [4] J2EE – <http://java.sun.com/j2ee/index.jsp>
- [5] J2ME – <http://java.sun.com/j2me/index.jsp>
- [6] J2SE – <http://java.sun.com/j2se/index.jsp>
- [7] JAIN initiative – <http://java.sun.com/products/jain>
- [8] JCC – <http://jcp.org/en/jsr/detail?id=21>
- [9] Location for J2ME – <http://jcp.org/en/jsr/detail?id=179>
- [10] JSIP – <http://jcp.org/en/jsr/detail?id=32>
- [11] JSLEE – <http://jcp.org/en/jsr/detail?id=22>
- [12] OSS/J initiative – <http://java.sun.com/products/oss>
- [13] SAMS Messaging – <http://jcp.org/en/jsr/detail?id=212>
- [14] Service Activation API – <http://jcp.org/en/jsr/detail?id=89>
- [15] SIP for J2ME – <http://jcp.org/en/jsr/detail?id=180>
- [16] Trouble Ticket API – <http://jcp.org/en/jsr/detail?id=91>
- [17] Quality of Service API – <http://jcp.org/en/jsr/detail?id=90>
- [18] WMA – <http://jcp.org/en/jsr/detail?id=120>
