Sametime

Version 9.0

Sametime 9.0
Software Development Kit

Introduction to the Community Architecture

IBM

Edition Notice

Note: Before using this information and the product it supports, read the information in "Notices."

This edition applies to version 9.0 of IBM Lotus Sametime (program number 5725-M36) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1 About This Guide

Intended Audience

This white paper is for developers who need to familiarize themselves with the IBM® Lotus® Sametime® environment before starting to create applications. It assumes no in-depth experience with IBM Lotus Sametime, but reasonable familiarity with software design and development in a network context.

Chapter 2 Introduction

The Sametime platform is a client/server environment in which the client is connected via TCP/IP to the server. The Sametime server supplies its users with awareness and real-time interaction services.

This document describes the basics of the Sametime community architecture. It provides an overview of the Sametime client/server topology and discusses Sametime communication mechanisms, how users are supported, and basic services and applications.

Chapter 3   Sametime Community Structure

A Sametime community consists of:

- Clients, which are connected via TCP/IP

- Multiplexers, which improve Sametime scalability by I/O concentration

- One or more community hubs, which log in Sametime clients, route messages between members, and notify subscribers of events in the community

- Server applications, which are connected to community hubs via TCP/IP

Clients, multiplexers, community hubs, and server applications are all Sametime community members.

Every community member can log in to the community; community hubs log in to each other, while clients, multiplexers, and server applications log in to the community hubs. When community members are logged in to the community, they are considered 'community participants'.

Basic Structure and Components

Below is a schematic diagram of a Sametime community:

C – Client (User), M – Multiplexer, C-HUB – Community Hub, ←→ - TCP/IP connection
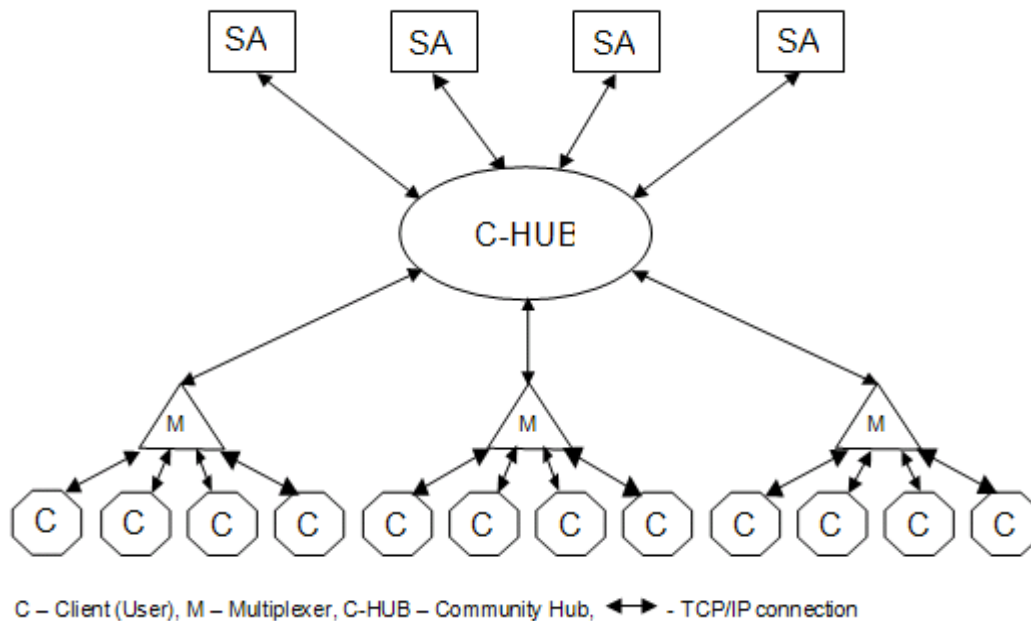
Client

The client is a program that logs into the community as a user. It displays the user interface (Windows, other operating system, or browser) and can perform some or all of the application processing. The Sametime user and login model supports multiple users running multiple applications concurrently.

Multiplexers

Multiplexers support Sametime community hubs with:

- **I/O concentration** – A multiplexer collects and distributes I/O data for multiple clients communicating with a single hub. This concentration limits the I/O overhead of the hub by lowering the number of connections it uses to send and receive data. The overhead of connection initialization, polling, and termination is greatly reduced.

- **Distribution of multi-recipient messages** – The community hub can send a single message (with a recipient list attached) to its multiplexer. The multiplexer then distributes the message to the indicated recipients.

- **Gateway** – A multiplexer can act as a gateway, translating protocols between clients or between third-party server software and the Sametime community server.

The multiplexer layer is transparent to the clients that connect to the community. Multiplexers can be connected in a multi-level structure, in which multiplexers connect to each other in a chain, rather than directly to the community hub. Chaining can be used to localize user connections or in the firewall environment, where the multiplexer acts as a proxy server.

Community Hubs

Community hubs, which are the core of the Sametime community, are hosted on Sametime servers that range in capacity from high-end PCs to mainframes. They act like switchboards between community participants. Hubs are responsible for the following:

- Managing community participants (clients, multiplexers, and server applications)

- Routing messages between community participants

- Supplying notifications to community participants

Community hubs manage the logins for each user and maintain order and security for Sametime clients.

Community Server Applications

Server applications enable the functionality provided by the hub by supplying additional services to the community. A server application connects to a hub and declares the services it supplies. The hub routes requests for these services to the appropriate instance of the server application.

Applications included with the Sametime server supply the following services:

- **Configuration** – Reads configuration data and distributes it to subscribing community participants.

- **Awareness (Buddy List)** – Notifies users about status and properties of other users in the community. The awareness service respects the privacy requested by users.

- **Authentication** – Authenticates users that ask to log in to the community. The authentication is performed against the community directory, using several authentication mechanisms.

- **Directory** – This category includes the following services:

  - **Resolve** – Resolves a given user name to a list of matching user IDs

  - **Group contents** – Handles requests for the content of a public group

  - **Browse & Search** – Browses or searches the user directory in order to find a certain user.

- **User Storage** – Responsible for storing user related data on the server, including the privacy and general attributes users set for themselves.

- **Logging** – Responsible for logging activity of and within the community server, including general logging of the community server activities and logging of chat between users.

- **Places** – Supplies place-based awareness and communication tools.

- **Online directory** – The "glue" of the community server. The online directory is responsible for creating a consistent view of the community. For example, the online directory knows the community hub at which each user is logged in. The Online Directory service uses a replication algorithm to keep the global view of the community.

- **Web client support** – Supports clients that use HTTP protocol to connect to the community hub. These services enable light protocol and the polling manner of connection required when using HTTP protocol.

- **External communities** – Enables awareness and IM services with other communities. Sametime uses the IETF SIP/SIMPLE protocol as the protocol for inter-community interactions. Note that these services are used also for negotiating audio/video sessions using the same IETF protocols.

Note: Instant Messaging (IM) is supplied directly by the Sametime community hub.

Third-party vendors can use the Sametime Community Server Toolkit to develop other services.

Sametime Communication

This section describes the communication mechanisms in a Sametime community.

Addressing Modes

There are several ways to address other community participants:

- User ID – Some user is addressed. If the recipient user has more than a single login, one of its logins is selected.

- Login ID – A particular login is addressed. No login selection is needed.

- Service Type – A server application providing the specified service type is addressed. There may be multiple instances of the application that provides this service. The community hub must decide (possibly by consulting the available providers) which provider will fulfill the request.

Addressing Scopes

The most frequent type of interaction in a community is one to one, in which two community participants exchange messages. There are also multi-recipient interactions, which are candidates for performance optimization. Sametime communication optimizes them as follows:

- One to many – The message is intended for multiple recipients. The common case is an N-way chat, in which a message sent by one member in the chat must reach all chat participants. Only a single instance of the message is sent from the source. The message contains a list of recipients. The community hub distributes a single copy of the message to the multiplexers involved, while the multiplexers distribute the message to the appropriate logins. Note that in an N-way chat, encryption may render this optimization unusable, since each chat participant may have a different encryption key. For more information, see "Security."

- Broadcasts – The message is broadcast to all community user participants. The community hub sends a single copy of the broadcast message to each multiplexer connected to it; each multiplexer dispatches the message

to its logins. In a multi-hub community, the message is sent to all community hubs; each community hub relays the message to its multiplexers. For more information about multi-hub communities, see "Leaving aside the issues of distribution and scalability, dependency on a single hub is still a problem. Users should be able to connect to the community even when servers crash or are in maintenance. ."

TCP/IP

There is a persistent TCP/IP connection between clients and multiplexer, between multiplexers and hub, between server applications and hub, and between all hubs in the community.

NoteoteUsers connecting via HTTP protocol create a new connection every X seconds, depending on their configuration.

Channels

Message routing typically relies on virtual connections called channels. A channel may span several TCP/IP connections. For example, when a client on multiplexer A creates a channel to a client on multiplexer B, the channel traverses the following TCP/IP connections:

- From the user login to multiplexer A

- From multiplexer A to the hub

- From the hub to multiplexer B

- From multiplexer B to the login of the other user

The following diagram shows the path of the channel created between two clients:
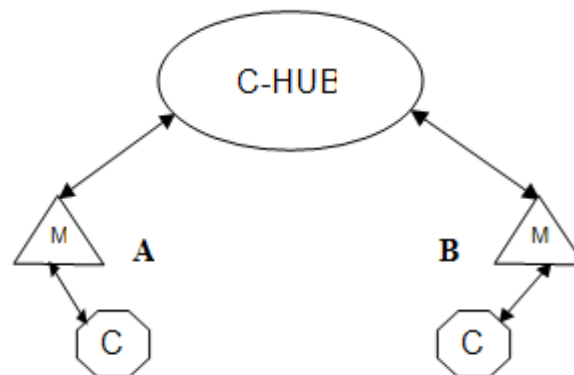
As shown, the network route between community members may involve several hops. The channel enforces the order of messages in the interaction and ensures connectivity by providing a notification when the route of the channel is broken (for example, when a crash occurs along the channel route). Each side of the breakage is

responsible for notifying all community participants along the channel route, including users, hubs, and multiplexers, that the channel has been disconnected. Data that enables routing of channel messages is stored in each community participant on the channel route.

Channels can also be redirected. A channel may be created to a community participant, and then transferred to another community participant, while the initiator of the channel remains unaware of the redirection. Redirection plays an important role in a distributed environment, as when a channel is created to a server application, and the application redirects the channel to another server application to help with load balancing.

Master Channel

The master channel is created between a community participant and its serving community hub when the participant logs in. The master channel is the route taken by all the subsequent communication between the client and its community hub. When a new channel is created (for example, from a client to a server application) it traverses the route of the master channel up to the community hub; from there it traverses a different route until it reaches the destination.

Distribution, Scalability and Redundancy

A single Sametime community hub is designed to serve about 100,000 simultaneous user logins. This capacity is adequate for many communities. However, a single hub cannot address some situations.

Distribution

Community users may be distributed over geographically remote sites. In a single-hub environment, remote clients depend on the network quality between their site and the hub. This means that if the network path to the hub is temporarily down, remote users cannot interact with even nearby users. In addition, network use is not optimized; a separate one-to-many or broadcast message is sent to each client, rather than a single message being sent to the remote multiplexer, which then takes care locally of distribution to clients. Each separate message also travels from the remote multiplexer over several hops until it reaches its destination. More commonly, IMs between two neighboring users connected via remote hub must travel to the remote hub and back, instead of traveling to a common multiplexer.

A multiplexer deployed remotely addresses the connectivity of users and the issue of sending multiple messages remotely, but it does not keep IMs between two local users from having to go the remote hub and back. (Chat logging and privacy checks still need to be performed on the hub before the message is sent to its recipient.) A remote multiplexer also does not address the issue of service interruption when the remote site is down.

Scalability

While the capacity of a Sametime community hub is enough for most organizations, it does not match expectations in very large organizations (for example, IBM) or in the services market (for example, Amazon). Sametime must provide a solution for serving millions of users in a single community.

Redundancy

Leaving aside the issues of distribution and scalability, dependency on a single hub is still a problem. Users should be able to connect to the community even when servers crash or are in maintenance.

Multi-Hub Solution

Below is a schematic view of a multi-hub Sametime community. Note that the only schematic change is that instead of a single community hub, several hubs connected to each other constitute the community core.

C-HUB   Community Hub, M   Multiplexer, SA   Server Application, C   Client
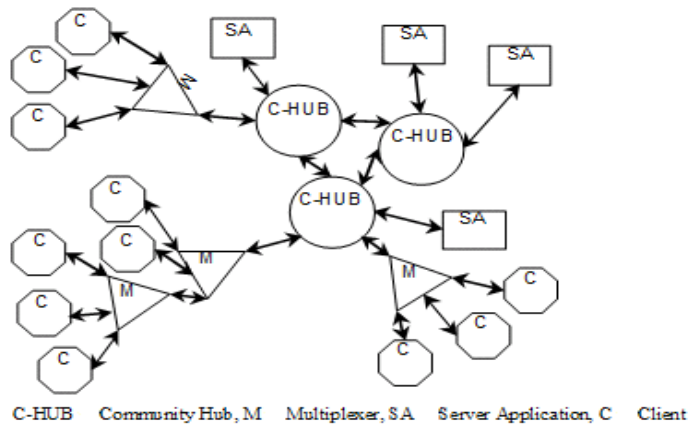
Every Sametime hub supplies all services. Hubs can be located at remote sites and are interconnected. Note that it is possible to have global services in Sametime that are not necessarily installed with each hub.

Home Hub

A user's properties are kept in persistent storage associated with a hub. In some scenarios (see "Scalability and Redundancy (Clustering)"), this storage is replicated between hubs. However, in a geographically distributed environment, it is not possible to replicate this data in real time, so a user who has changed his properties on one hub and then logs into a different hub could receive outdated data or (if replication does not occur) no data at all.

To solve this problem, a 'home' community hub is defined for each user in the community, by attaching the address of a hub to each user. The home hub contains the directory database and/or user information (privacy and other general attributes).

The client program can get the home hub from its local storage, or simply try to connect to some hub in the community. If the connection is established to a hub other than the user's defined home hub, the connection is internally redirected, using the channel mechanism, to the user's home hub.

Note: You cannot be logged in to two community hubs at once. If you try, you will be disconnected from the both hubs.

Special considerations

Connecting to a multi-hub community poses no special problem for users. However, various tasks, such as message routing and community participant addressing, require special handling in a multi-hub community. Sametime community server components support these tasks in a multi-hub environment by default, through the Online Directory server application (see "Community Participant Addressing," below).

Routing

Depending on the number of community hubs, a channel or a message destined for another community participant might need to traverse several hubs, and an efficient route would have to be chosen. Sametime avoids this problem by connecting every hub in the community to every other hub. With this type of connection, known as a clique, messages must traverse at most two hubs:

- the hub to which the first community participant is connected

- the hub to which the second community participant is connected.

A virtual channel is established and all messages follow this route. Clique connectivity of community hubs allows a simpler algorithm for selecting the channel route.

Community Participant Addressing

In a single-hub community, when a participant is addressed, the hub routes the message to the appropriate community participant as it appears in its local tables. In a multi-hub environment, that is not possible.

There are various ways to address this problem. The Sametime solution is to use a dedicated server application called the Online Directory (OLD), which has global knowledge of the entire community. The OLD gathers and maintains minimal location data, enough to find online entities (such as user logins) anywhere in the community.

Note: It is important to understand that OLD data gives only enough information to locate an entity. Subsequent queries to other community participants are needed to gather other information needed.

The default (and only) implementation of Sametime is multi-hub enabled. The Online Directory is always installed and there is no way to install a community without an OLD.

Online Directory Usage

When a hub has to locate a community participant by user ID, or locate a place, it simply consults its OLD.

When a community participant is addressed by login ID, the services of the OLD are not needed. Since a login ID contains the IP address of the server on which the login resides, the hub simply extracts the IP address and contacts the indicated server.

When a server application is addressed by service type, the community hub is responsible for locating the correct server application instance. The OLD is not consulted for server applications.

Scalability and Redundancy (Clustering)

A multi-hub community as described above can solve the distribution issue, but not the scalability and redundancy issues. To address these issues, we introduce the notion of clustering. A cluster is a set of community hubs in which every hub can act as the home hub of all users in the cluster. A home cluster is defined for the users in the cluster, replacing the users' home hubs.

All hubs in a cluster reside in the LAN, with high bandwidth network connectivity. Replication of user properties in a cluster occurs (almost) in real-time.

Clusters are named. A single hub name can be translated, using DNS or some other mechanism, to a list of the multiplexers serving the hub. A cluster name is translated to a list of the multiplexers serving the hubs in the cluster.

Load balancing in a cluster is performed via this translation mechanism. Each translation yields a different order of the cluster multiplexers, in a DNS round robin. The client tries to connect to them in the order given.

A cluster can also be connected to an IP sprayer, such as the IBM eND (e Network Dispatcher). The IP sprayer can choose one of the multiplexers to connect the user.

Connecting communities

Often, community users would like to get services from different communities. For example, a user in the Mars community would like to know when his friend/colleague from Venus community is online. Sametime enables this kind of inter-community service sharing.

Sametime uses the IETF protocols - Session Initiation Protocol (SIP) and SIP Instant Messaging and Presence Leveraging Extensions (SIMPLE) - to connect communities. The SIP Gateway server application acts as a SIP proxy to the external world, and enables awareness of and two-way IM exchanges with users in other Sametime communities. It also supports meetings between users in different communities and handles privacy and secure transport.

The following drawing illustrates the architecture of the SIP solution. It shows both the physical and the logical (channel) connections between the different components.
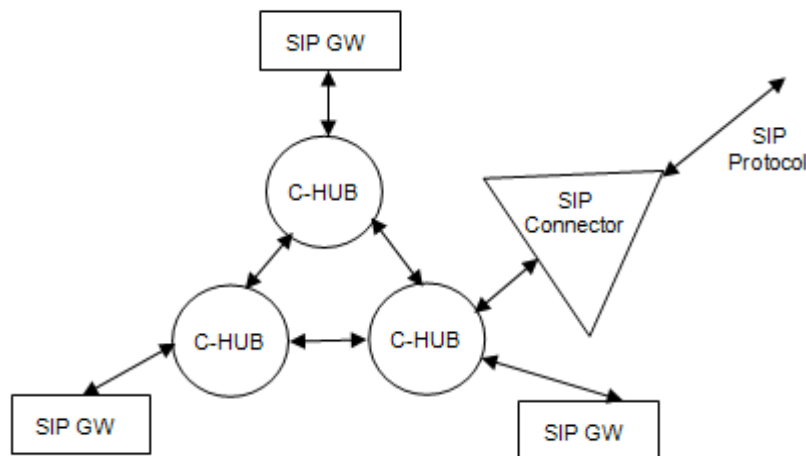
There is one instance of the SIP Gateway for each hub in the community. Each SIP Gateway serves the clients connected to its hub.

In addition, there are a number of SIP Connector processes (actually server applications), each connected to one of the hubs in the community. Each SIP Connector is responsible for one or more external communities. It is responsible for creating the connections to the external domains and receiving connections from them. It is also responsible for parsing received SIP messages and constructing outgoing messages. The SIP Gateway uses a proprietary protocol to exchange messages with the connector.

Chapter 4   The Sametime User and Login Model

The Sametime user and login model supports multiple users running multiple applications concurrently. The model is divided into persistent and runtime parts.

Persistent User Data

Each user in the Sametime community has the following basic properties:

- **User ID** – A string representing the persistent and unique identifier of a user in the community (for example, a@example.com or "UN=Smith/ON=IBM").

- **Login parameters** – A set of fields used to create a login into the community. Common fields will be a login name and a password. The login name is a string that has no further usage beyond the login phase.

- **Display Name** – A name by which other users know the user, such as 'John Smith'. A user might have different user names for different logins. (See "Runtime User Structure" below.)

- **Description** – Descriptive text about the user. The description may be different for different logins.

- **Privacy List** – A list of user IDs that may or may not know that the user is online. A privacy list is maintained for each user. If a user logs into the community several times simultaneously, the hub synchronizes the server-hosted privacy list of the user between his different logins.

- **User attributes** – General list of user attributes.

The above model allows a user to have multiple login names and still be known as the same user by other community members, regardless of the login names she/he uses. This mechanism is called *aliasing*.

Runtime User Structure

The diagram below shows the runtime structure of a user.
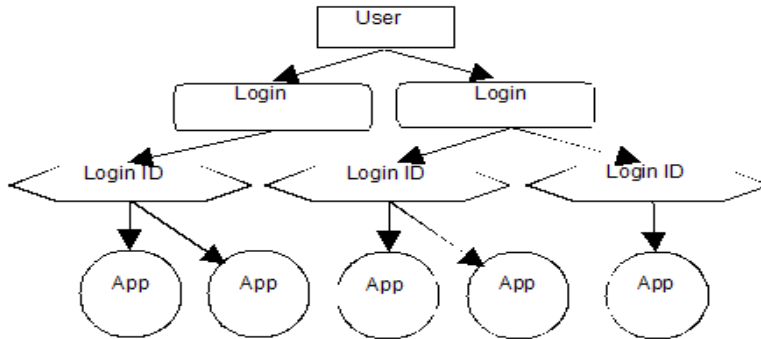
Each TCP/IP connection by a user is regarded as a single login of the user to the community. A login ID that is unique in the entire community is assigned to each connection.

All applications accessed by a user can use the same login, except when running non-cooperative applications (for example, Java and OCX components). Currently, the default behavior of a Sametime community is to limit user logins to a single machine (a single IP address). This ensures a good user experience, since a user who logs in at successive machines can be sure that only the current connection is open, and that IMs will therefore always reach him at his current machine.

Anonymous (Guest)

A guest is a non-authenticated user of the community. A community administrator may choose the services of the community that are available for guests. A guest does not have a persistent user ID (that is, there is no user ID associated with the guest in the user directory), and only the login ID can be used to address the guest. Furthermore, there is no way to associate different logins of the same guest.

Chapter 5 Server Applications

Server applications enable the functionality supplied by the community server. A server application is a container of one or more services. A service can be contained in different server applications according to logic or development considerations.

The server application connects to the community hub and declares the services it supplies. The hub routes requests for such services to an appropriate instance of the server application.

A service can be local or global. A local service can be supplied to the users of a community hub only by a server application that is directly connected to that hub. In contrast, a global service can be supplied to the users of a community hub by any server application in the community.

Sametime includes the following server applications. Each application supplies one or more services:

- Community Hub

- Configuration

- Admin Service

- ST Admin Act

- Directory

- Resolve

- Users

- Privacy

- User Storage

- Online Directory

- Buddy List

- Places

- Chat Logging

- Logger

- Polling Proxy

- Sametime Links

- SIP Gateway

- SIP Connector

Community Hub

The Community Hub supplies some of its functionality through the following services:

- **COMMUNITY** – A local service that provides community events on the status of objects (users, logins, and server components) in the system (for example, user login/logout, user status, user privacy list, and so on).

- **LOGIN_COP** – A global service that verifies that each user is logged in to only a single community hub at all times. If, for any reason – such as a temporary disconnection between hubs -- two logins of the same user are connected to different community hubs, it is the responsibility of the login cop to discard both logins.

- **STORAGE_EVENT_OTM** – A global service that is a One Time Message (OTM) sent from the hub to the logins of a user, notifying them that there is a change in the user storage.

Configuration

The Configuration server application periodically reads the server configuration from the Sametime configuration database and sends the data to subscribed server components. The configuration service in this application is one of the components that push community-wide attributes to the awareness service to be delivered to subscribers.

Admin Service

The Admin Service server application gathers status information from various server components and can provide the information to requestors. The Admin service sends the information it gathers to the logger, as specified in the Sametime server configuration. The Admin service also creates files needed for the operation of the Sametime Links server application.

ST Admin Act

The ST Admin Act server application is actually a CGI binary file that is able to get information from the Admin service upon request. It is used for creating Web pages with information about the status of the community server components.

Directory

The Directory server application provides Sametime directory browsing capabilities. It also allows users to receive the content of a public group.

Resolve

The Resolve server application supplies the Resolve service that resolves a user name into a unique user ID.

Users

The Users server application logs in users via password or token. It also creates tokens for users upon request.

Privacy

The Privacy server application is responsible for providing a storage service that lets users save their privacy information on the server. As with the User Storage server application, this frees users from dependency on the machines they are currently using.

Online Directory

The Online Directory server application holds a status snapshot of the entire community (who is online, on which community hub, and which places are active on each hub). It also detects user attempts to log in to two hubs at the same time and places that become active on two hubs at the same time.

Given its community-wide view, the Online Directory application is also responsible for:

- Allocating a hub on which to create a resource

- Helping the Buddy List server application locate a user in the community either immediately or when s/he goes online

- Notifying the Community server that a user is online on two different hubs simultaneously

User Storage

The User Storage server application provides a storage service that lets users save preferences, attributes, and other information on the server. This frees users from dependency on the machines they are currently using.

Buddy List

The Buddy List server application provides the "Who Is Online" service. This component tracks user login status and maintains user attributes. Clients send their presence or awareness list to this component and are notified of status and attribute changes. This component takes into account privacy information, status, and who is interested in this user.

The Buddy List server application is also responsible for maintaining and distributing values of community attributes that control the behavior of the client (for example, services that are enabled or disabled in the community hub).

The Buddy List Server application is capable of supporting awareness of general objects (for example, stock quotes). Currently, it provides only awareness of users and server components.

Buddy List distribution in a community is a key feature in the operation of the Buddy List server application.

Places

The Places server application provides the basic services related to N-way interactions. A single place is the holder of the interaction and contains participants and activities. All participants in a place are typically connected to each activity in the place. (A user's connection to each activity depends on whether the application s/he is running supports the features required by the activity, such as audio or video.)

The Places server application provides awareness of its participants, a basic interaction mechanism, and an interface to external activities. This interface enables the place to be a single synchronization point for multiple activities.

Chat Logging

The Chat Logging server application catches all the chat messages on the community hub, stores and retrieve them via an API.

When chat logging is enabled, the Chat Logging server application enters all places as an additional activity that listens to the chat in the place. In the case of Instant Messages (IM), the Chat Logging server application "listens" to the IM channels between the two parties. The server application sends the chat to be logged by the Chat Logging API.

Logger

The Logger server application is responsible for logging various statistic events to a persistent device using the Logging API. The Logger server application does not supply a service but uses the Community service of the Community server to log the events of the community.

Polling Proxy

The Polling Proxy server application serves logins that connect to the Community server via HTTP polling. The Polling Proxy server application receives client HTTP requests from the multiplexer and responds to them. The application also handles authentication between consecutive polling requests, buffering of messages, and other mechanisms required to support HTTP polling.

Sametime Links

The Sametime Links server application serves logins that connect to the Community server via a specialized light protocol that uses hybrid HTTP polling. The Sametime Links server application receives client HTTP requests from the multiplexer and responds to them. The Sametime Links server application also handles authentication between consecutive polling requests, buffering of messages, and other mechanisms required to support HTTP polling. The difference between the Sametime Links server application and the Polling Proxy server application is that the Sametime Links server application uses a reduced, lighter protocol, while the Polling Proxy server application uses the full Community protocol.

SIP Gateway

The SIP Gateway server application bridges between the Sametime protocol and the SIP protocol. The SIP Gateway is responsible for representing Sametime users to other communities via the SIP/SIMPLE protocol and for representing external users to the Sametime community via the Sametime protocol.

SIP Connector

The SIP Connector server application connects to an external community that uses the SIP protocol. The SIP connector is a global service (that is, one instance may serve the users of several community hubs). Different SIP connectors can be configured for different external communities. In addition, different SIP connectors may reside in different external networks of the organization, enabling firewall control.

Chapter 6   Service Provider Interfaces

Communication between server applications and databases or directories is based upon Service Provider Interfaces (SPIs). Sametime supports SPIs for Lotus Notes databases or directories and for LDAP directories. Interfaces to other directories (for, example Oracle), can be implemented by third parties or by customers.

Sametime includes the following SPI implementations:

Directory SPIs:

      Authentication, Resolve, Browse and Groups (for Notes and LDAP databases)

Database SPIs:

      Logging, Privacy, User storage and configuration (for Notes and DB2 databases)

Chapter 7   Security

The Sametime server supports community security through authentication of community members and encryption of sensitive information.

Authentication

Authentication is the process of verifying that a user or a server component is who it claims to be. There are many ways to perform authentication. The Sametime community server uses several.

User Authentication

User authentication is performed by the Users server application, which supplies the authentication service. This server application works with the user directory, rather than having the community hub access the directory directly, thereby speeding data transfer throughout the community. The Users server application uses an SPI to interact with the directory in order to perform the authentication. The default SPIs provided with Sametime support LDAP and Notes directories.

Currently, the authentication of users is performed via a user name and password or a token mechanism. The default implementation of this SPI supports both LTPA and Sametime tokens.

The installed Sametime Connect client prompts the user for name and password. Applets, including Java Connect and the MRC, receive an authentication token as an applet parameter, which is then used to authenticate the connection to the Sametime community server. Note that this approach assumes that the web page containing each applet is downloaded to the client's browser via HTTPS, so that the authentication token is properly protected. Domino and WebSphere contain settings that ensure this.

It is important to note that the authentication token is treated by Sametime as an opaque string. It is set by the Web application that pushes an applet to the client. It is then passed back to the server and handed to the authentication SPI as described above. No assumptions are made about the format of the token. Sametime uses LTPA tokens by default, but it is possible to use a different type of token by replacing both the Web application code that sets the token and the authentication SPI that checks it.

Note that if the Sametime community server is configured to allow anonymous users, a user may log in to the community without identifying him/herself.

Server Authentication

In many cases, one server component needs to connect to another. Examples include hub-to-hub, multiplexer to hub, and server application to hub connections. These connections are authenticated by checking the IP address from which the connection comes and ensuring that the IP is listed in the configuration as an IP from which the specific server component should connect.

Encryption

User connections are authenticated using either username/password or an authentication token. Either way, the client and server use Diffie-Hellman to agree on a secret key. That key is then used to encrypt the authentication information that is sent to the server.

Note: In N-way chats, the community hub distributes a single copy of each message to the multiplexers involved, while the multiplexers distribute the message to the appropriate logins. Note that encryption may render this optimization unusable, since each chat participant may have a different encryption key.

Currently, Sametime encrypts the following information:

- Authentication information

- Chats including multi participant conferences.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
5 Technology Park Drive
Westford Technology Park
Westford, MA 01886

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp.
Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

These terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM

AIX

DB2

DB2 Universal Database Domino

Domino

Domino Designer

Domino Directory

i5/OS

iSeries

Lotus

Notes

OS/400

Sametime

System i

WebSphere

AOL is a registered trademark of AOL LLC in the United States, other countries, or both.

AOL Instant Messenger is a trademark of AOL LLC in the United States, other countries, or both.

Google Talk is a trademark of Google, Inc, in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.