

JAINTM and Open Networks

A white paper describing the positioning of the JAIN Application Programming Interfaces (APIs) within open network architectures

August 2003

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

Copyright © 2003 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, Java Servlet, 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 PURPOSE	4
3 OPEN NETWORK TECHNOLOGY	4
4 JAIN AND OPEN NETWORKS.....	5
4.1 JAVA APPLICATION INTERFACES AND OPEN NETWORKS	6
4.2 JAVA CONTAINER INTERFACES AND OPEN NETWORKS	7
5 REFERENCES	8

1 Introduction

The desire for new business growth for communication network operators has been a major driving force towards the development of open network interfaces within communications networks. The belief is that in providing open network interfaces to core network functionality, this enabling technology will ignite a proliferation of new services that can be offered by communication network operators, thus opening up new revenue streams. Many approaches to providing open network interfaces are being developed, and trials are underway. This document provides an overview of the most promising standardization work in this area, and undertakes the difficult task of showing how they relate with each other.

2 Purpose

A specific purpose for this document is to establish a common understanding of the relationship between the JAIN APIs, the Open Mobile Alliance (OMA) interfaces and the Open Service Access (OSA) interfaces within open network architectures. It is also intended to encourage, shape and guide discussions amongst the open network community so as to facilitate a comprehensive, consistent and coordinated set of interfaces.

This document is written with the intent to inform the Java™ community in general and the JAIN Community in particular of how JAIN is architecturally positioned within the open network landscape.

3 Open Network Technology

There are many communities and initiatives set up to address the development of open network interfaces. The following describes the most current and most relevant interface standardization work that is related to communications networks:

- The JAIN Community [1] is defining a suite of Java APIs that target all aspects of communications networks. These consists of:
 - *Java Application Interfaces for Communications* that are either network protocol specific or network protocol agnostic. These APIs enable both 3rd party¹ and network operator application development upon one or more Java platforms, dependent on the specific application interface.
 - *Java Container Interfaces for Communications* that host components utilizing the above application interfaces and, hence, are either network protocol specific or network protocol agnostic. These APIs enable both 3rd party and network operator applications to run in a managed, controlled and real-time fashion.

Protocol agnostic application interfaces may be layered upon standardized or proprietary protocol specific application interfaces.

¹ External companies, such as Mobile Virtual Network Operators (MVNOs) and enterprise businesses, operating outside the security domain of the network operator.

- The OMA [2] is defining sets of Web Services (WS) interfaces, realized in Web Services Description Language (WSDL), that focus only on the access and control of network servers. These WSDL interfaces, whilst still being defined and agreed, essentially consist of:
 - *Service enablers* that are network protocol agnostic and enable mass-market 3rd party application development.
 - *Non-Service enablers* that facilitate the use of service enablers in a secure, discoverable and reliable fashion. These enablers will re-use and extend some of the already available Web Service technologies, such as Security Assertion Mark-up Language (SAML), Universal Description, Discovery and Integration (UDDI) and Java Web Services Developer Pack (JWSDP).
- The Joint Working Group² is defining a set of Unified Modelling Language (UML) interfaces that focus only on the access and control of network servers. Known as the Open Service Access (OSA) APIs, these consist of:
 - *OSA Service APIs* that are network protocol agnostic and can be viewed as next generation Intelligent Network (IN) interfaces that enable both 3rd party and network operator application development.
 - *OSA Framework APIs* that facilitate the use of service APIs in a secure, discoverable and reliable fashion.

The above three technology areas are addressing valuable pieces of the open network architecture. Whilst these technology areas are different in their own right, they are also complimentary, and it is conceivable that real ensuing open network solutions may employ varying combinations of these technologies.

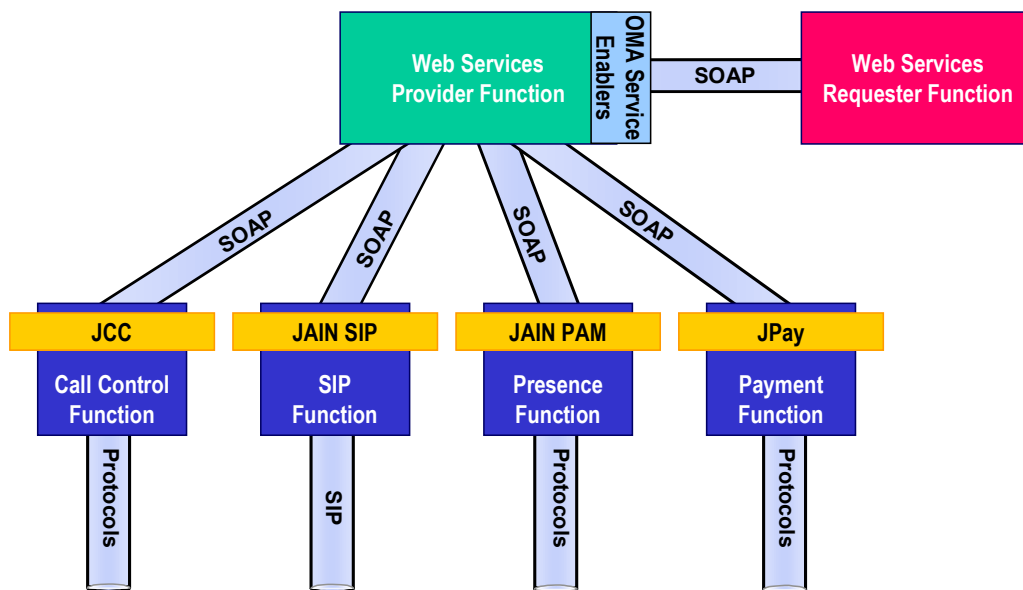
4 JAIN and Open Networks

The JAIN APIs, together with the OMA and OSA interfaces, make up a significant portion of the open network architecture. Inevitably, there are interface specification overlaps both across and within standardization activities; however, this has been kept to a minimum. At this present time the overlaps can be justified, however, through market demand and given time, the overlaps are likely to diminish. The following sections illustrate how the JAIN APIs are positioned with respect to OMA and OSA interfaces. A functional architecture positioning only the functions, interfaces and protocols is considered first; then, container technologies are overlaid on the functional architecture to provide a physical architecture.

² A collaboration between 3rd Generation Partnership Project (3GPP) [3], 3rd Generation Partnership Project 2 (3GPP2) [4], European Telecommunications Standards Institute (ETSI) [5] and Parlay [6].

4.1 Java Application Interfaces and Open Networks

Figure 1 shows how some of the Java application interfaces relate to an OMA Web Services open network architecture. The figure is completely unconstrained from deployment options; for example, any/all of the functions may be deployed either within or external to the network operator environment.



Key:

JCC = Java Call Control; JPay = Java Payment;
SIP = Session Initiation Protocol; PAM = Presence and Availability Management

Figure 1: Java Application Interfaces in the Open Network

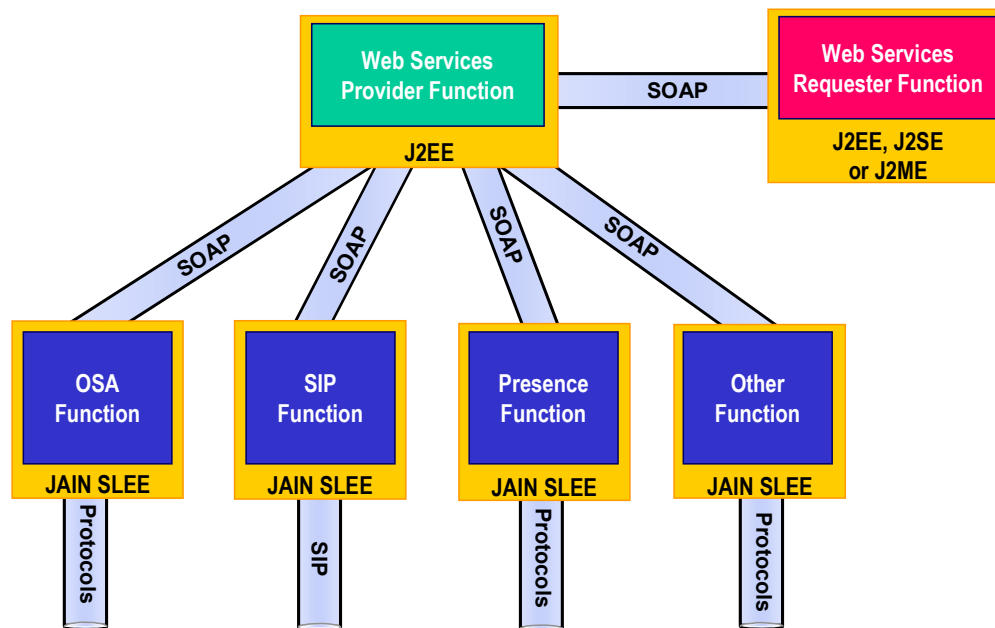
The JAIN application interfaces (orange boxes as shown in the figure) would be implemented on specialized communications functions (blue boxes), which offer discrete communications and communications-related capabilities. These specialized communications functions would communicate with other functions and/or network elements, via network protocols (blue pipe labelled 'Protocols' and 'SIP'). In addition, the specialized communications functions would service requests/messages from the *Web Services Provider* function(s) that, in turn, would service requests/messages from *Web Services Requester* function(s), such as an enterprise application. The Web Service requests/messages are shown in the figure as using Simple Object Access Protocol (SOAP) for transport (blue pipe labelled 'SOAP'). The Web Services Provider would make the OMA Service Enabler interfaces available to the Web Services Requester.

As mentioned earlier, the figure is completely unconstrained from deployment options, so it is feasible that the Web Services Provider could be combined with the specialized communication functions. It is also feasible that the use of SOAP for transport is not mandated either between the

Web Services Requester and Web Services Provider, or between the Web Services Provider and specialized communication functions.

4.2 Java Container Interfaces and Open Networks

Figure 2 shows how the Java container interfaces relate to OMA and OSA functions in the open network architecture. As with the previous figure, this figure is also completely unconstrained from deployment options.



Key:

J2EE = Java 2 platform, Enterprise Edition; SLEE = Service Logic Execution Environment
 J2SE = Java 2 platform, Standard Edition; J2ME = Java 2 platform, Micro Edition

Figure 2: Java container interfaces and OMA/OSA server functions in the Open Network

The specialized communications functions (blue boxes as shown in the figure) should typically be hosted in a JAIN SLEE container. This is because of their need to support applications with asynchronous, event-driven components that exhibit transactional characteristics requiring high performance and low latency. The exception for this is for SIP based networks, when circumstances may dictate servlet based application development specific to the SIP protocol. In this case, SIP functions may be hosted in a SIP Servlet container. Note that the above figure introduces an OSA server function, which may be an OSA client application server function or an OSA gateway server function.

The Web Services Provider function(s), depending upon deployment requirements, would be hosted in a Java 2 Platform, Enterprise Edition (J2EE™) Enterprise JavaBeans™ (EJB™) or Java

Servlet container. The benefit of J2EE is that it provides the developer with a well-defined and accepted programming model, and all the necessary Web Services tools support.

J2EE, Java 2 Platform, Standard Edition (J2SE™) or Java 2 Platform, Micro Edition (J2ME™) may host the Web Services Requester function(s), depending upon deployment requirements. J2EE containers are likely to be used if they are also supporting server functionality, such as also acting as a Web Services Provider; the J2SE platform will be used to support ‘fat’ client functions, such as desktop computers; and the J2ME platform will be used to support ‘thin’ client functions, such as mobile phones.

5 References

- [1] JAIN Community – <http://java.sun.com/products/jain>
- [2] OMA – <http://www.openmobilealliance.org>
- [3] 3GPP – <http://www.3gpp.org>
- [4] 3GPP2 – <http://www.3gpp2.org>
- [5] ETSI – <http://www.etsi.org>
- [6] Parlay – <http://www.parlay.org/>
