

The JAIN™ APIs: Integrated Network APIs for the Java™ Platform

*A white paper describing the JAIN objectives, overall technical
architecture and program structure*

September 2001

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

Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 U.S.A. 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. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Solaris, Solaris ISP Server, Solaris JumpStart, Java, HotJava, JDK, JAIN, JINI, JavaBeans, Sun Internet Administrator, Sun Internet Calendar Server, Sun Internet FTP Server, Sun Internet Mail Server, Sun Internet News Server, Sun Internet Services Monitor, Sun WebServer, and SunScreen are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. 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. Netscape Navigator is a trademark of Netscape Communications Corporation.

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 ARE HELD TO BE LEGALLY INVALID.

Copyright 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Solaris, Solaris ISP Server, Solaris JumpStart, Java, HotJava, JDK, JAIN, JINI, JavaBeans, Sun Internet Administrator, Sun Internet Calendar Server, Sun Internet FTP Server, Sun Internet Mail Server, Sun Internet News Server, Sun Internet Services Monitor, Sun WebServer, et SunScreen sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Netscape Navigator est une marque de Netscape Communications Corporation

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Contents

Executive Summary.....	4
1 What is the JAIN Initiative?.....	5
Objectives and Scope.....	5
Business Drivers and Industry Goals	6
2 Technology Overview.....	7
Architecture.....	8
Network Topology.....	10
Major Components.....	12
Related Java Technology Initiatives.....	16
3 The JAIN Program.....	18
JAIN Specification Process.....	18
Organization.....	19
JAIN Program Management Responsibilities.....	20
Levels of Participation.....	21
Appendix A - JAIN Community Members	23
Appendix B - References	26

Executive Summary

The JAIN™ APIs for Integrated Networks bring service portability, convergence, and secure network access to telephony and data networks.

By providing a new level of abstraction and associated Java™ interfaces for service creation across Public Switched Telephone Network (PSTN), packet (e.g. Internet Protocol (IP) or Asynchronous Transfer Mode (ATM)) and wireless networks, JAIN technology enables the integration of Internet and Intelligent Network (IN) protocols. This is referred to as Integrated Networks. Furthermore, by allowing Java applications to have secure access to resources inside the network, the opportunity is created to deliver thousands of services rather than the dozens currently available.

Thus, JAIN technology is changing the telecommunications market from many proprietary closed systems to a single network architecture where services can be rapidly created and deployed.

JAIN technology is being specified as a community extension to the Java Platform. Development is being carried out under the terms of Sun's Java Specification Participation Agreement (JSPA), Java Community ProcessSM (JCP), and Sun's Community Source Code Licensing (SCSL) terms.

The JAIN initiative consists of two areas of development:

- " The **Protocol API Specifications** specify interfaces to wireline, wireless and IP signaling protocols
- " The **Application API Specifications** address the APIs required for service creation within a Java framework spanning across all protocols covered by the Protocol API Specifications

Today, over eighty companies are participating in the JAIN initiative at various levels (see Appendix: JAIN Community Members).

1

What is the JAIN Initiative?

Objectives and Scope

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

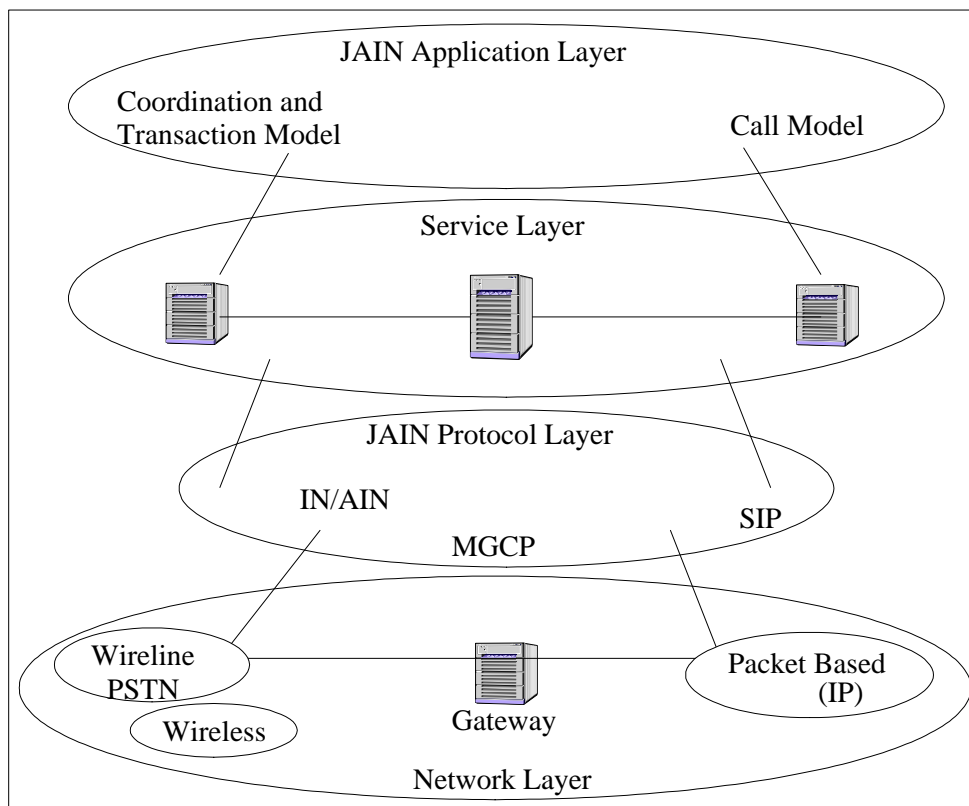


Figure 1: JAIN Initiative

The JAIN initiative integrates wireline, wireless, and packet based networks, as illustrated in the diagram above. The adaptation of network specific protocols to the JAIN model is covered in the Protocol API Specifications. Additionally, the JAIN initiative abstracts the protocols covered by the Protocol API Specifications into a single call control, coordination, and transaction model to be used by compliant services. This is being driven by the work in the Application API Specifications.

Business Drivers and Industry Goals

The JAIN initiative brings service portability, convergence, and secure network access to telephony and Internet networks. This will positively alter the current business structure of these networks as follows:

- **Service Portability:** - *Write Once, Run Anywhere.* Technology development is currently constrained by proprietary interfaces. This increases development cost, time to market, and maintenance requirements. With the JAIN initiative, proprietary interfaces are reshaped to uniform Java interfaces delivering truly portable applications.
- **Network Convergence:** (Integrated Networks) - *Any Network.* By delivering the facility to allow applications and services to run on PSTN, packet (e.g. IP or ATM) and wireless networks, JAIN technology speeds network convergence. As demand for services over IP rises, new economies of scale are possible as well as more efficient management and greater integration with IT.
- **Secure Network Access** - *By Anyone!* By enabling applications residing outside the network to directly access network resources and devices to carry out specific actions or functions, a new environment is created for developers and users. The market opportunity for new services is huge when controlled access is provided to the available functionality and intelligence inside the telecommunications networks.

The JAIN initiative takes the telecommunications/Internet market from many proprietary closed systems to a single open environment able to host a large variety of services. By opening the network to Java applications, the opportunity is created to deliver thousands of portable, integrated services rather than the dozens currently available. Java and JAIN technologies will allow carriers to extend the services and make them more feature-rich. JAIN technology makes next generation telecom application development faster, simpler and less expensive through the use of Java technology.

The removal of proprietary roadblocks will set the stage for an open market where Network Equipment Providers (NEPs), Independent Software Vendors (ISVs), protocol stack vendors, service providers and carriers can market a variety of Java technology-based components. Participants will then be able to select their components and vendors on the basis of functionality and value. This 'open value chain' market model will stimulate the re-use of existing components and the development of additional or missing functionality - maximizing efficiency as well as innovation. It also opens the market for innovative new players.

The next generation architecture provided by JAIN technology creates a level playing field for deploying new services. This model is best served when all network levels participate - hardware companies, stack providers, network equipment providers, service providers, and carriers. In the fiercely competitive telecom market, the carriers that embrace these next generation capabilities will succeed by leveraging their ability to create new services to differentiate themselves from less nimble competitors.

2

Technology Overview

As previously noted, the JAIN specification effort is divided between two areas of development:

- The Protocol API Specifications specify interfaces to wireline, wireless and IP signaling protocols
- The Application API Specifications address the APIs required for service creation within a Java framework spanning across all protocols covered by the Protocol API Specifications

The areas of focus of the Protocol & Application API Specifications is detailed in the section ‘Major Components’ later in this chapter.

These two groups of API Specifications bring together Intelligent Network and Internet technologies to provide state-of-the-art telecom services. Services examples include: 800 or Free-Phone, Follow-Me, Time-Of-Day, Calling Name Delivery, Do-Not-Disturb, and more for the phone, the Internet, cell phone, web device, etc. On Integrated Networks these services will include not only voice but also a variety of data, media, programs, and unified messaging elements. Such an integration of the Internet and the Intelligent Networks also provides access to personalized data and network services for end clients, e.g. through a simple browser customers can access services and even define their own network environment.

A JAIN service provisioned network includes support for service creation, service logic execution, and service policies.

Service Creation allows the development of new service building blocks and the assembly of services from these building blocks, typically using one or more commercially available, off-the-shelf tools such as an Integrated Development Environment (IDE) or Bean Boxes (providing a library of re-usable JavaBeans™ components). Next generation network services will be assembled on the fly in a plug-and-play fashion, drastically reducing the time and effort to develop services.

Services are then deployed into a JAIN™ Service Logic Execution Environment (SLEE) allowing the provisioning and lifecycle management of these services. Once a service is deployed in the JAIN SLEE, rules will be defined for service impacting network and connection parameters (e.g. Quality of Service), billing and usage parameters, network integrity management, etc. This is called Policy Management, and will be addressed in subsequent JAIN specifications.

Architecture

At its core, the JAIN architecture defines a software component library, a set of development tools, a service creation environment, and a carrier-grade service logic execution environment to build next generation services for integrated PSTN, packet (e.g. ATM and IP), and wireless networks.

While many new systems may be developed using the JAIN architecture, several basic communications abstractions are carried forward:

A Network layer:

- Telecommunications - (Advanced) Intelligent Networks (AIN/IN) or Signaling System 7 (SS7) with many protocols - ISUP, INAP, TCAP, etc.
- Wireless - SS7 with Mobile Application Part (MAP) layer
- Internet or Packet - SIP, MGCP, Megaco, H.323.

A Signaling Layer:

- Telecommunication - Signaling Service Point (SSPs) or switches
- Wireless - Mobile Switching Centers (MSCs)
- Internet - emerging softswitches or call agents, and media gateway controllers or H.323 gatekeepers.

A Service Layer:

- Telecommunication - Service Control Points (SCPs)
- Wireless - any combination of Base Station Controllers (BSCs), Home Location Registers (HLRs), Visitor Location Registers (VLR), and MSCs.
- Internet - Application Server

As illustrated in the figure below, the JAIN architecture includes a Service Creation Environment (SCE) for both trusted next generation network services and untrusted third-party applications. Trusted services (and policies) reside within the core of public networks. Untrusted services are services written by third parties that access functions within the core public networks. Through a secure service provider interface, these third party applications are kept from compromising the reliability or integrity of those networks.

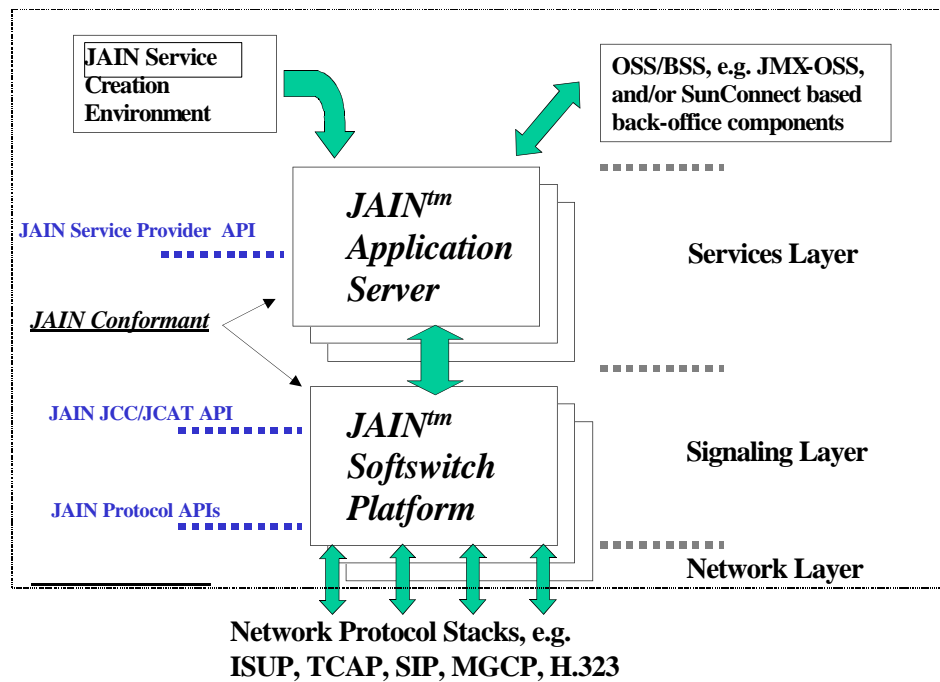


Figure 2: JAIN Architecture

The JAIN architecture provides for the (trusted) next generation network services in a carrier-grade service logic execution environment. It is expected that many services and implementations of the JAIN SLEE will be implemented using Enterprise JavaBeans™ (EJB).

Untrusted services like trusted services might have similar requirements for the SLEE depending on their scope. Untrusted services also will rely on container infrastructures that can be used to 'hold' services (e.g. EJB, Java™ Embedded Server (JES), and JINI™). The JAIN SCE is compatible with both environments (e.g. trusted and untrusted service creation).

Standard Interfaces for Signaling and Services

The JAIN initiative defines a standard set of interfaces for signaling and service protocols. With a well-defined enforceable Java standard, services readily port, unchanged, to any vendor platform. By defining a common interface between two or more protocols, networks converge. The JAIN™ Call Control API (JCC) and the JAIN™ Coordination and Transactions (JCAT) within the JAIN community are defining such a call control/session interface.

The objective of the JCC and JCAT APIs are to provide applications with a consistent mechanism for call control and call processing transactions. JCC and JCAT include the facilities required for observing, initiating, answering, processing and manipulating calls, where a call may include multimedia, multiparty sessions over the underlying Integrated Network (PSTN, packet and/or wireless). JCC is a unified call model that incorporates the basic features of the Java™ Telephony API (JTAPI) as well as Parlay's Call Control Service, and provides an extensible Java framework for supporting complex call processing applications.

Coordination and transaction includes facilities for additional applications to be invoked before, during or after call processing, generally for value added features, such as Virtual Private Networks, media-rich (multiparty) sessions, etc.

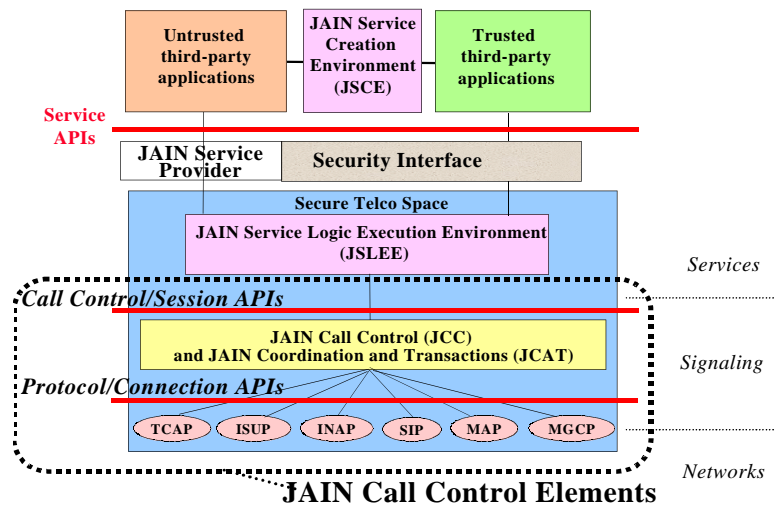
Network Topology

The JAIN network topology provides carriers with the ability to deploy next generation network services on devices inside or at the edge of the Integrated Network, including any Java technology enabled end user device. Furthermore, support for all the necessary telephony protocols that are used between the different network elements in IN, AIN, and IP based (telephony) networks is mandatory.

A key aspect of the the JAIN component architecture is to move the signaling layer away from proprietary switches into open call control servers, also known as call agents, media gateway controllers, or softswitches . Signaling, that is, the protocols used to establish and terminate communication connections, is the common thread between conventional telecommunications switches and softswitches. The ability to adapt the signaling components between networks is paramount to the success of carriers and network service providers.

Figure 3 is a pictorial representation of where JAIN APIs are defined within a communications platform. The softswitch architecture is centered on mapping the call control/session interfaces onto the underlying protocol APIs. Since softswitches perform signaling on IP networks, most are equipped with a SIP, MGCP, MEGACO or H.323 underlying protocols. Several softswitches also include SS7 protocols to address interfaces for the existing telephone network.

Figure 3: JAIN APIs



The JAIN initiative offers key components to build open softswitch architectures and provide service portability through a unified Java interface to develop and deploy next generation services. The JAIN framework of APIs is broken into four categories:

- Connection or Protocol Interfaces
- Call Control or Session Interfaces
- Service Logic Interfaces
- Service provider access

As an example of a complete service, let's have a closer look at a conference call. The fact that a conference call is needed for about 1 hour with 12 participants is defined on the services layer, which will also create the overall accounting record used for billing. The actual call legs (12) attached to the conference call are handled at the signaling layer by either participants calling in with a participant number or the call control element setting up some of the call legs. Call legs could be using voice over IP (VoIP), wireless or wireline protocols. The network layer protocol stacks handle individual call legs, e.g. a leg is disconnected on hang-up.

It should be emphasized that the JAIN compliant components do not necessarily reside on a single server, since the Gate Keeper, Gateway Control, and Signaling Gateway functionality is typically implemented as a multi-tier, distributed application on all signaling network elements. Such an approach provides significant advantages for scalability, performance, reliability, manageability, reusability, and flexibility.

Major Components

JAIN Protocol API Specifications

The JAIN SS7 APIs define Java classes to interface Transaction Capability Application Part (TCAP), Integrated Services Digital Network (ISDN) User Part (ISUP), Intelligent Network Application Part (INAP) and Mobile Application Part (MAP). The JAIN IP APIs include SIP, MGCP, MEGACO and H.323. This list will be expanded to meet industry requirements.

JAIN™ TCAP

TCAP is a layer of the SS7 communications protocol that was developed to add transaction based functionality to the existing telephone network. This includes functions such as maintaining a dialogue with a database or providing the mechanism to access remote switches and activating features within those switches. The TCAP layer is designed for signaling related messages and provides a means for transfer of information from one application at a switch to another application within another network entity. The information passed through the TCAP layer must be transferred between applications, transparently through the network.

The JAIN TCAP API specifies a Java API that will provide the interfaces and classes required to initialize and terminate a TCAP session, manage TCAP dialogue identifiers, retrieve, build and send dialogue and component primitives.

TCAP provides the means for the invocation of remote operations between signaling point nodes and thus provides generic services to applications such as mobile telephony and free phone service.

TCAP is used in the real world today by a variety of applications including:

- "800" Number Translation – One of the first uses of TCAP, where a virtual "800" phone number is converted into a physical route-able phone number.
- Calling Name Delivery – A subscriber can see the name of a caller instead of just a caller id.
- Do Not Disturb – A subscriber can temporarily block incoming call and revert the calls to a voice mail box based on criteria such as time of day, caller id, etc.

JAIN™ ISUP

The JAIN ISUP provides an API to the signaling functions that are needed to support switched voice and data applications. ISUP is defined within the SS7 specifications as a communications protocol used to set-up, manage and release trunk circuits that carry voice and data call over the PSTN.

JAIN™ MAP

JAIN MAP classes manage the Pan European standard for cellular processing Global System for Mobile Communications (GSM) and the North American standard for cellular processing (IS41). The JAIN MAP interface is concerned with:

- Network Topology: Home Location Register, Visitor Location Register, Base Station Controllers, Mobile Switching Centers, etc.
- Mobile applications such as roaming, hand-off, lookup, etc.

- Wireline interconnect
- Services such as subscriber voting, reporting, short message service, news, etc.

JAIN™ 3G MAP

The JAIN 3G MAP API specification is for the mobile application in the 3G domain. The entities in a Public Land Mobile Network (PLMN) need to exchange information to manage roaming mobile stations (MSs). The 3G MAP defines protocol through which the core network entities in the 3G domain can transfer information between them. It defines the message syntax and the procedures for message exchange between network entities e.g, MSC,HLR,VLR,GMSC,gsmSCF,IWMSC,SIWF,SGSN,GGSN,GMLC. The representative set of services between these entities are:

- Location Management services(HLR/VLR, LCS, gsmSCF)
- Authentication Management services(HLR/VLR)
- Subscriber Management services(HLR/VLR)
- Subscriber Information services(HLR/gsmSCF)
- Supplementary services (MSC/VLR,HLR/VLR)
- Messaging services (MSC/HLR,SGSN/HLR)

JAIN™ MGCP

Media Gateway Controller Protocol (MGCP) controls voice and media over packet gateways. Gateways allow for the interconnection of the PSTN with packet networks. This allows users to make calls that span both the PSTN and packet networks. MGCP has been defined as a protocol for controlling these gateways.

The JAIN MGCP API allows developers to write MGCP services while the standard is maturing. The JAIN MGCP API will be backward compatible as the MGCP protocol is enhanced. JAIN MGCP API provides an industry standard Java technology interface into proprietary MGCP protocol stacks.

The JAIN MGCP API includes gateway interfaces and control interfaces necessary to create and release connections, to modify connections, and to audit connections.

JAIN™ SIP

Session Initiation Protocol (SIP) enables voice over IP gateways, client end points, PBXs and other communications systems to interface with each other. SIP addresses the call setup and tear-down mechanisms over the Internet. SIP does not include the mechanism for media streaming between caller and "callee".

Similar to HTTP protocols, SIP is an application client/server protocol, with requests issued by clients and responses managed by servers. SIP enables users to participate in multimedia sessions (or calls) and communicate in both unicast and multicast sessions.

JAIN SIP API provides a industry standard interface into proprietary SIP protocol stacks. The JAIN SIP API includes:

- User/Agent Server interfaces

- Proxy Server interfaces
- ReDirect Server interfaces

JAIN™ SIP Lite

JAIN SIP Lite is a high-level Java API that will be to allow application developers to create application's that have SIP as their underlying protocol without having the need for extensive knowledge of the SIP protocol. This will enable developers to rapidly create applications, such as user-agent type applications. JAIN SIP Lite is a thin API that can be used as a high-level wrapper around the SIP protocol providing application developers with a more simple abstracted SIP API.

SIP Servlets

The Session Initiation Protocol (SIP) is used to establish and manage multimedia IP sessions. The SIP Servlet API defines a high-level extension API for SIP servers. It enables SIP applications to be deployed and managed based on the servlet model and the existing Servlet API.

The SIP Servlets API enables SIP servers to be augmented with Java extension code which can be deployed and managed as a unit. In addition to the actual SIP Servlet API, XML based deployment descriptors and related file formats are defined. The SIP Servlet API is similar in nature to the HTTP Servlet API, however there are some important differences – the main difference being that HTTP servlets run only on origin servers. In SIP networks, proxy servers play a much more significant role, as do application servers which initiate requests. As such, SIP servlets must be able to run on these other server types.

JAIN™ INAP

JAIN INAP is a non-call-related control protocol that allows applications to communicate between various nodes/functional entities of an Intelligent Network . The protocol defines the operations required to be performed between service providers for providing IN services, such as number translation, time of day, and follow me. The JAIN INAP API will be based on the American National Standards Institute (ANSI)/Telcordia Advanced Intelligent Network (AIN 0.2) and International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) CS-2 INAP specifications.

JAIN™ MEGACO

MEGACO standardizes the interface (Reference N) between the Call Control entity (Media Gateway Controller or MGC) and the Media processing entity (Media Gateway or MG) in the decomposed H.323 Gateway architecture proposed by ETSI TIPHON and adopted by IETF. MEGACO has been defined by IETF MEGACO WG and ITU-T SG-16. MEGACO is central to VoIP solutions and may be integrated into products such as Central Office Switches, Gateways (Trunking, Residential, Access), Network Access Servers, Cable Modems, PBXs etc., to develop a convergent voice and data solution.

JAIN™ H.323

H.323 is the ITU-T standard for “Packet-based Multimedia Communications Systems” and the protocols necessary to achieve the defined system. The H.323 System is designed to move real-time bi-directional multimedia (audio, video, data, fax) across packet-based networks while maintaining connectivity to PSTN, including SS7 circuit-switched networks and H.320 systems. The signaling protocols are defined by H.225.0 and include RAS and Q.931. RAS, or Registration, Administration and Status, handles user management; Q.931, an adaptation and optimization for packet networks of the original ISDN Q.931, handles call signaling; H.323 also uses the H.245 protocol which provides media control by opening the media channels and negotiating the capabilities for each endpoint. H.245 can also change the characteristics of the channel, allowing for asymmetrical rate and codec adaptation between endpoints, features necessary for multimedia interactive communications.

JAIN™ H.323 API provides an industry standard interface for implementation of H.323 building blocks: Gatekeepers, Terminals, Gateways and Multipoint Control Units (MCUs) compliant with H.323v.4 and is backwards compatible with earlier H.323 versions.

JAIN™ Operations, Administration, and Maintenance (OAM)

OAM provide a standard portable interface to a service provider to provision and maintain components in a PSTN/IP network. Transmission rates, hardware characteristics, routing configurations, etc. are all part of provisioning a protocol. While other JAIN APIs, such the JAIN TCAP API and the JAIN ISUP API, define a common interface to proprietary implementations of protocol layers in an SS7 protocol stack, the JAIN OAM API defines a common interface to proprietary management interfaces for SS7 protocol stack and also to IP based protocol stacks such as MEGACO, MGCP, H.323 and SIP. The JAIN OAM API allows for the creation, deletion, modification and monitoring of network components.

JAIN™ SDP

The JAIN SDP API describes multimedia IP sessions. The SDP API is intended to be complimentary to other APIs such as: JAIN H323, JAIN Megaco, JAIN MGCP, JAIN SIP, JMF and the Servlet APIs.

The SDP messages encode a description of a session and are composed of two types of elements: fields and descriptions. SDP messages are formatted as a set of lines. A field is a line of text in the SDP message. Each field contain information specific some aspect of the session. There are several field types. Each field type is identified by a unique character (e.g., 'c' for connection). Each field type has a fixed format for its content. A description is an aggregation of fields.

The SDP API version 1.0 will enable the proper encoding and decoding of SDP messages and allow a user to get, set or modify any element of a field or description in a SDP message.

JAIN Application API Specifications

JAIN™ Call Control

The JAIN Call Control (JCC) API provides applications with a consistent mechanism for interfacing with underlying divergent networks. The application needs only interface once to a JCC interface and the subsequent JAIN adapters will allow calls and data to pass to various networks.

JAIN™ Coordination and Transaction

The JAIN Coordination and Transaction (JCAT) API includes the facilities required for applications to be invoked and return results before, during or after calls; to process call parameters or subscriber-supplied information; and to engage in further call processing and control. JCAT perceives JAIN Call Control as its core call control package and extends it with concepts to model and control terminal capabilities.

JCAT extends the JCC call control model with terminal capabilities and it enriches JCC's state diagrams such that an even more diverse range of applications can be supported such as the AIN/IN class of applications.

JAIN™ Service Logic Execution Environment

Once services are created, they can be tested and deployed in the JAIN Service Logic Execution Environment (SLEE). JAIN SLEE defines interfaces and requirements mandatory for telco/Internet operations within carrier grade and Internet networks.

JAIN™ Service Provider API for the Parlay Specification

The JAIN Service Provider APIs (SPA) for the Parlay specification will provide the secure access mechanism to network capabilities. This set of APIs will focus on a Java technology based implementation of Parlay and with extensibility to allow other services to be exported by the network operator and discovered by the service provider/user.

JAIN™ Common API

There are currently over twenty JAIN specifications which have a consistent architecture and share common design patterns. As a result, they have common base interfaces and classes, such as data types and exception definitions.

The JAIN Common API documents and specifies these common base interfaces and classes to avoid duplication in each of the JAIN specifications and to maintain consistency of these interfaces and classes across these JSRs.

Related Java Technology Initiatives

The Enterprise Java Platform has been designed to provide a flexible, reliable, and scalable environment and is an excellent candidate technology for the development and deployment of the JAIN architecture and JAIN next generation services. The JAIN architecture leverages the platform independence of Java technology. However it is also designed to be distributed and independent from any specific protocol, or middleware infrastructure (e.g., Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), and Distributed Common Object Model (DCOM)).

The JAIN architecture also leverages and integrates many existing and on-going development efforts in the Java programming library space. Some of the most important related initiatives are developing (or have developed) a Java technology-based encapsulation to data base access (JDBC), access to naming and directory services (JNDI) and a Java™ Transaction Service (JTS) based on CORBA's Object Transaction Service.

Unifying a Family of Call Models

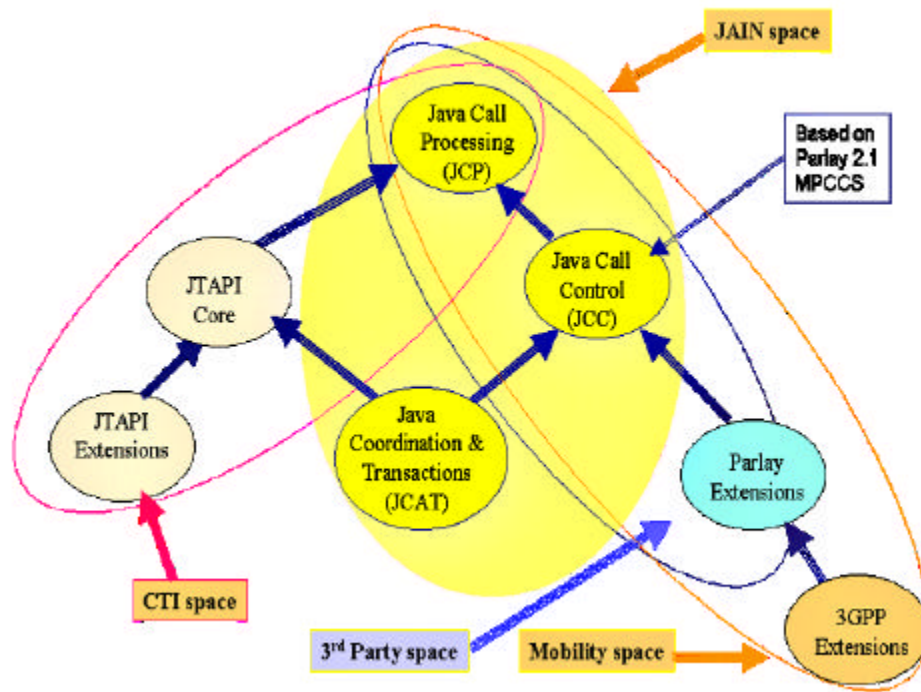


Figure 4: Call Control Hierarchy

A Java extension of particular importance is JTAPI, the Java Telephony API. JTAPI is expressed in Java code and defines a core call model to support basic call setup, and a number of extensions, mostly designed to model call center features, multiparty conference calls, call routing, and so on. The JCC API and the future JCAT API efforts are attempting to build on the best aspects of JTAPI and other existing call models in order to provide an interface supporting a family of call models.

The call control service of Parlay has been used as a basis for the JCC work within the JAIN architecture. Together, the Java technology and the JAIN architecture provide an ideal framework and toolkit on top of which Parlay services and a Parlay gateway network element can be implemented.

Parlay (<http://www.parlay.org/>) is a technology-independent, secure API for telecom services. The Parlay API specification enables a new generation of dynamic telecom applications created and maintained by the networks' customers, using their own private data. The API helps the network operator to maximize the value of their technology by directly passing on that functionality to third parties in a safe and controlled manner, while hiding much of the complexity of network signaling from the developer.

The JAIN initiative and Parlay are complimentary and together will provide significant opportunity to expand the access and breadth of services available. The Java programming language provides an ideal mechanism to make Parlay services available on the Internet while Parlay is a great way to bring security to the JAIN community, expanding the reach of the JAIN activity.

3

The JAIN Program

JAIN technology is being designed, specified, and developed as a community extension to the Java Platform. JAIN technology will be built under the terms of Sun's JSPA, Java Community Process, and Sun's Community Source Code Licensing terms.

JAIN Specification Process

Sun has implemented a formal process for developing Java specifications that produces high-quality specifications in "Internet-time" using an inclusive, consensus building process that not only delivers the specification, but also the reference implementation and its associated kit of compatibility tests.

Sun's experience has proven that the best way to develop a specification is to start with a handful of industry experts who have a deep understanding of the technology in question and then have a strong technical lead work with them to create a first draft. Consensus is then built using an iterative review process that allows an ever-widening audience to participate and to see their comments and suggestions incorporated into successive draft versions of the specification prior to its final release.

This formal process was designed to be fast, flexible, and adaptable to a wide variety of working styles present in the community today. Each use of the process will be audited by the independent auditing firm of PriceWaterhouseCoopers.

Steps in specification development strictly follow the guidelines set forth in the Java Community Process (JCP) version 2.0.

In building upon the Java Community Process, the JAIN initiative leverages the requirements for participants and the procedures for driving community approval on specifications.

Further information on the Java Community Process can be found at: <http://jcp.org> .

Organization

The JAIN program is fully compliant with the Java Community Process (JCP) 2.0 and is organized by a number of Expert Groups for particular API Specifications.

Each Expert Group is headed by a Specification Lead who assumes the responsibility to deliver a consensus based API Specification, a Technology Compatibility Kit (TCK) and a Reference Implementation (RI). The API Specifications developed are appropriate to one of the following areas:

- **Protocol API Specifications** - standardizing interfaces to wireline, wireless and IP signaling protocols
- **Application API Specifications** - dealing broadly with the APIs required for service creation within a Java framework that span across all protocols covered by the Protocol API Specifications

JAIN Protocol API Specifications

The JAIN Protocol API Specifications are focused on creating the specifications for the generic APIs to deliver SS7 plus IP protocol (convergence) functionality through the Java programming language. Individual Expert Groups have the responsibility for specific specifications within the JAIN Protocol API Specifications domain. Currently, the following JAIN Protocol API Specifications are either completed or underway:

- **TCAP - Specification Lead: Sun Microsystems**
- **OAM - Specification Lead: Sun Microsystems**
- **ISUP - Specification Lead: Ulticom**
- **MAP - Specification Lead: Ericsson**
- **3G MAP – Specification Lead: Hughes Software Systems**
- **SIP - Specification Lead: dynamicsoft**
- **SIP Lite – Specification Lead: Ubiquity Software Corporation**
- **SIP Servlets – Specification Lead: dynamicsoft**
- **MGCP - Specification Lead: Telcordia**
- **INAP - Specification Lead: Mahindra BT**
- **MEGACO - Specification Lead: Hughes Software Systems**
- **H.323 – Specification Lead: RADVision**
- **SDP – Specification Lead: dynamicsoft**

JAIN Application API Specifications

The Application API Specifications are focused on developing facilities built on top of the protocol layer to enable easy application development utilizing the breadth of Java objects and services, while shielding the developers from protocol specifics. Currently the following JAIN Application API Specifications are under development:

- **SLEE - Specification Lead: Motorola**
- **Call Control - Specification Lead: Telcordia**
- **Coordination and Transaction – Specification Lead: Telcordia**
- **SPA Trust & Security Management, and Service Discovery - Specification Lead: Ulticom**
- **SPA Integrity Management – Specification Lead: Ulticom**
- **SPA Mobility – Specification Lead: Siemens**
- **SPA Generic User Interaction – Specification Lead: AePONA**
- **Presence and Availability Management – Specification Lead: Teltier Technologies**
- **Service Creation Environment – Specification Lead: Telcordia, Hughes Software Systems**
- **Common API – Specification Lead: Sun Microsystems**

JAIN Program Management Responsibilities

A Program Manager has been assigned by Sun with responsibility for coordinating the activities of the various Expert Groups in the JAIN community. The JAIN Program Management provides:

- Overall JAIN program strategy
- JAIN program roadmap
- Maintenance of the list of JAIN Program Members
- Drive completion of JSRs (in conjunction with the Expert Group's Specification Lead)
- Assistance in the recruitment of Experts for new JSRs
- Publish auditing process results

Levels of Participation

How To Get Involved

There are various levels of participation in the JAIN initiative. The following will describe each level of participation and how to get involved at each level.

You can participate in the Java Community Process (JCP) in four ways:

- Public
- Member
- Expert
- Specification Lead

Public:

Anyone with an internet connection can freely review and comment on:

- All specifications developed using the JCP.
- All proposals for new or revised specifications.
- All proposed error corrections and changes to existing specifications.

Member:

A Member is any individual, organization, or company that signs the Java Specification Participation Agreement (JSPA). Members enjoy all the privileges of Public participants plus they can:

- Propose new or revised Java API specification projects by filing a new JSR.
- Nominate themselves to serve on the Expert Groups that create or revise Java specifications.
- Review and comment on all specifications developed using the JCP before Public review.

Expert:

Any Member can nominate an Expert to serve on one of the Expert Groups that write Java specifications. Experts will:

- Actively shape the content and direction of new and revised Java specifications in the Expert Group.
- Review comments from Members and the Public, and use them to improve the quality of a specification.

Specification Lead:

The Specification Lead is an Expert with additional responsibilities who will:

- Serve as the lead for the API Specification and be responsible for choosing the other Members of the Expert Group.
- Assume responsibility to deliver a consensus based specification (including associated documentation), a Technology Compatibility Kit (TCK) and a Reference Implementation (RI).
- Have the option of assuming responsibility for maintaining a specification after it is written, therefore becoming the Maintenance Lead.

Note that participation in an Expert Group takes significant commitment by the Members and their respective companies. This is particularly true for Specification Leads. Prior to assuming such responsibilities, a thorough review should be undertaken as to the nature and magnitude of the commitment and the ability of the Members to follow through to completion.

Becoming a Member

You must sign a Java Specification Participation Agreement (JSPA) or a Individual Expert Participation Agreement (IEPA) in order to become a Member or Expert within the JCP. The JSPA and IEPA are one year, renewable agreements between you or your organization and Sun Microsystems, Inc. The JSPA can be found at <http://jcp.org/aboutJava/communityprocess/JSPA.pdf>. The IEPA can be found at <http://jcp.org/aboutJava/communityprocess/IEPA.pdf>. You need to sign the JSPA or IEPA with Sun to become a Member of the JAIN Community.

The following procedure should be followed:

- Download the appropriate agreement. Print the agreement, fill in required information about yourself or your organization and fax the completed agreement to Sun.
- Once the JSPA has been received, your company will be invoiced for the JSPA yearly fee if applicable (see the JSPA for details). When Sun receives your payment, we will execute the JSPA and fax a copy back to you.

Additionally, to become an active JAIN Community participant, upon signing the JSPA, send email to: JAIN-interest@sun.com and the JAIN Program Manager will contact you.

Appendix A - JAIN Community Members

- 8x8
- ADC Newnet
- AePONA
- Alcatel
- AT&T
- Bay Packets
- British Telecommunications PLC
- BroadSoft
- CCL/ITRI
- Cisco Systems
- CMG Telecommunications & Utilities B.V.
- Comstellar
- DataKinetics Ltd
- dynamicsoft, Inc.
- ECTF
- EmpowerTel Networks
- Ericsson
- Eurescom (20 European Operators)
- Forschungszentrum Telekommunikation Wien
- France Telecom
- Fujitsu Ltd
- Hitachi Ltd
- Huawei Technologies Ltd
- Hughes Software Systems
- IBM
- Incomit
- ipVerse

- Logica Mobile Networks
- Longboard
- Lucent Technologies
- Mahindra-British Telecom Ltd
- Matsushita Electric Industrial Co., Ltd.
- Motorola
- Narad Networks
- NEC Corporation
- Net4Call
- Nokia
- Nortel Networks
- NTT Communicationware
- NTT Labs
- NTT Software Corp
- OKI Electric Industry Co.
- Open Cloud Ltd
- Periphonics Corporation
- Personeta
- Pingtel
- RADVision, Inc.
- Qwest
- Samsung
- Siemens
- Sun Microsystems, Inc.
- Symsoft AB
- Taral Networks
- Tata Consultancy Services
- Telcordia Technologies
- TELSIM-Oxygen Technology
- Trillium Digital Systems
- TrueTel Communications
- Ubiquity Software Corporation
- Ulticom
- Vodafone

- Westwave Communications
- Zy Technologies

Appendix B - References

Published Articles and Books:

- Jepsen, Thomas C. (Editor), Farooq Anjum, Ravi Raj Bhat, Ravi Jain, Anirban Sharma, Douglas Tait, *Java in Telecommunications: Solutions for Next Generation Networks*. New York: John Wiley & Sons, Inc., 2001.
- D.Tait, J. de Keijzer, and R. Goedman, "JAIN: A New Approach to Services in Communication Networks" *IEEE Communications Magazine*, January 2000
- R. Jain, F. Anjum, P. Missier, and S. Shastry, "Java Call Control, Coordination, and Transactions" *IEEE Communications Magazine*, January 2000
- R. Bhat and R. Gupta, "JAIN Protocol APIs" *IEEE Communications Magazine*, January 2000
- S. Beddus, G. Bruce, and S. Davis, "Opening Up Networks with JAIN Parlay" *IEEE Communications Magazine*, April 2000

Web Resources:

- ***JAIN Interest Alias***
JAIN-interest@sun.com
- ***JAIN Discussion Lists***
<http://archives.java.sun.com>
- ***JAIN Web Resources***
<http://java.sun.com/products/jain/>
- ***Java Software and Developer Information***
<http://java.sun.com>
- ***Java Community Process***
<http://jcp.org>
- ***Parlay***
<http://www.parlay.org>
- ***JTAPI***
<http://java.sun.com/products/jtapi>