

A large, abstract graphic on the left side of the page, consisting of several overlapping, curved, semi-transparent shapes in shades of gray, creating a sense of depth and movement.

SUN JAVA™ SYSTEM INSTANT MESSAGING 8

Real-Time Collaboration for Service Providers
and the Extended Enterprise

White Paper
August 2009

This Page Intentionally Left Blank

Table of Contents

Executive Summary	1
Java System Instant Messaging Product Overview	4
What is Java System Instant Messaging?	4
The instant messaging environment	5
Basic LDAP-based configuration	6
Instant messaging configuration with identity	7
Interoperability	9
Multiple server configurations	10
Modular server architecture	11
Built for Service Providers and the Extended Enterprise	16
Standards-based interoperability	16
Identity-enabled instant messaging	17
Architected for high performance, scalability, and availability	18
Secure access, secure communication	23
Secure document archiving	25
Intuitive, easy-to-use, feature-rich clients	26
Presence-enabled applications	29
Open extensible platform	30
Conclusion	32
High-performance server	32
Security, flexibility, and extensibility	32
Based on standards and proven technology	33
Appendix A	34
References	34
Abbreviation/acronym list	34

Executive Summary

When deploying an instant messaging platform, an enterprise or service provider must ensure the platform addresses the needs of many users, each with a unique set of challenges, as well as the organization's strategic operations. Examples include:

- A developer who must create an application that utilizes presence and filters incoming and outgoing messages based on content
- The business' need to deploy a secure instant messaging system that provides secure transmissions as well as a secure data archive
- A service provider's goal to deploy a highly scalable, highly available instant messaging system to its customer base
- An extended enterprise that must provide real-time communication, including mobile instant messaging communications, to its partners and customers through its portal

In addition, an enterprise or service provider must also meet a variety of additional demands such as massive interoperability, enhanced security, vast scalability, and a feature-rich interface.

Sun Java™ System Instant Messaging is a comprehensive instant messaging application that can easily be extended and integrated with other components of the Java Communications Suite and Java Enterprise System to enable an enterprise to solve such complex communications challenges. Java System Instant Messaging provides the benefits of instant messaging already enjoyed by many consumers to both service providers and enterprises, while addressing the critical issues of security, reliability, extensibility, scalability, directory integration, and archive capability. It complements the functionality of the other Java System Communications products such as the Java System Messaging Server and the Java System Calendar Server and supports the eXtensible Messaging and Presence Protocol (XMPP), thereby enabling it to interoperate with many popular instant messaging systems through XMPP gateways. Java System Instant Messaging also enables users across the extended enterprise to communicate and collaborate instantly and securely, combining presence-awareness with instant messaging (IM) capabilities such as chat, alerts, news, polls, and file sharing to create a rich collaborative environment.

The Java System Instant Messaging solution's modular, flexible architecture is scalable both vertically, by increasing the number of CPUs and/or memory per system, and horizontally, by introducing additional servers and multiplexor components into the network. Its vast scalability and unparalleled security make it ideally suited for service providers seeking to add instant messaging to their list of offerings. It supports both

Lightweight Directory Access Protocol (LDAP)-based and identity-based deployments, and integrates with the Java System Access Manager and the Java System Messaging Server to provide additional functionality. Java System Instant Messaging leverages:

- The community management, policy management, service management, and single sign-on (SSO) capabilities of the Java System Access Manager, providing easy integration with other Java System Communications products
- The archive and search capabilities of the Java System Messaging Server

Java System Instant Messaging is a component of the Sun Java Communications Suite, an open and integrated infrastructure software system that delivers industry-leading email, calendaring, and real-time collaboration functionality for service providers and large organizations worldwide. Java System Instant Messaging uses Java technology, including Java Foundation Classes/Swing as its client component and XMPP as its communication protocol. Messages can contain embedded hyperlinks that open in a full browser window. Java System Instant Messaging is based on open standards such as XMPP, HyperText Markup Language (HTML), TCP/IP, and Java technology, so it can integrate with other software as part of a total messaging solution. Because both the server and the client support XMPP, the client can interoperate with other XMPP-compliant servers and the server can interoperate with other XMPP servers, clients, and gateways. This level of interoperability provides a great advantage to those enterprises seeking to extend their ability to practice real-time communication with their partners and customers, many of which may have separate instant messaging systems.

Java System Instant Messaging delivers the following capabilities, many of which differentiate it from competitive offerings:

- Provides secure instant collaboration for service providers and users across the extended enterprise to enable rapid information dissemination and decision-making
- Delivers interoperability with other instant messaging systems such as AOL AIM, Yahoo Messenger, and MSN Messenger through currently available XMPP gateways. Co-packages AIM, Yahoo, and MSN gateways for interoperability with AOL Instant Messaging, Yahoo Messenger, and MSN Messenger clients
- Delivers interoperability with other XMPP-based servers, components, gateways, and clients, including open source components, gateways, and clients as well as mobile clients
- Delivers a fully integrated solution comprised of chat, presence, e-mail, calendar, and address book through Sun™ Convergence, a state-of-the-art AJAX-powered Web 2.0 client
- Supports Open Mobile Alliance (OMA) Instant Messaging and Presence Service (IMPS) protocol which enables mobile instant messaging and presence services. Includes an IMPS gateway for interoperability with IMPS 1.1 capable mobile clients. Supports multi-user chat for IMPS capable mobile clients

- Includes instant messaging, chat, alerts, polling, news channels (virtual bulletin boards), and file transfer
- Supports client-to-client voice chat using the Jingle protocol extensions (XEP 0166, XEP 0167, XEP 0177)
- Facilitates group or project-based collaboration through dynamic contact lists, group chat, persistent conference rooms, moderated conference rooms, real-time polling, and file sharing
- Includes a Short Message Service (SMS) gateway that enables the delivery of chat messages and alerts in the form of SMS messages to instant messaging users who are offline
- Integrates with the Java System Messaging Server to forward instant messaging alerts to offline users and to provide message archiving in the message store. The message archiving capability works with virtually any messaging server
- Integrates with the Java System Calendar Server to provide automatic calendar pop-up reminders
- Provides native LDAP support for directory authentication and search. Provides support for LDAP failover such that if one LDAP server becomes unavailable, the Instant Messaging Server is then able to fail over to another LDAP server
- Integrates with the Java System Access Manager to provide identity-enabled instant messaging incorporating policy management, community management, authentication, and single sign-on; takes advantage of the Java System Access Manager's powerful, flexible administrative interface for administrating globally, by organization, by role, or by user
- Employs a standards-based design using XMPP, HyperText Transfer Protocol (HTTP), TCP/IP, and Java technology
- Provides Transport Layer Security (TLS) support for secure server-to-server communications and secure client-to-server communications
- Provides legacy Secure Sockets Layer (SSL) support for clients that do not support TLS
- Supports product branding as well as modification of client functionality through a customizable interface
- Includes developer application programming interfaces (APIs) to allow applications to access instant messaging and presence services
- Provides server support for Linux, and the Solaris™ Operating System (OS), and client support for Microsoft Windows, Linux, Mac OS X, and the Solaris OS
- Supports federation of deployments, thereby enabling multiple IM communities to be joined together into a single federated entity
- Support for hosted or virtual domains
- Support for hundreds of thousands of users through a highly available, scalable, distributed architecture

This paper explores the architecture, design, deployment features, and benefits of the Java System Instant Messaging solution.

Chapter 1

Java System Instant Messaging Product Overview

What is Java System Instant Messaging?

Java System Instant Messaging is a Java technology-based product that enables users to communicate and collaborate instantly and securely. It provides the ability to participate in instant messaging (IM) and group chat sessions, share instant information through news channels or virtual bulletin boards, conduct real-time polls, and generate and view immediate alerts on important subjects.

Java System Instant Messaging is suitable for both intranets and the Internet, and can be deployed in a single-server configuration, a multi-server configuration with multiple servers supporting the same domain, a hosted domain configuration, or in a federated environment consisting of multiple IM servers communicating with each other over separate domains. It can work in cooperation with other components of the Java Communications Suite such as the Java System Messaging Server and the Java System Calendar Server. In addition, Java System Instant Messaging can be deployed with an LDAP directory server or in conjunction with the Java System Access Manager to provide an identity-enabled solution that takes full advantage of key services offered by these products, such as the authentication service, session service, logging service, and single sign-on.

Java System Instant Messaging can be configured to communicate over SSL and TLS, including server-to-server communications and client-to-server communications. And through its modular architecture, Java System Instant Messaging allows for great scalability both horizontally (by adding servers and components) and vertically (by adding CPUs and memory).

Java System Instant Messaging utilizes one of two standalone Java technology-based clients to provide the native instant messaging user interface:

- The Java Web Start application
- A browser-based applet via a Java Plug-in product

Java System Instant Messaging also utilizes a highly configurable extensible AJAX Web client, Sun™ Convergence, that combines e-mail, calendaring, address book and instant messaging into a rich integrated Web experience. Single sign-on and tight integration enable users to view a contact's presence from within the e-mail and calendar applications.

Java System Instant Messaging uses XMPP as both its client and server communications protocol. As a result, the standalone client can communicate with virtually any XMPP-compliant server. Similarly, the server can interoperate with other XMPP-compliant servers, gateways, and clients.

The instant messaging environment

Java System Instant Messaging is a component of the Java Communications Suite, an open and integrated infrastructure software system that provides industry-leading email, calendaring, and real-time collaboration functionality for service providers and large organizations worldwide. The Java System Instant Messaging architecture consists of a set of core components and auxiliary components that work in cooperation with each other and with other components of a networked environment to produce an instant messaging capability. Java System Instant Messaging includes:

- Core Components
 - Instant Messaging Server (IM Server)
 - Instant Messaging Multiplexor (IM Multiplexor)
 - Instant Messenger (IM client)
- Auxiliary Components
 - Web container, such as a Web server or application server (required for deploying the Instant Messenger client component)
 - Directory server (required for LDAP-based deployments)
 - Access Manager (required for identity-based deployments)

Although not a part of the core components for instant messaging, the auxiliary components are required in certain instant messaging environments and are provided for use with the Java Communications Suite. However, in addition to the core and auxiliary components, there are other components that can enhance the instant messaging environment in numerous ways. Some of these components include an IM Redirector for use when multiple IM servers are involved; an HTTP Proxy Gateway for use with HTTP clients, mobile clients, or firewall traversal; an IMPS Gateway for use with IMPS capable mobile clients; an AIM gateway for use with AOL Instant Messaging clients; an MSN gateway for use with MSN Messenger clients; a Yahoo gateway for use with Yahoo Messenger clients; user provisioning tools; and more.

Java System Instant Messaging is capable of running in a minimal environment that consists simply of the Instant Messaging Server, Instant Messaging Multiplexor, Instant Messenger client, and Web container; however, the more interesting and certainly more common environments will be LDAP-based and identity-based. The diagrams in the next few pages (Figures 1, 2, 3, and 4) illustrate the relationship between the components that could comprise each of these environments:

- Figure 1 depicts a simple LDAP-based deployment
- Figure 2 provides an example of a more functionally rich deployment using the Java System Access Manager
- Java System Instant Messaging is also capable of running in a mixed-vendor environment consisting of other XMPP-compliant gateways, servers, clients and other IM systems, as depicted in Figure 3
- Multiple Instant Messaging Servers can be configured to communicate with each other within a single domain or between multiple domains, as illustrated in Figure 4

Basic LDAP-based configuration

At a minimum, the following components are required in the LDAP-based instant messaging environment: The Instant Messaging Server, Instant Messaging Multiplexor, Instant Messenger, LDAP directory server, and Web container. In this basic configuration, it is assumed that the native Instant Messenger client is used. However, this is not required. Since the instant messaging server communicates over XMPP, virtually any XMPP-compliant client may be used. If a customer decides to standardize on an XMPP client other than the native Sun client, the Web container will no longer be necessary since the purpose of the Web container is to deploy the Sun client.

Although the Simple Mail Transfer Protocol (SMTP) server is not required to facilitate instant messaging, it is necessary if the user wishes to take advantage of the alert-forwarding feature. In the portal/identity environment, many of these components are present after installing and running the portal server. For example, if the Java System Portal Server is already deployed, there is no need to install an additional directory server or an additional Web container because instant messaging can use the same directory and Web container already deployed with the portal server.

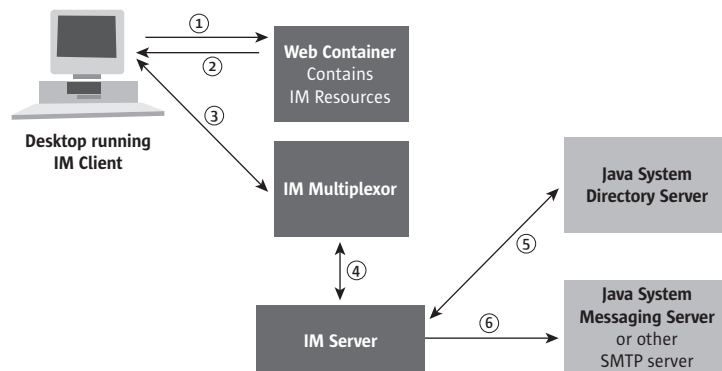


Figure 1. Java System Instant Messaging LDAP-based environment

In simplistic terms, the relationship between the various components in the Java System Instant Messaging LDAP-based environment is as follows:

1. The user enters the Web container's URL in a Web browser running on the client desktop machine.
2. The browser invokes Java Web Start or the Java Plug-in. Java Web Start or the Java Plug-in downloads the necessary Instant Messenger resource files and starts the Instant Messenger (IM client).
3. The login window is displayed and the user enters the LDAP user name and password. The data is sent to the Instant Messaging Multiplexor (IM Multiplexor).
4. The IM Multiplexor routes the data received from the Instant Messenger to the backend Instant Messaging Server (IM Server).
5. The IM Server communicates with the LDAP server to authenticate the user. When the user authentication is complete, the IM client displays the contact list for the user. The user can now participate in instant messaging sessions with other users.
6. An SMTP server such as the Java System Messaging Server, when notified by the IM Server that users are offline, forwards alerts to their e-mail. Users must set their preferences to have alerts forwarded as e-mail when they are offline. The SMTP server can also be used to archive instant messages.

Instant messaging configuration with identity

The environment depicted in Figure 2 consists of the Java System Access Manager, Instant Messenger, LDAP directory server (Java System Directory Server), SMTP server (Java System Messaging Server), and the Java System Calendar Server. The Java System Access Manager host itself contains the Java System Access Manager and Admin Console, Instant Messaging Server, Instant Messenger Resources, Instant Messaging Multiplexor, and Web container. Neither the Instant Messaging Server nor the Instant Messaging Multiplexor need to run on the same host containing the access manager software, however, it is shown this way for simplicity. A portal server has been added to this configuration to demonstrate how a portal could utilize access manager to generate a session token that can be used to establish an instant messaging session.

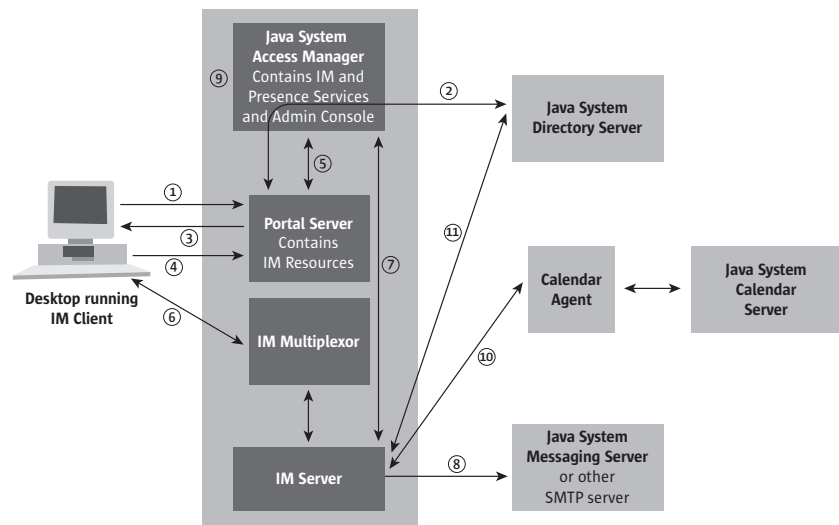


Figure 2. Java System Instant Messaging Identity-based environment

In simplistic terms, the relationship between the various components in the Java System Instant Messaging identity-based environment is as follows:

1. The user logs on to a portal server by entering the appropriate URL in a Web browser and inserting the proper user name and password.
2. The Java System Access Manager authenticates the user with the configured authentication mechanism, communicating with the LDAP directory server to get the user ID. The Access Manager returns a session token which is later used to authenticate the instant messaging user.
3. The portal server generates the user's portal Desktop and serves it to the client browser. This Desktop contains a link to launch the Instant Messenger application.
4. The user clicks the Instant Messenger link in the portal Desktop instant messaging channel.
5. Software running on the portal server validates the user and the Instant Messenger is launched. The Web container within the portal server provides the necessary client files to the IM client.
6. The Instant Messenger connects to the IM Multiplexor and requests authentication to the IM Server using the session token.
7. The IM Server asks Access Manager to validate the session token. If the session is valid, Instant Messenger displays the end user's contact list and the end user can then use the Instant Messenger services such as chat, alerts, polls, and so forth.

8. An SMTP server such as the Java System Messaging Server, when notified by the IM Server that users are offline, forwards alerts to their e-mail. Users must set their preferences to have alerts forwarded as e-mail when they are offline.
9. The Java System Access Manager Admin Console is used to add and delete user IDs, change passwords, and provide access control.
10. The Calendar Agent subscribes to Java System Calendar Server events and generates an alert to the IM Server, which manifests itself as a pop-up notification or calendar reminder on the client desktop.
11. The Instant Messaging Server must query LDAP directly to get or set end-user information, such as contact lists or subscriptions.

Interoperability

The environment depicted in Figure 3 consists of Java System Instant Messaging and other XMPP-based components, thereby enabling interoperability with other instant messaging systems such as AOL AIM, Yahoo Messenger, and MSN Messenger. Open source and third-party XMPP clients may communicate directly with Java System Instant Messaging. Likewise, the Instant Messaging Server communicates to other open source and third-party servers and gateways using XMPP. The purpose of the gateway is to convert XMPP protocol directives into the native protocol that the particular IM system understands. Hence, in the case of AOL, an XMPP to AOL AIM gateway provides the bridge necessary to support communication between the XMPP network and the AOL network. Java System Instant Messaging co-packages AIM, MSN, and Yahoo gateways for out-of-the-box interoperability with AIM, MSN Messenger, and Yahoo Messenger. Figure 3 also illustrates the use of an HTTP Gateway in this configuration. This gateway is provided with Java System Instant Messaging and is XEP 124 compliant, making it useful for enabling mobile instant messaging communications, HTML based IM communications or simple firewall traversal. Sun Convergence also provides an HTTP gateway specifically for use with the instant messaging component of Convergence. Either or both of these gateways can be used in the instant messaging environment. Java System Instant Messaging also provides interoperability with SMS networks through an SMS gateway. The SMS gateway converts the chat message to an SMS message for forwarding to a mobile device when a user is offline. Finally, this figure illustrates the use of an IMPS Gateway, also provided with Java System Instant Messaging, to enable mobile instant messaging communications with a large number of IMPS capable mobile phones.

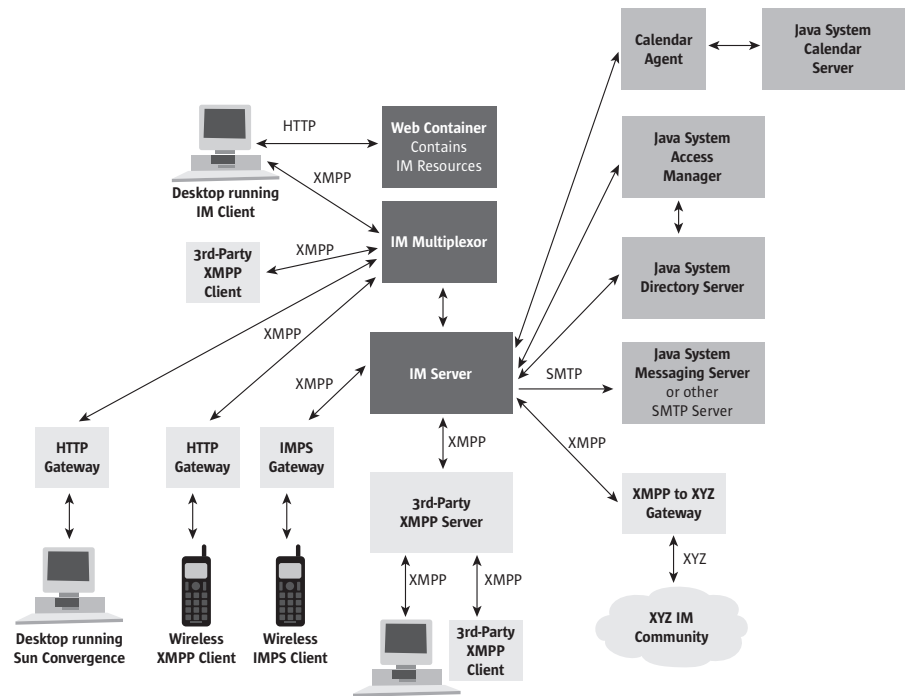


Figure 3. Java System Instant Messaging mixed-vendor environment

Multiple server configurations

There are essentially two types of multiple server configurations: Multiple server with multiple domains, and multiple server with a single domain. In the multiple server, multiple domain configuration, each IM server represents a separate IM domain. Domains are then federated together to produce a large IM community. In the multiple server, single domain configuration, additional IM servers are added to a single domain in order to support a growing user population. The multiple server, single domain configuration makes use of an IM Redirector component that is not utilized in the multiple server, multiple domain configuration. The IM Redirector is used to connect the client to the correct server when multiple servers are deployed in a single domain.

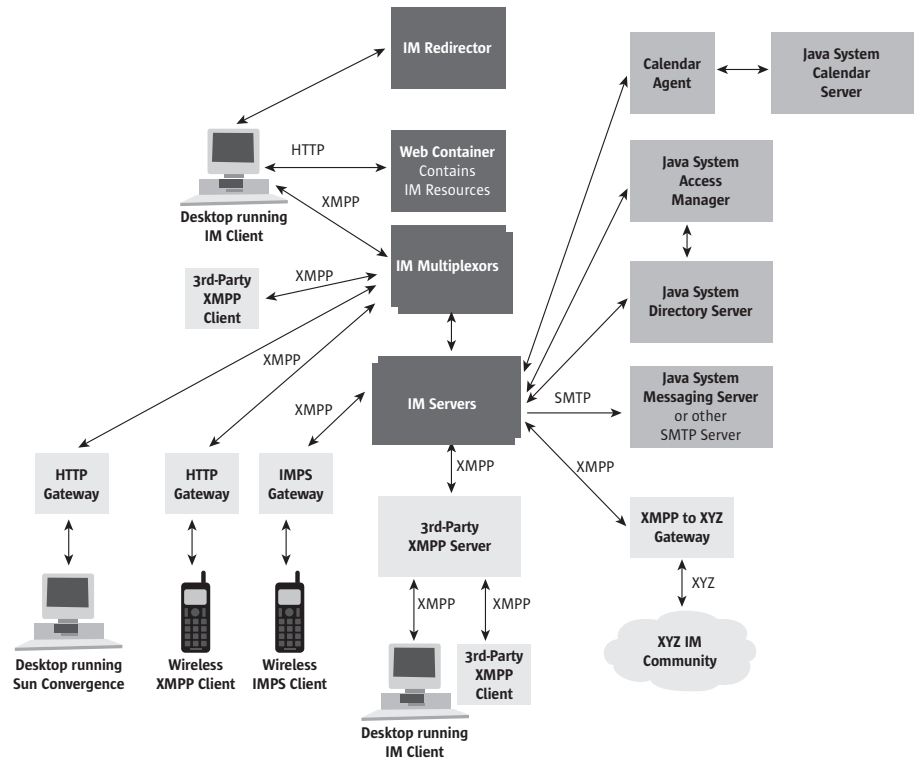


Figure 4. Multiple IM Server mixed vendor environment

Modular server architecture

The Java System Instant Messaging architecture consists of a set of internal core components that must interoperate with external services in order to provide an instant messaging environment. The role of each of these components and services is as follows.

Instant Messaging Server

The Instant Messaging Server handles tasks such as controlling client privileges and security, enabling Instant Messenger clients to communicate with each other by sending alerts, initiating conferences, and posting messages to available news channels. The Instant Messaging Server supports the connection of an Instant Messaging Multiplexor that concentrates connections over one socket.

Instant Messaging Multiplexor

The role of the Instant Messaging Multiplexor is to add scalability to the instant messaging environment. The administrator can install multiple multiplexors as needed, depending on the configuration. As the user population grows beyond that which is easily supported by a single instant messaging server, additional instant messaging servers can be deployed within the IM environment to which additional multiplexors are connected.

Instant Messenger Client

The Instant Messenger, written in the Java programming language, is a client that can be configured to be browser-based (applet) or independent of a browser (the Java Web Start application). The client is actually a set of files deployed in a Web container and downloaded by the browser applet or the Java Web Start application. The Instant Messenger provides the following communication modes:

- Chat — Chat is a real-time conversation capability that enables users to complete projects, answer customer questions, and complete other time-critical work assignments. Essentially, it is instant messaging.
- Alerts — Alerts are time-critical messages that users instantly receive.
- Poll — The polling function enables a user to poll other users for their response to a question. Polls can be used to provide instant surveys.
- News Channels — News channels are essentially virtual bulletin boards or forums for posting and sharing information. Users subscribe to news channels of interest to see updates.

XMPP/HTTP Gateway

The XMPP/HTTP Gateway, also referred to as the HTTP Proxy Gateway, is an optional component of Java System Instant Messaging. It is not required to establish basic instant messaging functionality, but it is used to provide extensibility to other client types such as mobile clients or Web-based HTML clients that are compliant with the XMPP Extension Protocol (XEP) 124 regarding HTTP Binding.

This gateway is also useful for firewall traversal. Typically, firewalls may not allow XMPP traffic in but will allow HTTP traffic. Users outside the firewall cannot connect to the XMPP server directly but will be able to communicate with the server through the gateway. In the case of mobile and HTML clients, most of these are not capable of having a dedicated TCP connection to the XMPP server. However, they can connect to an HTTP server and post or retrieve data from it. Essentially, the gateway proxies instant messaging traffic to the XMPP server on behalf of HTTP clients. The XMPP/HTTP Gateway is deployed as a webapp in the web container.

Sun Convergence, the integrated AJAX Web client provides its own HTTP Gateway that is deployed in the Java System Application Server. This gateway is part of Sun Convergence and can also be deployed in the instant messaging environment to support the instant messaging and presence functionality within Convergence. Both gateways can be deployed simultaneously as long as unique identifiers are assigned to each gateway.

AIM, Yahoo, and MSN Gateways

The AIM, Yahoo, and MSN Gateways are open-source gateways that are co-packaged with Java System Instant Messaging. These gateways enable Java System Instant Messaging users to communicate with their contacts on AOL Instant Messaging, Yahoo Messenger, and MSN Messenger. Co-packaging the gateways enables easy installation and configuration of the gateways with Sun Java System Instant Messaging.

SMS Gateway

The SMS Gateway is an optional component of Java System Instant Messaging that provides a connection between the XMPP network and the Short Message Service (SMS), enabling users to deliver chat messages and alerts to offline contacts in the form of SMS messages. The SMS Gateway uses the Short Message Peer-to-Peer (SMPP) protocol and XMPP to convert the XMPP message to a series of SMPP Protocol Data Units (PDUs), and submits these SMPP PDUs to a Short Message Service Center (SMSC), which is responsible for delivering the messages to the mobile phone. Because there is a character limit per SMPP PDU, the SMS Gateway splits the instant messages into several SMS messages if the length of the XMPP instant message exceeds a certain value. The SMS Gateway can be configured to reject XMPP messages that exceed a given value, and it can also be configured to limit the number of SMS messages that can be sent per sender, per hour.

IMPS Gateway

The IMPS Gateway is an optional component of Java System Instant Messaging and is used to enable instant messaging communications with OMA IMPS capable mobile phones. With the IMPS Gateway, users can chat with each other, view their contact's presence, participate in conferences or group chat sessions, and manage conferences from a mobile device. Conversations and presence updates can occur between mobile devices or between mobile and desktop users. The IMPS Gateway is deployed as a webapp in the web container. Because the IMPS Gateway multiplexes multiple mobile client connections into a single connection to the IM Server, it can be thought of as a multiplexor for IMPS clients.

IM Redirector

The IM Redirector is useful in multiple server deployments (also referred to as multi-node deployments) where all the IM servers support a single IM Domain. Typically, these are large-scale deployments or deployments that anticipate growth. The job of the redirector is to redirect the IM connection to the appropriate multiplexor/server combination based on an algorithm that examines the user's contact list to determine the best server to use. If a given user's contacts are primarily on server1, the redirector will direct that user to server1 as well.

Web Container

The Web container is an auxiliary component of Java System Instant Messaging and is required if the Instant Messenger client is used in the instant messaging environment or if the IMPS or XMPP/HTTP Gateway is deployed. The main purpose of the Web container is to deploy the client. It serves up HTML, including:

- An initial index.html file, provided by the product or a customized home page, with a link to invoke the Instant Messenger
- The product's client .jar files (imbrand.jar, imres.jar, imnet.jar, imjni.jar, and so on)
- The Instant Messenger online help
- Embedded URLs in messages and news channels

The Instant Messenger Resources must be installed on the same host where the Web container is installed. If the Instant Messaging Server is deployed in a portal environment, there is no need for a separate Web container, since Java System Instant Messaging could use the same instance of the Web container that the portal server uses.

LDAP Directory Server

The LDAP Directory Server is responsible for user authentication and searching for users. It can also be used for storing user preferences such as contact lists. In an LDAP-based deployment, Java System Instant Messaging requires an external LDAP directory server and includes a restricted license of the Java System Directory Server for use with Java System Instant Messaging. In this case, the external directory server provides authentication, user search, and preference storage functionality. When Java System Instant Messaging is deployed with the Java System Access Manager or a portal server, the directory server is typically used for search and preference storage only. Authorization is provided by the identity session service in which the user ID is derived from the identity session ID. There is no need to consult the directory server to obtain this information. Java System Instant Messaging supports LDAP failover and hence may be configured to have multiple backend LDAP servers. If one LDAP server becomes unavailable, the Instant Messaging Server is then able to fail over to another LDAP server.

Java System Access Manager

The Java System Access Manager provides an interface for managing users, policies, and services for organizations using the Java System Directory Server. It also provides the authentication plug-in solution and single sign-on API. With respect to Java System Instant Messaging, the Java System Access Manager can be used to manage the instant messaging and presence services or specific attributes of these services. It can be used to define policies for the IM service or its attributes and to establish access control for users or communities of users based on organization or role. Java System Instant Messaging does not require the Java System Portal Server to take advantage of the Java System Access Manager and can be integrated with the Java System Access Manager alone.

Java System Messaging Server

The Instant Messaging Server uses an SMTP server such as the Java System Messaging Server to forward alerts as e-mail to users who are offline and have configured their preferences to use this feature. The Java System Messaging Server also provides archive capability for the instant messaging environment. Virtually any SMTP server that can access a message store can be used to provide the archive.

Java System Calendar Server

Java System Instant Messaging can also interface with the Java System Calendar Server through a calendar agent to provide automatic pop-up reminders for calendar appointments. The calendar agent is a separate component that can run independently on a system other than the Instant Messaging Server host machine. The agent registers itself to the Java System Calendar Server's notification service. Upon receiving an event notification, the agent generates an alert message and sends it to the server. This alert message manifests itself as a pop-up notification. The advantage of using an agent to generate the calendar alert is that this approach is modular, distributed, and extensible — without changing the server.

Chapter 2

Built for Service Providers and the Extended Enterprise

In order for an instant messaging system to meet the needs of service providers and the extended enterprise — consisting of internal employees, partners, and customers — the instant messaging system must be able to:

- Interoperate with other instant messaging communities
- Identify the user and provide the appropriate interface based on the user's role
- Scale with the number of users
- Provide secure access and secure communication for internal employees and external partners and customers
- Provide a feature-rich customizable client, secure document archiving system, and an open extensible platform that enables developers to integrate other applications into the instant messaging platform

Standards-based interoperability

As enterprises attempt to improve real-time communications among departments, field personnel, partners, customers, and other communities, interoperability plays a key role in extending communications beyond traditional boundaries. Certain communities of users may use public IM services such as AOL AIM; others may use open source solutions. Without interoperability, instant messaging communities would continue to exist in isolation and provide a real-time communication service to only those in the immediate community.

Java System Instant Messaging is an XMPP-based solution and uses XMPP as its native communications protocol. Consequently, it takes advantage of the features and services that XMPP provides. Such features include:

- Interoperability with XMPP-compliant systems and proprietary IM systems through XMPP gateways
- Ability to leverage off-the-shelf third-party components and tools from the open source Jabber community, thereby giving a customer maximum flexibility; components may include agents such as calendar agents, directory agents, or business application agents
- The advantage of a complete specification rather than one that is only partially complete; a partially complete specification leads to many differences in implementation — differences that affect interoperability
- Robust security and congestion control
- Straightforward client-server architecture
- Low resource requirements and low overhead in packet size for presence packets

By leveraging the advantages of XMPP, Java System Instant Messaging can offer customers the means to achieve immediate interoperability with several non-Sun instant messaging systems. This level of interoperability is not currently available with products that do not communicate using XMPP. In addition to supporting XMPP, Java System Instant Messaging supports the Open Mobile Alliance IMPS protocol that enables mobile instant messaging and presence services. Many mobile phones, particularly Nokia handsets, provide an IMPS capable instant messaging client. Java System Instant Messaging can immediately leverage the existence of many of these clients and extend instant messaging capabilities to them. Java System Instant Messaging also provides great advantages to the customer through its integration with other components of the Java Communications Suite and the Java Enterprise System. Finally, standards-based interoperability assists developers by greatly reducing the development time of new applications that enhance the overall functionality of the instant messaging solution.

Identity-Enabled Instant Messaging

Identity-Enabled Instant Messaging is useful for traditional businesses as well as extended enterprises. Traditional businesses may seek to deploy a secure instant messaging system in which user privileges are determined by the role of the user, and extended enterprises will find identity-enabled instant messaging useful for providing real-time communication to their partners and customers. When Java System Instant Messaging is deployed with the Java System Access Manager, an instant messaging service and presence service are added to the Java System Access Manager. The instant messaging and presence services enable the administrator to enforce policy mechanisms for accessing Java System Instant Messaging. The administrator can set a variety of instant messaging and presence service attributes, such as the ability to:

- Access conference rooms
- Send alerts
- Transfer files
- Access presence

Policy enforcement can occur at the individual level, the organizational level, or even the role level. For example, the administrator can enable teachers to send polls, but deny this feature to students. Identity-enabled access control effectively enables the administrator to add or remove functionality from the Instant Messenger graphical user interface. When used in conjunction with an identity-enabled portal server, Java System Instant Messaging provides chat services, alert services, news services, and poll services to the portal server user community. It can make use of existing portal services to deliver immediate and in-context communication and collaboration to portals. A user's context is based upon use of the portal, including user login, role, workgroup or project team, user location, and more. These features can be provided in the context of business-to-business (B2B), business-to-consumer (B2C), and business-to-employee (B2E) portals.

Architected for high performance, scalability, and availability

Service providers and large enterprises require an instant messaging platform that is highly scalable, delivers outstanding performance, and is highly available. Java System Instant Messaging is based on a flexible architecture where the underlying components can coexist on a single system or be distributed across a variety of systems to provide additional scalability. The core component of Java System Instant Messaging is the IM Server. The IM Server provides all the intelligence, such as the ability to manage presence, control client privileges and security, enable client chat, and manage news channels, alerts, and polls. The IM Server is written using the Java programming language. All presence status is stored in the IM Server memory for optimal response times.

The IM Server requires a multiplexor to enable highly scalable communication to thousands of clients. The IM Multiplexor is responsible for bundling many client connections into a single connection to the IM Server. Without the IM Multiplexor, the instant messaging environment would only be able to support as many client connections as can be supported by the Java Virtual Machine instances on the IM Servers.

The IM Multiplexor solves this problem by multiplexing thousands of client connections onto a single connection to the IM Server. The IM Multiplexor is solely a connection multiplexor; it is not a proxy. It is a transparent pass through. It listens for Instant Messenger clients and opens only one connection to the back-end Instant Messaging Server. The IM Multiplexor reads data from the Instant Messenger client and writes it to the IM Server. Similarly, when the IM Server sends data to the Instant Messenger client, the IM Multiplexor reads the data and writes it to the appropriate client connection. The IM Multiplexor does not perform any user authentication or parse the client-server protocol; it always acts as a front-end component to the IM Server.

All client connections must pass through a multiplexor, and all communications to the IM Server through the IM Multiplexor are over TCP/IP; there are no direct connections from the Instant Messenger client to the IM Server. If the IMPS Gateway is deployed, the gateway actually performs the role of the multiplexor for IMPS clients. The multiplexor can run on the same physical hardware as the IM Server, but it is not necessary. There can be several multiplexors communicating to the same IM Server. In this case, it is best to deploy the multiplexors on separate machines from the IM Server. Figure 5 illustrates the use of multiplexors in a deployment.

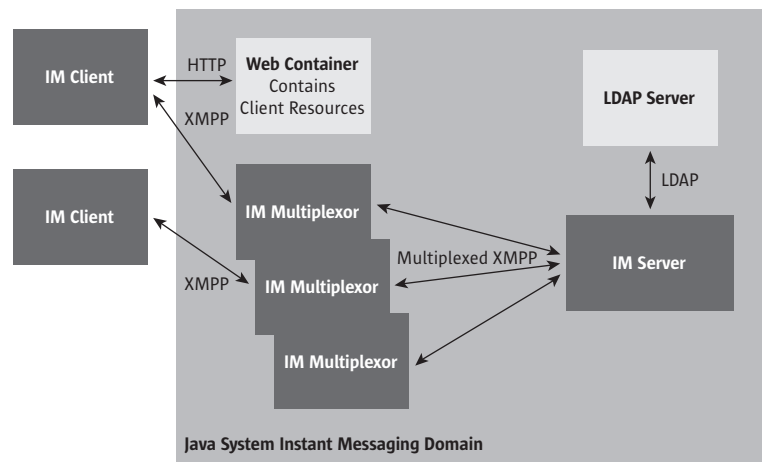


Figure 5. Use of multiplexors in the Java System Instant Messaging environment

Multiple IM servers can communicate with each other. There are two distinct configurations that support this: server federation and server pooling. In the federated case, multiple IM servers are federated to form a larger IM community. In this configuration, each server represents a separate instant messaging domain, although it is possible for a server to host multiple domains. In an LDAP-only deployment, end users from different servers can communicate with each other, use conference rooms on other domains, and subscribe to news channels on remote servers based on access privileges. In the federated case, since each IM server represents a separate domain, each will naturally use a separate LDAP directory server or at least point to a different part of the directory tree to ensure that the namespace within the IM domain is unique (where namespace is defined by a node in the directory under which all user IDs are unique). It is possible to configure the IM Servers to use the same directory server as long as each server points to a different namespace within the directory tree; the IM Server will simply use everything underneath that part of the directory tree. Figure 6 illustrates the use of server-to-server communications in a federated domain. Server-to-server communications enables the enterprise to join separate organizations together into a single federated entity.

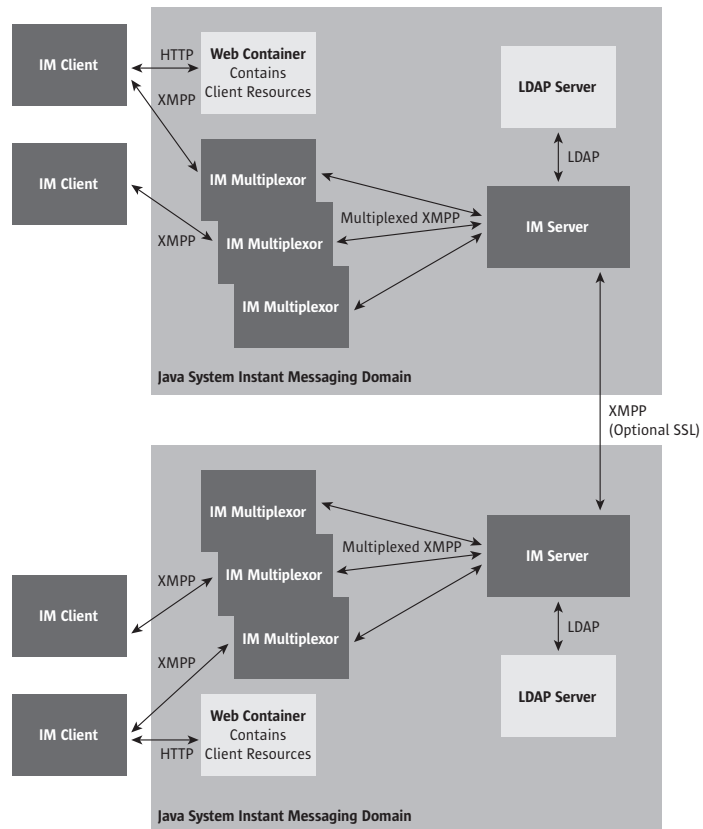


Figure 6. Federated server-to-server communications

Server pooling is very distinct from server federation. There is no requirement to limit additional IM servers to their own unique directory namespaces. Rather, the administrator creates a server pool consisting of multiple IM servers that communicate with each other over the server-to-server port and get user data from the same storage location. All servers point to the same directory namespace. The number of users that can be supported in an instant messaging deployment is no longer constrained by the capacity of a single server system. Instead, the resources of several systems can support the users in a single domain. In addition, server pools provide redundancy so that if one server in the pool fails, affected clients can reconnect and continue their sessions through another server in the pool with a minimum of inconvenience. Deploying more than one server in a server pool creates a multi-node deployment. Figure 7 illustrates the use of server pooling to provide vast scalability.

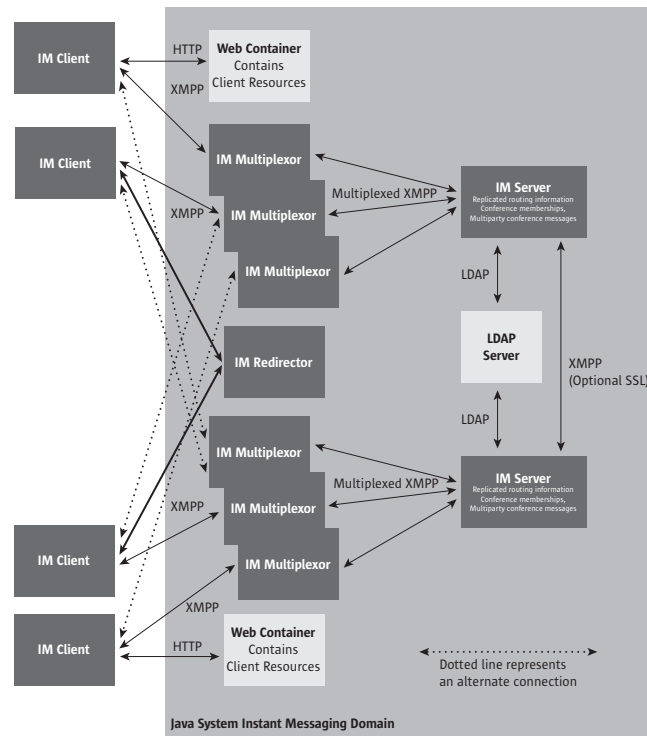


Figure 7. Server pooling

The use of server pools provides redundancy and is one way to establish a highly available network. Another way to ensure reliability is to configure Java System Instant Messaging to be highly available by using the Sun Cluster technology. A Sun Cluster is a loosely coupled system of nodes that provides a single system image to clients of network services or applications such as Web, mail, or instant messaging services. A key feature of Sun Cluster software is that upon detecting a fault, it transparently relocates a service's daemons from one node to another, including host name, Internet Protocol (IP) address, and access to devices configured as part of the service, thus providing a single system image. This capability provides automatic failover when a system shutdown or failure occurs.

Java System Instant Messaging supports all high-availability topologies that are supported by Sun Cluster technology in asymmetric, symmetric, and N+1 configurations. In the asymmetric configuration, the instant messaging service daemons run only on the primary node and not on the secondary. Detection of a fault in any of the resources (such as storage, host system, or instant messaging daemons) causes the instant messaging service to be stopped on the primary and started on the secondary. In the asymmetric configuration, each node in the cluster is a complete Solaris™ Operating System (OS) installation, with its own disk that contains the OS, the Java System Instant Messaging binaries, and instant messaging server cluster agents. Each node has at least one network interface connected to the public network and at least two

additional private network interfaces that connect to corresponding private network interfaces on the other cluster members. These interfaces are used by the Sun Cluster software on each node for system status monitoring and cluster configuration data sharing. Only one pair of private network interfaces is in use at any given time; the other is a redundant interface, so there is no single point of failure.

Each node has a connection to the disk cluster that contains the database and configuration files. Although both nodes are continuously connected to the disk cluster, the volumes in the disk cluster are most likely mounted on only one of the nodes at any given time using HAStoragePlus technology. .

Only one node in the cluster runs the Java System Instant Messaging Server at any given time. In an asymmetric HA configuration, the configuration files and the IM database may reside on a shared disk. As a result, when a failover occurs, the disk may be unmounted from the failing system and mounted on the surviving system. The logical IP address is then configured on the public network interface of the other system. Users and agents on the public network always connect by using the logical IP address. Reconnecting after a failure automatically connects to the other system. A failover appears to be a very quick disconnection and reconnection of a single system.

In the Java System Instant Messaging asymmetric HA configuration, the secondary node in the cluster is idle as far as Java System Instant Messaging usage is concerned. This node, however, remains a fully functional Solaris OS system, and is available for other work as long as procedures to terminate or limit the other work after a failover are in place. When the CPU speed and memory size of the two nodes in the cluster are alike (recommended) in this asymmetric HA configuration, performance does not suffer during a failover.

In the symmetric configuration, two IM services are configured to run, each connected with its own logical host. Normally, logical hosts are mastered by two different nodes. When one service fails, both logical hosts are mastered by the same physical node, which usually results in a degradation of performance. However, the services keep running. Put another way, in the symmetric configuration, both nodes are active and each node is typically connected to its own database and its own configuration within the shared disk. When one node fails, the load of the failed node is assumed by the surviving node and the surviving node must then handle its own load in addition to the load of the failed node. Hence, the surviving node must be capable of handling the increased load for the time it takes to replace or repair the failed node. Of course, the administrator should repair the failed node as quickly as possible.

A symmetric configuration makes sense when resources are scarce or when large servers that cannot be dedicated as backups are involved.

In the N+1 configuration, multiple IM servers are configured to run as active nodes. Each node typically has its own configuration; there is no sharing of configuration data between these nodes although the disk itself is a shared resource. Essentially, then, each node mounts its own filesystem within the shared disk. In addition to the N active nodes, there is also a single idle node that serves as the backup node. If any of the active nodes fails, the backup node assumes the load of the failed node. Using HAStoragePlus, the failed node most likely unmounts the filesystem to which it is connected, and the standby node mounts the filesystem. Hence, the filesystem becomes a part of the resource group (that collection of resources that the cluster manages as a unit) and is mounted locally on the node that belongs to that resource group.

All Instant Messaging Server Sun Cluster configurations support both HAStorage and HAStoragePlus resource types for making filesystems highly available within a Sun Cluster environment. HAStoragePlus is the recommended resource type because it can work with any filesystem that the Solaris OS supports (UFS, VxFS, etc) and results in significant disk-I/O performance improvement over HAStorage.

To maintain the availability of the directory server, Java System Instant Messaging supports LDAP failover. LDAP failover works on a multi-master replication (MMR) setup of LDAP servers. In this configuration, all the LDAP servers are master servers and have permissions to read and write data. The Instant Messaging Server uses only one server at a time and fails over to the surviving LDAP server when the current LDAP server becomes unavailable.

Java System Instant Messaging also leverages the Java Enterprise System monitoring framework to actively monitor and manage instant messaging processes. It also uses log4j, the open source, extensible logging mechanism developed by Apache for creating log files that record various events, system errors, or other aspects of interest. In addition, using log4j, you can define your own logging configuration according to the log4j syntax, including the ability to configure Java System Instant Messaging to generate a separate log file for XMPP traffic only. Furthermore, there is an instant messaging watchdog process that detects potential problems and helps to maintain the availability of the instant messaging service.

Secure access, secure communication

Security plays a critical role in the day-to-day operations of today's enterprise. Corporations need to maintain at least the same level of security for instant messaging communication as they do for other forms of communication within the enterprise. A compromise in security can result in downtime and increased operations costs. Common use cases for secure instant messaging include businesses that require secure transmission as well as secure data archival, and service providers or extended enterprises that wish to establish user privileges based on a user's role.

Java System Instant Messaging provides secure communication at multiple levels — from policy enforcement to encrypted communication. The basic level of security is through LDAP namespace ownership, which means that instant messaging communication can be limited to those users provisioned in a given corporation's LDAP directory. This level of security is employed in isolated LDAP-based IM networks. The IM Server communicates to an LDAP directory server for both authentication — in the case of an LDAP-based, non-identity deployment — and user search. In order for chat to occur, the users must be in LDAP. The IM Server cannot communicate to a user outside its LDAP namespace unless the server has been configured to communicate with another server that owns a separate namespace. The fact that the IM Server owns the namespace ensures that enterprise users' communications will remain within a contained environment.

With traditional consumer IM solutions such as AOL AIM or Yahoo Messenger, IM communication is not secured against the corporate directory and is not contained within the enterprise, and as such, may be subject to eavesdropping or virus attack. This is an obvious consideration should the customer decide to connect the instant messaging server to a public IM network through an XMPP gateway. Of course, only the traffic that enters the public IM network is insecure. Communications contained within the Java System Instant Messaging domain remain secure.

Through the Java System Access Manager, Java System Instant Messaging can be used to provide communication to the extended enterprise beyond the corporate employees. The Java System Access Manager provides service-level and feature-level access control for Java System Instant Messaging. This enables the administrator to define roles as well as a corresponding access policy for those roles. For example, an administrator could define the access policy for customers, which is different from the access policy for partners or employees. Through the use of the Java System Access Manager, the user could disable the file transfer capability for customers yet retain this capability for partners. The policies that can be enforced using the Java System Access Manager include but are not limited to:

- The ability to send alerts
- Chat access
- Conference room access
- Conference room management
- Conference room moderation
- Contact list management
- File transfer
- News access
- News management
- Ability to send polls
- User settings access

It should be noted that in cases where Java System Instant Messaging is not integrated with the Java System Access Manager, access control is implemented through a set of text-based files that are stored on the IM Server and accessed by the server when required. Access controls are used for administration, users, news channels, and conference rooms. These access controls are implemented by the IM Server in an LDAP-based deployment or by the Java System Access Manager in an identity-based deployment.

Integration with the Java System Access Manager also enables Java System Instant Messaging to take advantage of the single sign-on capability, allowing the use of a variety of authentication methods such as LDAP, SecurID, Radius, Membership, and others. Required authentication ensures users are who they say they are.

Java System Instant Messaging also provides security through encryption. Users can deploy Java System Instant Messaging in conjunction with the Transport Layer Security (TLS) protocol to provide client-to-server and server-to-server encrypted communications as well as certificate-based authentication between servers. Secure server-to-server communications is supported in both LDAP-based and identity-based deployments. In addition, the IM Server itself supports a legacy implementation of SSL for encrypted communications between the client and multiplexor.

Furthermore, Java System Instant Messaging provides privacy profiles and contact list authorization mechanisms to ensure further security. Privacy profiles are user-defined lists that apply traffic and presence blocking or enabling rules to individuals or groups of users. Users can create profiles that specify who may or may not communicate with them or who may or may not view their presence or status. Contact list authorization provides a mechanism for end users to control who has access to see a given user's status. When a user attempts to add a contact to his or her contact list, an authorization request is sent to the target contact for approval. If the target contact approves the request, status or presence updates may be viewed by the initiating user. Contact list authorization is enabled by default, but the end user can specify that all status requests are automatically approved.

Secure document archiving

While some view IM communications as transient and not worth archiving, many enterprises seeking to use IM as a business tool view archiving as an essential requirement. For example, in cases where IM is used as the primary tool for providing customer service, the ability to archive the communications between the customer and the customer service representative is paramount. Solutions to customer problems can be archived for later retrieval by other customer service representatives solving similar problems. Furthermore, certain businesses must adhere to government or industry regulations and may require all communications to be archived.

Java System Instant Messaging provides a flexible archiving solution that consists of out-of-the-box archive providers such as the email archive provider, as well as a programmatic interface that enables developers to create custom archive providers.

Java System Instant Messaging provides an email archive provider that leverages the existing store and archive capabilities of traditional messaging systems as well as the search capabilities of existing email clients. With the email archive provider enabled, IM communications of participants will be issued as email messages to each user participating in the conversation, poll, or other form of communication. Using this technique, each participant can retain a transcript of the IM conversation, and virtually any email client can be used to search and manage instant messages. When used in conjunction with the Java System Messaging Server, the email archive provider can take advantage of the wide range of features offered by the Java System Messaging Server Message Store such as flexible rules for aging and expiring messages, and quota enforcement.

Finally, Java System Instant Messaging provides the APIs to create a custom archive provider. The Archive Provider API is useful for creating an archive provider that can be invoked for each of the following server processes:

- When an instant message is sent, such as an alert, poll, chat, news or conference message
- During an authentication event, such as a login or logout
- When there is a change in the presence status
- During a subscription event, such as when someone joins or leaves a conference or subscribes or unsubscribes to a news channel

Intuitive, easy-to-use, feature-rich clients

Java System Instant Messaging supports two native client interfaces out of the box: the standalone Instant Messenger client and the integrated Sun™ Convergence AJAX Web 2.0 client. The Instant Messenger is the standalone client component of Java System Instant Messaging. It supports two flavors of Java technology out-of-the-box. The Instant Messenger client can be run within a Java technology-enabled browser or within the Java Web Start application. Java Web Start has the benefit of automatically detecting when the client files are out of date and automatically downloading them from the Web container when the user attempts to log into the IM Server.

The Instant Messenger is a feature-rich client that delivers a high level of functionality:

- Provides real-time presence management that indicates who is online, offline, away, or idle, thus enabling users to describe their availability or location. The Instant Messenger is capable of displaying additional availability information beyond what is provided by presence, such as offline but forwarding alerts. The IM Server manages presence, which manifests itself within the client application as an icon of an individual who appears to be talking

- Provides a robust contact list for creating dynamic contact groups — as many as are needed for collaborating with different users and groups. Users can initiate chat with individuals in the contact list or with an entire contact group. Contact groups could represent project teams, working groups, or accounts
- Supports real-time alerts and polls to make users aware of time-critical messages or to conduct instant surveys. Alerts can be tied to individual messages or sent as one-way broadcasts to a group. Alert storage and forwarding enables users to specify routing of alerts received when they are offline. Alerts may be forwarded to wireless devices or pagers. The alert capability is extended to generate automatic calendar pop-up reminders when used in conjunction with the Java System Calendar Server
- Supports news channel creation and subscription, thereby enabling users to access published information, such as company announcements, product updates, or event notices, on a subscription basis. News channels can contain Web links and attached files, and are under control of the administrators; users can be automatically subscribed to news they need to view regularly
- Supports the use of conference rooms that are persistent, pre-established, private discussion rooms where multiple users can chat. Content from these rooms can be archived when a project is complete. Moderated conference rooms are also supported in which a moderator approves the content before it is posted into the discussion room
- Supports Jingle, the peer-to-peer voice chat capability, thereby enabling users to initiate a voice call over the internet by simply clicking the phone icon in the chat window. Since the call takes place over the internet between computers, both users should have headsets or the computers may have built-in microphones and speakers
- Provides user-level access control and the ability to create individual privacy profiles that enable users to exercise control over who can “see” and chat with them
- Supports single cast and broadcast of messages for sending to individuals or groups of contacts
- Provides collaborative file transfer capability. Files can be readily attached to chat discussions and sent over a secure channel
- Supports multi-format messages including text, rich text, URLs, and binary messages such as images. Users can control the message font size and color, or can send a variety of emoticons to vary the “mood” of the chat
- Interoperates with other XMPP-compliant servers
- Provides built-in proxy support enabling the Instant Messenger client to use an HTTP proxy, an HTTPS proxy, or a SOCKSV5 proxy. Built-in proxy support is useful for those organizations that require users to go through a proxy server before connecting client hosts to the Internet. The support for the SOCKS proxy is useful for firewall traversal
- Provides an autologin capability whereby a user can self-provision an instant messaging account for instant access to the instant messaging service
- Offers easy customization of the Instant Messenger client to modify the branding or logos

Sun™ Convergence

The standalone Instant Messenger client works well for customers who need a versatile instant messaging application that needs to exist as an individual component on their desktop. However, other users may want a more integrated client that combines presence and chat with other communications services such as e-mail, calendar, and address book. The Sun Convergence Web client provides a fully integrated fat client experience inside a browser. Drag and drop, drag and resize, auto-completion of addresses, context sensitive actions, customizable themes, and more are weaved into the fabric of Convergence making it a compelling alternative to standalone communications applications. Sun Convergence is deployed as a Web application within the Java System Application Server. Consequently, it can be deployed on a machine that is physically separate from the machine running the Instant Messaging Server, or it may be deployed on the same machine. Sun Convergence supports e-mail, calendar, address-book, chat, presence, and integration of 3rd party functionality to provide new integrated services (mashups). In terms of instant messaging functionality, Convergence delivers the following:

- Provides real-time presence management that indicates who is online, offline, away, or idle, thus enabling users to describe their availability or location. Convergence is capable of displaying additional availability information beyond what is provided by presence, such as On the Phone or Out to Lunch
- Provides integrated presence throughout the user interface. If an e-mail sender or recipient is an authorized contact in the user's buddy list, the user can instantly view the presence of the e-mail sender or recipient in the e-mail header. Calendar users can instantly see the presence of intended invitees
- Provides a robust contact list for creating dynamic contact groups — as many as are needed for collaborating with different users and groups. Contact groups could represent project teams, working groups, or accounts. Users can initiate chat with individuals in the contact list or group. Users can also upload Avatars or images so that buddies can immediately see the user's avatar when the user comes online or during a chat
- Supports multi-format messages including text, rich text, and URLs. Users can control the message font size and color, or can send a variety of emoticons to vary the "mood" of the chat
- Supports presence authorization so that contacts can only see a user's presence if the user has given approval. Approvals can be given automatically or manually depending on the user's preference
- Supports chat transcripts. Users can print chat transcripts or email chat transcripts to themselves or others. The date and time of the transcript is automatically inserted into the e-mail subject line

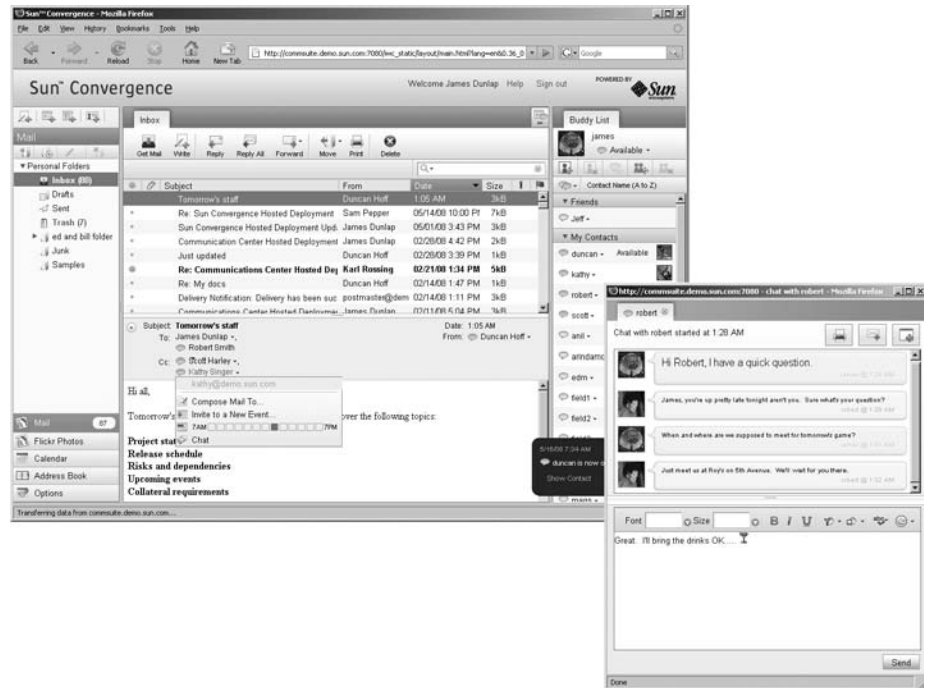


Figure 8. Sun Convergence chart and presence integration

The Sun Convergence UI is highly customizable. It is based on AJAX and uses the open source Dojo JavaScript toolkit. As a result, the instant messaging client can be customized to suit the needs of the deployment or the domain. Customization examples include customizing the skin or theme, the login screen, the icons used in the instant messaging component, the text within Sun Convergence, the application banner and application bar, the branding, the toolbars, the instant messaging pop-ups, and so on. Customization also includes the integration of 3rd party applications into the user interface not only to enable a new service but to integrate with the existing services as well. To the end user, the application appears in the UI as another component, just like mail or calendar.

Presence-enabled applications

First and foremost to instant messaging is the concept of presence. It is presence, after all, that enables individuals to know who is available right now to engage in an instant messaging conversation. However, instant messaging is not the only application that can take advantage of presence. Consider the utilization of presence in certain services such as Web services. Processes can be engineered to use presence as an indicator to automatically use these services when they become available. Java System Instant Messaging provides visual presence indicators and enables individuals to modify their presence provided they have the proper access controls. Individuals can even limit the ability for others to see their status. Perhaps more importantly, Java System Instant Messaging provides an open API for accessing the internal presence service for use in other applications.

Open extensible platform

Java System Instant Messaging provides a highly extensible and customizable framework. It offers extensibility to additional clients through a Java developer API, and also offers extensibility to core identity services through a set of Java identity APIs. Through the Java APIs, the developer will have access to the policy configuration service, logging service, session service, authentication service, and more.

Public Java APIs

In addition to providing a modular, highly flexible architecture, Java System Instant Messaging provides the following application programming interfaces (APIs) to extend the functionality of the product:

Instant Messaging Service API

The Instant Messaging Service API is a developer API used by applications to access internal instant messaging services such as the presence service, notification service, news service, conference service, and polls service. With the Instant Messaging Service API, developers can integrate instant messaging services into Java-based or Web-based clients. Other uses may include the development of bridges or gateways that enable another class of clients, or the integration of instant messaging and presence into existing applications.

Messenger Beans

A Messenger Bean is a dynamically loaded module used to extend Instant Messenger functionality. Messenger Beans can add action listeners, such as buttons and menu items, and item listeners, such as check boxes and toggle buttons, in the existing Instant Messenger window.

Archive Provider API

The Archive Provider API is used to provide IM message archiving. An Archive Provider is a software module that provides integration with some type of archive or auditing system. Each configured Archive Provider is invoked each time the server processes a message (alert, poll, chat, news, or conference), an authentication event (login or logout), a presence status change, or a subscription event (someone joins or leaves a conference or subscribes or unsubscribes to a news channel).

Message Conversion API

The Message Conversion API is used to perform virus checking and removal, translation engine integration, or message content filtering. This API invokes a message converter for every message or each message part going through the server. The message converter may leave the message part intact or may modify or remove the message part. Messages could potentially include text as well as attachments. The text parts are processed as Java string objects. The message converter processes the other attachment as a stream of bytes and returns a potentially different stream of bytes — or nothing at all if the attachment is to be removed.

Authentication Provider API

The Authentication Provider API provides the ability to deploy Java System Instant Messaging in environments that are not using the Java System Access Manager password-based or token-based authentication service. Hence, this API can be used to integrate Java System Instant Messaging with other authentication systems. A few examples of how additional applications benefit from using these APIs include:

- Portal channels can display contact lists and allow real-time communication
- Presence status can be displayed and conferences initiated from other Web-based collaboration applications such as Web-based e-mail, calendar, or address books
- Desktop-based instant messaging clients and mobile instant messaging clients can deploy Java System Instant Messaging
- Server applications can display presence status in order to notify end users optimally
- News feeds can be displayed as Instant Messenger news

User interface customization

Java System Instant Messaging not only offers a flexible development environment for extending services and clients, it also offers the ability to customize the client user interface. The Instant Messenger client user interface runs within a Java application or applet, but certain aspects of the user interface can be customized, such as logos, images, audio sounds emitted, the title, and any branding associated with the product. This enables service providers and enterprises to specify their own unique branding for the product. Typical examples of customization might include changing the title of the application and the name of the vendor, changing the icons used in the user interface, changing certain text in the user interface, changing the background, or changing the launch page that launches the Instant Messenger client.

Chapter 3

Conclusion

Java System Instant Messaging is a compelling solution for any business seeking to provide instant messaging services to its employees, partners, or customers. It has several key features that make it an exciting component of the Java Communications Suite.

High-performance server

Java System Instant Messaging is built on an extensible, distributed, modular architecture. This underlying architecture employs connection multiplexors, which are used to provide a certain level of horizontal scalability to the environment. The multiplexors scale vertically with the introduction of additional CPUs. The IM Server component scales with the introduction of additional memory. The distributed nature of the architecture enhances the overall scalability of the instant messaging solution, since various IM components and services can be distributed throughout the network and run on dedicated hardware if necessary. Furthermore, to provide vast scalability, IM servers may be pooled together to support an ever-growing number of end users.

Java System Instant Messaging employs a watchdog process and leverages the Java Enterprise System monitoring framework for system monitoring and managing capabilities.

Security, flexibility, and extensibility

Java System Instant Messaging supports secure transactions over TLS and SSL and provides a robust and flexible authentication framework for developing pluggable authentication modules (PAMs). It can be deployed with the Java System Messaging Server in order to utilize its built-in archive capabilities. It can also be deployed on multiple operating system platforms. The standalone instant messaging client can be deployed as an applet or a full-blown Java application. Java System Instant Messaging also supports the use of an integrated AJAX communications client, Sun Convergence. The individual components can be run on the same physical hardware or they can be distributed throughout the network.

Java System Instant Messaging provides support for multiple clients through a Java developer API that enables additional IM clients to access core IM services. Additional clients can also access IM services through XMPP interoperability and the built-in HTTP proxy gateway. Finally, the user interface is customizable, enabling administrators or developers to add their unique branding to their deployment.

Based on standards and proven technology

Java System Instant Messaging is built on standards such as Java technology, XMPP, HTTP, LDAP, IMPS, TCP/IP, TLS and SSL. As a result, it can be easily integrated into environments that employ these standards. Java System Instant Messaging integrates seamlessly with the proven, industry-leading, highly scalable Java System Directory Server, Java System Access Manager, Java System Messaging Server, and Java System Calendar Server. The integration with other products produces an overall communications solution that is greater than the sum of its parts.

Integration with Java System Access Manager provides a unique policy and roles-based access control solution that can be used to effectively control instant messaging policy for employees, partners, and customers. Integration with the Java System Calendar Server leverages the alert capability of instant messaging to provide pop-up notifications for calendar appointments. Integration with the Java System Messaging Server produces a flexible archiving solution that can be integrated with third party message archiving systems such as the AXS-One Compliance Platform. Because Java System Instant Messaging relies on LDAP to perform user search, users can be easily identified for inclusion into contact or buddy lists without having to know the user's handle or user ID.

In summary, when Java System Instant Messaging is used as a standalone product, it is an effective instant messaging application. However, Java System Instant Messaging is much more than an instant messaging application. Its extensibility and integratability with other components of the Java Communications Suite and Java Enterprise System identify it as a platform capable of solving complex problems introduced by an ever changing communications landscape.

Chapter 4

Appendix A

References

The following Web sites provide additional information about the product and relevant technologies:

- *Sun products, tools, and technologies documentation* — docs.sun.com
- *Java System Instant Messaging product documentation* — <http://wikis.sun.com/display/CommSuite>
- *Java System Instant Messaging* — sun.com/software/products/instant_messaging/index.xml
- *Java System Messaging Server* — sun.com/software/products/messaging_srvr/index.xml
- *Java System Calendar Server* — sun.com/software/products/calendar_srvr/index.xml
- *Java System Access Manager* — sun.com/software/products/access_mgr/index.xml

Abbreviation/acronym list

AIM	AOL Instant Messaging
AJAX	Asynchronous JavaScript and XML
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
ICAL	Internet Calendaring (also iCalendar)
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMAP	Internet Messaging Access Protocol
IMPS	Instant Messaging and Presence Service
IP	Internet Protocol
JRE	Java Runtime Environment
JVM	Java Virtual Machine

LDAP	Lightweight Directory Access Protocol
OMA	Open Mobile Alliance
PAB	Personal Address Book
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
SIP	Session Initiation Protocol
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SMPP	Short Message Peer-to-Peer
SMS	Short Message Service
SMSC	Short Message Service Center
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDDI	Universal Description, Discovery, and Integration
UDP	User Datagram Protocol
WAP	Wireless Application Protocol
WCAP	Web Calendar Access Protocol
WML	Wireless Markup Language
XEP	XMPP Extension Protocols
XML	eXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol

This Page Intentionally Left Blank

