Sametime
Version 9.0

IBM Sametime Advanced 9.0 Software Development Kit

Toolkit Guide

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 Sametime Advanced (program number 5725-M36) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Introduction

The organizational collaboration server for IBM® Sametime® release 9.0, referred to in this guide as Sametime Advanced 9.0, is the latest release of new collaboration technology built on the IBM WebSphere® Application Server platform. This product provides a set of features that extend the traditional Sametime solution with collaborative functionality which can be exploited by external applications by accessing the underlying data and using it in ways that best suit the organization's needs. This document, along with the samples and other documentation in the Sametime Advanced 9.0 Software Development Kit (SDK), provides you with the information you need to build applications that extend the capabilities of Sametime Advanced 9.0 and integrate your own applications with Sametime. This document presents information in the following order:

- An overview of Sametime Advanced 9.0
- An overview of Sametime Advanced 9.0 architecture
- A high-level tutorial on using the Sametime Advanced 9.0 API
- Descriptions of the included samples.

For more information about the contents of the IBM Sametime Advanced 9.0 SDK, see the readme.txt file in the *sta80sdk* subdirectory of the directory in which you installed the SDK.

Terms to know

The following terms are used throughout this integration guide:

Term	Definition
API	Application Programming Interface
Eclipse	An open platform for rich client development. Although Eclipse is a Java™ -based platform, it can be used to build tools for other programming languages
Extension	A mechanism that expands the functionality of a plug-in by connecting to an extension point. An extension is also referred to as a "contribution" to another plug-in.
Extension point	The specification that declares how extensions can add to the functionality of a plug-in. Several plug-ins can contribute to an extension point by defining extensions in the plug-in's extension manifest file, plugin.xml.

IBM Expeditor (Formerly known as WebSphere Everyplace® Deployment)	The platform used by Sametime Connect and IBM managed client products. Expeditor 6.1 provides a runtime environment and integrated middleware components for extending many enterprise applications to server-managed laptop and desktop systems, and mobile devices running supported operating systems. Expeditor includes the Eclipse Rich Client Platform (RCP) and Java Runtime Environment (JRE), as well as additional services used by managed client products. The Expeditor platform is available as a separate product, so that third parties can build their own Rich Client applications.
IDE	Integrated Development Environment, The IBM Rational® Application Developer IDE and Eclipse IDEs are examples of IDEs.
ISV	Independent Software Vendor
J2SE	Java™ 2 Platform, Standard Edition. This is the standard JRE for desktop applications.
J9 JCL Desktop	A custom runtime environment that provides a wide set of features from the Java 2 Platform API core libraries. The J9 JCL Desktop is the runtime environment used by Sametime Connect on Windows and Linux platforms..
JRE	Java Runtime Environment. This is the technology that allows Java applications to run.
OSGi ™	The OSGi Service Platform is a standard that defines, among other things, how Eclipse plug-ins are packaged.
Plug-in	An Eclipse platform feature component. A plug-in is the basic building block of an Eclipse application.
Plug-in registry	Registry of declared plug-ins, extension points, and extensions managed by the Eclipse Runtime Platform.
RTC	Real-time Collaboration, which describes synchronous technologies such as instant messaging, presence awareness, Web conferences, telephony, and so on.
SIP	Session Initiation Protocol, a standard protocol for managing interactive sessions between users. SIP is used for instant messaging, presence, telephony, and a number of other applications.

Suggested reading

IBM Sametime Connect is built using a number of different technologies, including Eclipse, Java and Expeditor.

In order to build plug-ins for the client for Sametime Connect, it is necessary to have some understanding of these technologies. Although you do not need to be an expert, the more familiar you are with these technologies, the more success you will have building plug-ins for Sametime Connect and other Expeditor-based products.

The following list suggests readings including tutorials that you can use to better acquaint yourself with different technologies Sametime uses.

What	Where
Eclipse Organization Community home page	http://www.eclipse.org
Eclipse Workbench User's Guide basic tutorial	http://help.eclipse.org/help32/index.jsp
Eclipse project resources	http://www.ibm.com/developerworks/opensource/top-projects/eclipse.html
The Java Tutorial	http://java.sun.com/docs/books/tutorial/
Java technology	http://www.ibm.com/developerworks/java

Chapter 2. What is IBM Sametime Advanced?

Sametime Advanced Architecture

IBM Sametime Advanced 9.0 is a product in the Sametime family, the leading enterprise Instant Messaging and Web Conferencing product and the platform for IBM's Unified Collaboration (UC2) strategy. IBM Sametime Advanced 9.0 enables advanced users to be more productive and assists forward-looking organizations to leverage their internal knowledge to act in real time to ever-changing market and business situations.

The features provided by IBM Sametime Advanced 9.0 include:

Persistent Chat Rooms

Persistent chat rooms provide server-side support so that users can create, read, and contribute to topic-oriented ongoing chats. Users can also monitor and/or be alerted to new content, new events, and new people in the chat room. Essentially, the features are:

- Users can create, enter, and read/contribute to ongoing chats at any time.
- User can be alerted to new content, events, and people in the chat room.
- Persistent Chat Rooms are linked to Broadcast Channels.

Real-time Broadcast & Collaboration Channels

The broadcast features can be described as having three parts:

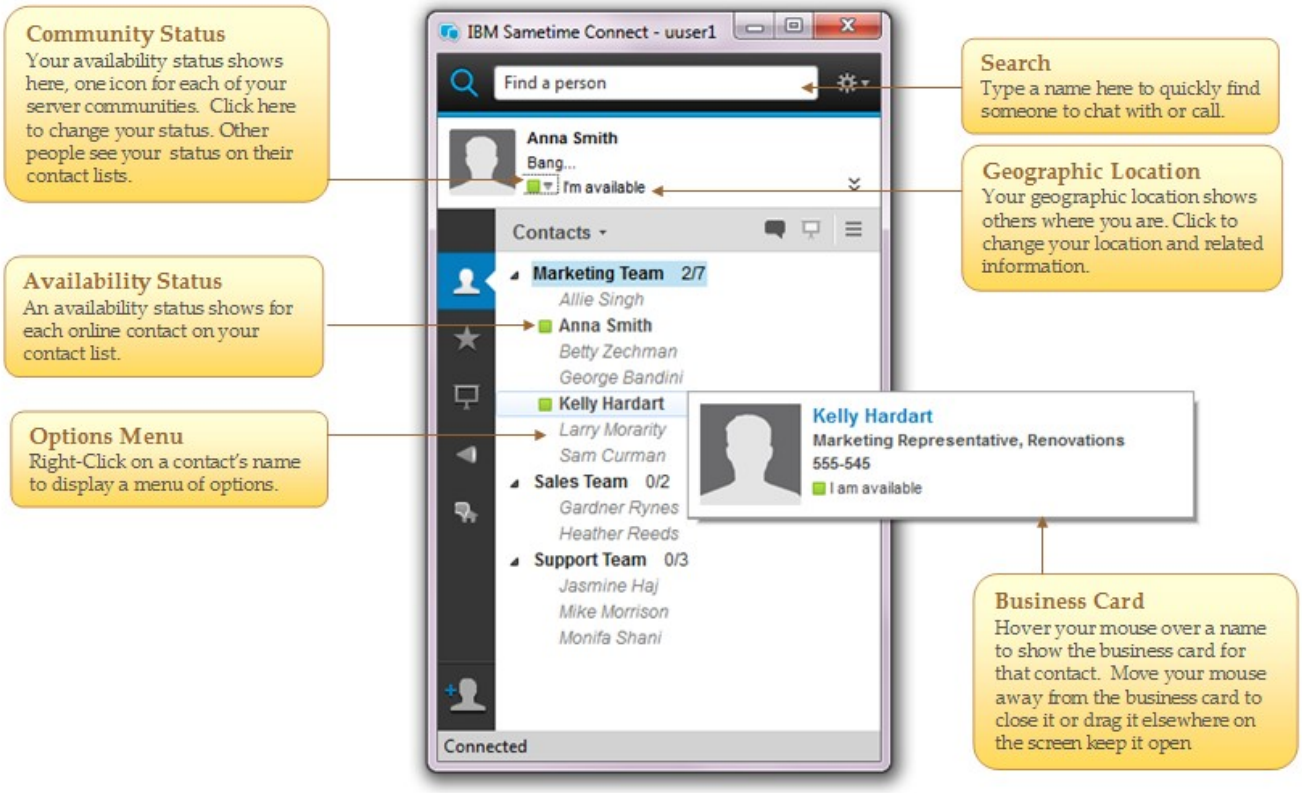
- Alerts –Users can send alert messages to other online users subscribing to the channel. These alerts can allow responses that become chats.
- Instant Polls – Users can create and send single-question polls to online chat subscribers. Subscribers can answer polls. Both sender and responders can see aggregated poll results.
- SkillTap – Users can find and interact with experts, sending questions/requests to channel subscribers. One or more subscribers can offer to answer, and the sender can chat with them to get the needed information. The responses can be easily saved into a set of searchable FAQs, where the original sender can rate the quality of the response.

Location services

- Server-stored locations
- Services built on Sametime dynamic location awareness, such as Nearby Buddies

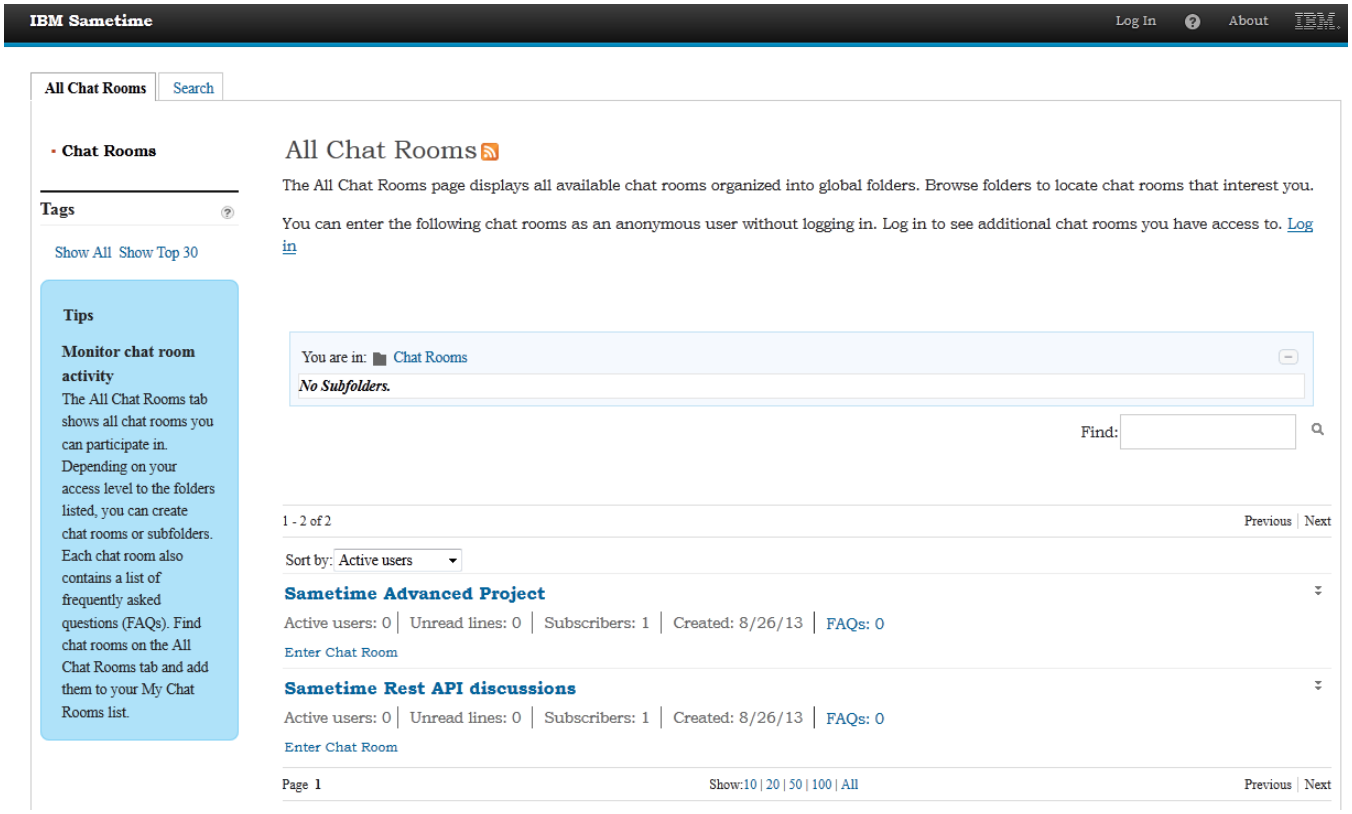
IBM Sametime Connect

The IBM Sametime Connect software gives you award winning and market leading instant messaging capabilities that help you collaborate and keep pace with your real-time work environment. Take a look at the picture below to see some of the basic features.


Community Status
Your availability status shows here, one icon for each of your server communities. Click here to change your status. Other people see your status on their contact lists.
Availability Status
An availability status shows for each online contact on your contact list.
Options Menu
Right-Click on a contact's name to display a menu of options.
IBM Sametime Connect - uuser1
Find a person
Anna Smith
Bang...
I'm available
Contacts
Marketing Team 2/7
Allie Singh
Anna Smith
Betty Zechman
George Bandini
Kelly Hardart
Larry Morarity
Sam Curman
Sales Team 0/2
Gardner Rynes
Heather Reeds
Support Team 0/3
Jasmine Haj
Mike Morrison
Monifa Shani
Connected
Kelly Hardart
Marketing Representative, Renovations
555-545
I am available
Search
Type a name here to quickly find someone to chat with or call.
Geographic Location
Your geographic location shows others where you are. Click to change your location and related information.
Business Card
Hover your mouse over a name to show the business card for that contact. Move your mouse away from the business card to close it or drag it elsewhere on the screen keep it open

IBM Sametime Advanced 9.0 Web Client

IBM Sametime Advanced 9.0 provides a rich web client interface that provides the same user experience as Sametime Connect as well as administration tools.



Why extend IBM Sametime Advanced 9.0?

IBM Sametime Advanced 9.0 can be extended using the provided SDK toolkit. The SDK toolkit exposes integration points that enabled the developer to utilize core Sametime Advanced 9.0 functionality within their own applications.
For example the SDK provides an integration point into the data storage of persistent chat details, via the API.

Chapter 3. The REST API

To retrieve information via the IBM Sametime Advanced 9.0 API in your programs, you use a "feed" to retrieve the data. A "feed" is a special data format optimized for retrieving information that may change over time. For example, the information in a chat is a feed -- each time a new response is added to the conversation, the feed for the chat will indicate what has been added. There are numerous standard formats for feeds, including RSS (Really Simple Syndication) and the Atom Syndication Format. The IBM Sametime Advanced API uses the Atom format, returning content, in the HTTP content type *application/atom+xml*.

REST

Representational State Transfer (*REST*) refers to the simple interfaces that transmit data over HTTP, with no additional messaging layer. The IBM Sametime Advanced 9.0 SDK exposes functionality using a REST API. Utilizing this API simply requires the calling of a URI and inspecting the returned HTML return code, and XML data.
REST maps traditional data manipulation concepts to HTTP verbs in a consistent and simple manner:

Table listing traditional data and mapped HTTP verbs.

Traditional	HTTP
Create	POST
Read	GET
Update	PUT
Delete	DELETE

The data in IBM Sametime Advanced 9.0 can be accessed by any mechanism that supports HTTP, and since XML is used as the data format for the returned data, you can use this API from any program that can send and receive XML over HTTP. Your program can process the XML itself, or use third party libraries such as Apache® Xerces, or Apache Abdera, to assist with the task. In other words, programs to access the data can be created using mark-up like DHTML in browsers, scripting languages like Perl and Python and traditional programming languages like Java, C++, C#, etc.

Parameters

Parameters are designated as required or optional. Superfluous parameters are ignored. If multiple instances of a parameter are specified, the value of the parameter that is last in the URI is used.
Parameter names are CASE SENSITIVE.
Omitting a mandatory parameter results in a Bad Request error (400).

Default values

If an optional parameter is omitted, its default value is used when a new resource is created, using the PUT method. In the case where an existing resource is updated, the value of any parameter that is not specified remains unchanged.

Type definitions

Parameters are passed in the HTTP request, These are of type String. However, certain parameters expect specific content in these strings. Incorrect data results in an error being returned in the form of an HTTP error 400 (Bad Request).
The following is a list of ,many of the the data types used in applications that you may be developing.

Table listing HTTP request types and description of each type.

Type	Description
String	A string
Integer	Only integral numeric values permitted
Boolean	Only the values *TRUE* and *FALSE* permitted (case insensitive)
User	The user's E-mail address (or group name for Groups)
Date	A date in the format MM/DD/YYYY for example: 12/30/2007
Double	Only numeric values permitted

IBM Sametime Advanced 9.0 provides additional parameter-types. These additional types are illustrated below. If values are specified other than those described, an HTTP error 400 (Bad Request) error is returned.

Table listing additional parameter-types for Sametime Advanced.

Name	Description	Permitted Values
ChatAccessMode	Access mode to chats	*public* - anonymous allowed *open* - any logged-in user allowed *restricted* - access by invitation only
FolderQueryType	Type of objects to be returned by a query	all, chats or folder
TextArchiveMode	Archiving mode	never or manual
CommunityType	Type of community	open, private, restrictedPublishers or restrictedRecipients
FolderAccess	Type of folder access	public, inherited or explicit
SearchScope	Scope of a search	all, chatRooms, chatRoomOwners, chatRoomFaqs, chatRoomFaqAuthors, chatRoomTranscripts, communityFaqs, or communityFaqAuthors
MonitoringPeriod	A period for grouping monitoring data	day, week, month, quarter, or year
ChatArchivingMode	Chat archiving mode	off, byDays, or byLines
ArchiveType	Type of chat rooms to filter search results	*NoArchived* – retrieve non-archived chat rooms only *IncludeArchived* – retrieve archived and non-archived chat rooms *OnlyArchived* – retrieved archived chat rooms only
Time	Date and Time	The time format can any of the following: YYYY-MM-DDThh:mm:ssZ YYYY-MM-DDThh:mm:ss YYYY-MM-DDThh:mmZ YYYY-MM-DDThh:mm YYYY-MM-DDZ YYYY-MM-DD Time format notes: The "T" in each of the above formats is the constant string "T", which is the ISO standard indicating the beginning of a time string. Z is the

		three letter time zone designator (e.g. GMT or EST). If the time is omitted, then 12:00:00 AM is used. If the time zone is omitted, then GMT is used.

Arrays

Arrays are passed as a string array of concatenated values, where the elements of the array are separated by the separator character "$" (dollar). An empty array is expressed by the null string.
Arrays are indicated in the parameter types below as the type followed by square brackets, e.g. *String[]*.

Returned information

Creating a new resource (POST), updating an existing resource (PUT), or deleting a resource (DELETE) will return no content if successful. If not successful, a description of the reason for failure is returned. Each of the REST API calls that return data (the GET requests), do so in the form of XML formatted according to the Atom standard. This data has the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:stx="http://www.ibm.com/xmlns/prod/SametimeAdvanced">
   <id>http://%server%:%port#%/stadvancedapi/api/folder/globalbyid/%folderId%</id>
   <!-- OR <id>http://%server%:%port%/stadvancedapi/api/folder/userbyid/%userid
%</id> -->
   <!--Same date as most recently edited chat
      Not currently available in JSON API -->
   <updated>%updated%<!-- RFC339 e.g. 2006-11-15T18:30:01.078Z --></updated>
   <!-- Display Name of the current folder -->
   <title type="text">Contents of %displayName%</title>
   <!-- generator the agent used to generate a feed for debugging -->
   <generator uri="http://www.ibm.com/xmlns/prod/sn" version="1.0">IBM Sametime
Advanced 8.0 - Persistent Chat</generator>
   <entry>
      <!-- An Entry for a Chat -->
      <id>http://%server%:%port%/stadvancedapi/api/chat/id/%id%</id>
      <title type="text">%displayName%</title>
      <archived>%archived%</archived>
      <summary type="text">%description%</summary>
      <link href="http://%server%:%port#%/stadvancedapi/api/chat/id/%id%"
rel="alternate" type="application/atom+xml"></link>
      <!-- Timestamp of chat creation Not currently available in JSON API  -->
      <published>%published%<!-- RFC339 e.g. 2006-11-15T18:30:01.078Z
--></published>
      <author>
         <name>%creatorDisplayName%</name>
         <id>%creatorId%</id>
         <email>%creatorEmail%</email><!-- NOT IMPLEMENTED YET -->
      </author>
      <!-- Not currently available in JSON API  -->
      <updated>%updated%<!-- RFC339 e.g. 2006-11-15T18:30:01.078Z --></updated>
      <category term="%type%(i.e. PersistentChat)"
scheme="http://www.ibm.com/xmlns/prod/SametimeAdvanced/type"></category>
   </entry>
   <entry>
      <!-- An Entry for a Subfolder -->
      <id>http://%server%:%port%/stadvancedapi/api/folder/globalbyid/%sub
folderid%</id>
      <!-- OR <id>http://%server%:%port%/stadvancedapi/api/folder/userbyid/
%subfolderid%</id> -->
      <title type="text">%displayName%</title>
      <summary type="text">%description%</summary>
```

```
      <link href="http://%server%:%port#%/stadvancedapi/api/folder/globalbyid/
%subFolderId%" rel="alternate" type="application/atom+xml"></link>
      <!-- OR <link href="http://%server%:%port#
%/stadvancedapi/api/folder/userbyid/%subFolderId%" rel="alternate"
type="text/html"></link> -->
      <!-- Timestamp of folder creation Not currently available in JSON API  -->
      <published>%published%<!-- RFC339 e.g. 2006-11-15T18:30:01.078Z
--></published>
      <author>
         <name>%creatorDisplayName%</name>
         <id>%creatorId%</id>
         <email>%creatorEmail%</email>
      </author>
      <!-- Same date as most recently edited chat in this subfolder Not currently
available in JSON API  -->
      <updated>%updated%<!-- RFC339 e.g. 2006-11-15T18:30:01.078Z --></updated>
      <category term="%type%(i.e. Folder)"
scheme="http://www.ibm.com/xmlns/prod/SametimeAdvanced/type"></category>
   </entry>
   <!-- May need other entries for InstantPolls/FAQs etc -->
</feed>
```

General calling conventions

The REST API calls all have the same basic format:

```
{HOST-SEGMENT}/stadvancedapi/api/{FUNCTIONAL_AREA}[/IDENTIFIER][?
PARAMETERS]
```

where:

- HOST-SEGMENT refers to the server name and port
- FUNCTIONAL-AREA specifies the kind of API call
- IDENTIFIER refers to the specific object being accessed
- PARAMETERS is the list of parameters on the URI

Note: That the maximum length of a URL is effectively 2083 characters, since this is the maximum size that IE will accept (the limitations for Firefox, Safari and Opera are over 100000 characters). In other words, when you send a message to the server, i.e. make a REST API call, the total size of the string must be less than 2083 characters after encoding.

The REST API calls all return XML data as an Atom feed. An example of this is provided below:

```
<?xml version="1.0" encoding="UTF-8"?>

<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:sta="http://www.ibm.com/xmlns/prod/sta">
   <id>http://localhost:9080/stadvancedapi/api/folder/globalbyid/20080312-1449-
0564-6634-000000000000</id>
   <updated>2008-03-12T14:49:05+00:00</updated>
   <title type="text">Contents of /testfolder</title>
   <sta:managerAccess></sta:managerAccess>
   <sta:managers>
   </sta:managers>
   <sta:writerAccess></sta:writerAccess>
   <sta:writers>
   </sta:writers>
   <sta:readerAccess></sta:readerAccess>
   <sta:readers>
   </sta:readers>
   <generator uri="http://www.ibm.com/xmlns/prod/sta" version="1.0">Sametime
Advanced 8.0</generator>
      <entry>
            <id>http://localhost:9080/chat/id/20080312-1611-4917-9756-
000000000000</id>
            <title type="text">test3</title>
            <category term="PersistentChat"
scheme="http://www.ibm.com/xmlns/prod/sta/type"></category>
            <archived>0</archived>
            <summary type="text"></summary>
               <link
href="http://localhost:9080/stadvancedapi/api/chat/globalbyid/20080312-1611-4917-
9756-000000000000" rel="self" type="application/atom+xml"></link>
```

```
            <link href="http://localhost:9080/stadvanced/controller?
meetingId=20080312-1611-4917-9756-000000000000" rel="alternate"
type="text/html"></link>
            <published>2008-03-12T16:11:48+00:00</published>
            <author>
               <name>IBM USER</name>
               <email>IBM_USER@ibm.com</email>
            </author>
      </entry>
</feed>
```

Functionality available through REST APIs

Since the API call is expressed as a URI, data can be read using any language that can process URIs and HTTP requests.

This code sample reads data from the specified URL, returning it as a string of xml.

```
private String getAtomData(String url)
      throws IOException, UnknownServiceException     {
        URL requestURL = new URL(url);
        HttpURLConnection myConn = (HttpURLConnection)requestURL.openConnection();
        myConn.setRequestMethod("GET");
        myConn.setAllowUserInteraction(true);
        myConn.setUseCaches(false);
        myConn.setRequestProperty("Content-Type", "text/xml");
        myConn.connect();
        StringBuffer sb = new StringBuffer("");
        if (myConn.getContentType().equals("text/xml")) {
         int atomLen = myConn.getContentLength();
         if (atomLen > 0 ) {
            InputStream is = myConn.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);
                String line = null;
                while ((line = br.readLine()) != null) {
                    sb.append(line);
                }
         }
        } else {
         throw(new java.net.UnknownServiceException("Unexpected type: "
                                            + myConn.getContentType()));
        }
        return sb.toString();
}
```

REST API Security Considerations

Authentication is required in order to use the REST APIs. In addition, there are extra security constraints in place for the Licensing APIs. See section 4.12 for more details.

Using secure connections

When the API call uses SSL, i.e. the protocol of the URI is HTPS, the behavior of the connection is slightly different. While the API will generally behave so that the extra layer of security is transparent, the connection may fail security checks at an application level.

Error conditions

If an API call encounters an error, the response is a predictable HTTP error status. While the status codes are described in RFC 2616, “Hypertext Transfer Protocol -- HTTP/1.1”, section 10, are relevant, the following are explicitly returned as a result of errors encountered by the API:

Table listing HTTP response status.

Code	Name	Description
200	OK	The request has succeeded.
201	Created	The request has been fulfilled and resulted in a new resource being created.
400	Bad Request	The request could not be understood by the server due to malformed syntax.
401	Unauthorized	The request requires user authentication.
403	Forbidden	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated.
404	Not Found	The server has not found anything matching the Request-URI.
405	Method Not Allowed	The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
410	Gone	The requested resource is no longer available at the server and no forwarding address is known.
500	Internal Server Error	Generic message when the server in unable to performing the request

Chapter 4. REST API Reference

Below is a description of the REST API. The calls are grouped according to functionality.

Global folders

A global folder can be addressed by its path or the folder id, using the following URIs:

```
/stadvancedapi/api/folder/global/<folder_path>
/stadvancedapi/api/folder/globalbyid/<folder_id>
```

The properties of global folders are listed here:

Table listing properties of global folders.

Name	Type	Element	Comment
Id	String	<id>	
Name	String	<title>	
ManagerAccess	FolderAccess	<sta:managerAccess>	
Managers	User[]	<sta:managers>	Empty if managerAccess is "public"
WriterAccess	FolderAccess	<sta:writerAccess>	
Writers	User[]	<sta:writers>	Empty if writerAccess is "public"
ReaderAccess	FolderAccess	<sta:readerAccess>	
Readers	User[]	<sta:readers>	Empty if readerAccess is "public"
group		<sta:group>	This tag is only included if the user is a group
globalParentFolder	String	<sta:globalParentFolder>	Folder path to the folder in the global folder tree.
allowsAnonymous		<sta: allowsAnonymous>	This tag is only included if the chat room allows anonymous access. The content of the tag is empty.
restrictedAccess		<sta:restrictedAccess>	This tag is only included if the chat room restricts access to the member/manager list. If the tag is omitted, then the chat room allows access to any authenticated user, regardless of the member/manager list.
globalParentFolder	String	<sta:globalParentFolder>	Returns the complete folder path to the chat room or folder in the global folder tree.

Getting the properties and contents of a global folder

To retrieve a folder's properties and content, use HTTP GET with the following parameters:

Table listing HTTP GET parameters for retrieving folder's properties.

Name	Type	Required	Default	Description
pageIndex	Integer	Optional	0	The page index (for paging)
elementsPerPage	Integer	Optional	100	Number of elements in a page (for paging)
type	FolderQueryType	Optional	*all*	Type of objects returned: *all*, *persistent* or *folder*
includeNestedFolders	Boolean	Optional	FALSE	If true, subfolders will also be included in the result
nestedFolderLevels	Integer	Optional	1	Only used if includeNestedFolders is true. If specified will recurse through subfolders the number of levels specified.
includeArchived	ArchiveType	Optional	NoArchived	Determines whether or not to return chat rooms that archived
owner	User	Optional		Will filter chat rooms with this owner
manager	User	Optional		Will search only chat rooms where this user is the manager, either explicitly or implicitly Note that this parameter cannot be used in conjunction with the owner parameter. If both are used, the manager parameter will be ignored.
participant	User	Optional		Will search only chat rooms where this user can be a participant, either explicitly or implicitly. Otherwise, if set to *anonymousAccess* only chat rooms allowing anonymous will be retuned. If set to *authenticatedAccess* only chat rooms which require authentication will be returned. Note that this parameter cannot be used in conjunction with the manager parameter. If both are used, the participant parameter will be ignored.

Creating a global folder

To create a folder, use HTTP POST with the following parameters. The folder whose path is specified in the URI is created. If the second form of the URI is used, the id provided is the id of the parent folder and must be followed by the name of the folder to be created. Since the name is part of the URI when including extended ASCII characters or any other special characters (e.g., space), the URI will need to be properly encoded. The URI has the format:

```
/
stadvancedapi/api/folder/globalbyid/<parent_folder_id>/<name_of_folde
r_to_
create>
```

To create a folder in the global root folder ("Chat Rooms") using this second URI format (globalbyid), use the special id "globalRootFolderId" as the parent folder id.

Table listing parameters using by creating a global folder.

Name	Type	Required	Default	Description
restrictSubfolderAccess	Boolean	Optional	false	When the user restriction is imposed, any users added to member roles for chat rooms or subfolders must also appear in the membership roles for the parent folder. When the user restriction is not imposed, users added to the chat rooms or subfolders will be able to see the parent folders to navigate to the chat room or subfolder they have access to
managerAccess	FolderAccess	Optional	explicit	Manager access: public, inherited or explicit
managers	User[]	Mandatory where managerAccess is *explicit*		Explicit list of managers
writerAccess	FolderAccess	Optional	inherited	Writer access: public, inherited or explicit
writers	User[]	Mandatory where writerAccess is *explicit*		Explicit list of writers
readerAccess	FolderAccess	Optional	inherited	Reader access: public, inherited or explicit
readers	User[]	Mandatory where readerAccess is *explicit*		Explicit list of readers

Updating a global folder

To update the properties of a folder, use HTTP PUT with the following parameters:

Table listing HTTP PUT parameter for updating a folder.

Name	Type	Required	Default	Description
name	String	Optional	the current name	The new name of the folder
restrictSubfolderAccess	Boolean	Optional	false	When the user restriction is imposed, any users added to member roles for chat rooms or subfolders must also appear in the membership roles for the parent folder. When the user restriction is not imposed, users added to the chat rooms or subfolders will be able to see the parent folders to navigate to the chat room or subfolder they have access to
managerAccess	FolderAccess	Optional	the current value	Manager access: public, inherited or explicit
managers	User[]	Mandatory where managerAccess is *explicit*	the current value	Explicit list of managers
writerAccess	FolderAccess	Optional	the current value	Writer access: public, inherited or explicit
writers	User[]	Mandatory where writerAccess is *explicit*	the current value	Explicit list of writers
readerAccess	FolderAccess	Optional	the current value	Reader access: public, inherited or explicit
readers	User[]	Mandatory where readerAccess is *explicit*	the current value	Explicit list of managers

Deleting a global folder

To remove a folder, use HTTP DELETE. No special parameters required.

User folders

User folders provide a mechanism where a particular user can organize chats that are of interest to them within a personal folder structure. This folder structure, which is the set of folders under "My Chats", is only accessible to that single user. Chats (which will always be contained within a global folder) may be added to these user folders for ease of access. Note that the user folders just reference the chats contained within the main global folder structure. Adding a chat into a user folder does not remove it from its parent global folder.

In a manner similar to accessing global folders, the following URIs are used to address user folders:

```
/stadvancedapi/api/folder/user/<folder_path>
/stadvancedapi/api/folder/userbyid/<folder_id>
```

User folders have the following properties:

Table listing parameters for addressing user folder.

Name	Type	Element	Comment
Id	String	<id>	
name	String	<title>	
userParentFolder	String	<sta: userParentFolder>	Returns the complete folder path to the chat room or folder in the user folder tree.

The actions available for global folders are also available for user folders, but accept a reduced set of parameters since non-public user folders don't have access rights restrictions.

Getting the properties and contents of a user folder

To retrieve the properties and content of a folder, use HTTP GET with the following parameters:

Table listing parameters for retrieving the properties and contents of a folder.

Name	Type	Required	Default	Description
pageIndex	Integer	Optional	0	The page index (for paging)
elementsPerPage	Integer	Optional	100	Number of elements in a page (for paging)
type	FolderQueryType	Optional	all	Type of objects returned: *all*, *persistent* or *folder*
includeArchived	ArchiveType	Optional	NoArchived	Determines whether or not to return chat rooms that archived
user	User	Optional	The authenticated user	In order to use this parameter, caller must be in the AdminUsers role. The return data will be identical to the return result as if the user id passed in was making the call

Creating a user folder

To create a user folder, use HTTP POST. The folder specified by *folder_path* is created if the first format of the URI is used. If the second format is used, the ID of the parent folder is specified and is followed by the name of the folder to be created; for example:

```
/
stadvancedapi/api/folder/userbyid/<parent_folder_id>/<folder_to_creat
e>
```

Note since the name is part of the URI when including extended ASCII characters or any other special characters (e.g., space), the URI will need to be properly encoded.

To create a chat in the user root folder ("Chat Rooms") using the second format (userbyid), use the special id "userRootFolderId" as the parent folder id.
Unlike global folders, user folders do not have configurable access control (each user can only see their own personal user folders), so there are no parameters used when creating a user folder.

Updating a user's folder properties

To update a folder's properties, use HTTP PUT with the following parameters:

Table listing parameters for updating user folder.

Name	Type	Required	Default	Description
name	String	Optional	the current name	The new name of the folder

Adding a chat to a user folder

To add a chat to a user folder, use HTTP PUT with the following parameters:

Table listing parameters for adding a chat to user folder.

Name	Type	Required	Default	Description
addchatid	String	Mandatory		The id of the chat to be added
index	Integer	Optional	0	The index of the chat in the folder

Removing a chat from a user folder

To remove a chat from a user folder, use HTTP PUT with the following parameter:

Table listing parameters for removing a chat from user folder.

Name	Type	Required	Default	Description
removechatid	String	Mandatory		The id of the chat to be removed

Deleting a user folder

To delete a user folder, use HTTP DELETE with no additional parameters.

Persistent Chats

Chats can be accessed by global or user path, or by id, addressed by the following URIs:

```
/stadvancedapi/api/chat/global/<chat_path>
/stadvancedapi/api/chat/user/<chat_path>
/stadvancedapi/api/chat/id/<chat_id>
```

The *chat_path* is the path of the parent folder followed by the name of the chat, for example:

```
/stadvancedapi/api/chat/global/globalfolder/subfolder/chat
```

The following are the properties of a persistent chat. Note that monitoring data is provided in the feed (unread lines, active users and faqs):

Table listing properties of a persistent chat.

Name	Type	Element	Comment
Id	String	<id>	
Name	String	<title>	
Description	String	<subtitle>	
createCommunity	Boolean	<sta:createCommunity>	The created community id, if any; empty otherwise
Tags	String[]	<sta:tags>	
Creator	User	<author>	The creator of the chat room
Owners	User[]	<sta:owners>	The owners/managers of the chat room
Invitees	User[]	<sta:invitees>	The members of the chat room
group		<sta:group>	This tag is only included if the user is a group
Unread	Integer	<sta:unread>	Unread entries for user (only if logged in)
Active	Integer	<sta:active>	Active users
Faqs	Integer	<sta:faqs>	Number of FAQs for user
globalParentFolder	String	<sta:globalParentFolder>	Folder path to the chat room in the global folder tree.
allowsAnonymous		<sta:allowAnonymous>	Only included if the chat room allows anonymous access. The contents of the tag is empty.
restrictedAccess		<sta:restrictedAccess>	Only included if the chat room restricts access to the member/manager list. If the tag is omitted, then the chat room allows access to any authenticated user, regardless of the member/manager list.
files		<sta:files>	This is a container for all of the files
file		<sta:file>	This tag will be returned for each file posted in the chat room. It will contain id, title, published and author nested tags similar to the chat text entry tag.
CurrentUsers	User[]	<sta:currentUsers>	Returns the list of users currently active in the persistent chat. Note that this field is only returned if the "includeUsers" optional parameter is specified.

Getting the properties and transcript of a persistent chat

To retrieve the properties of a chat along with a transcript of its content, to include posted files, use HTTP GET with the following parameters:

Table listing parameters for getting the properties and transcript of a persistent chat.

Name	Type	Required	Default	Description
password	String	Optional		The password for chat room access – required to access persistent chats that are password protected
pageIndex	Integer	Optional	0	The index of the page to retrieve
elementsPerPage	Integer	Optional	100	The number of elements per page
byDays	Boolean	Optional	FALSE	Retrieval by days (*true*) or elements (*false*)
includeArchived	ArchiveType	Optional	NoArchived	Determines whether or not to return chat rooms that archived
startTime	Time	Optional		Returns only transcript entries after the specified time.
endTime	Time	Optional		Returns only transcript entries before the specified time.
includeUsers	Boolean	Optional	FALSE	If *true*, the <*sta:currentUsers*> section will be added to the chat data returned by the request

Creating a persistent chat

To create a persistent chat, use HTTP POST with the following parameters. The persistent chat whose path is specified in the URI is created. If the URI with the id is used, the id provided is the id of the parent folder and must be followed by the name of the chat to be created. Since the name is part of the URI when including extended ASCII characters or any other special characters (e.g., space), the URI will need to be properly encoded. The URI has the format:

```
/stadvancedapi/api/chat/id/<parent_folder_id>/<name_of_chat_to_create>
```

To create a chat in the global root folder ("Chat Rooms") you should use the special id "globalRootFolderId" as the parent folder id. Creation of a chat must use the global folder by path or parent folder by id URI pattern. It is not possible to create a chat using the user folder URI pattern.

Table listing parameters for creating a persistent chat.

Name	Type	Required	Default	Description
description	String	Optional	empty string	The chat description
password	String	Optional	empty string	Password to the chat
createCommunity	Boolean	Optional	FALSE	Creates a community also if set to true
tags	String[]	Optional	empty list	Tags to be associated with the chat
owner	User[]	Mandatory		The owners/managers of the chat room
invitees	User[]	Optional	empty list	The members of the chat room
access	ChatAccessMode	Optional	open	Access Level to the chat room
emailNotify	Boolean	Optional	FALSE	Send a notification email to members
invitationAllowed	Boolean	Optional	FALSE	Whether invitees can invite others
autoHide	ChatArchivingMode	Optional	byDays	Automatic hiding mode
autoHideArg	Integer	Mandatory where autoHide is not *off*	5	If autoHide = "byDays", number of days to keep; if autoHide = "byLines", number of lines to keep

autoArchive	ChatArchivingMode	Optional	byDays	Automatic archiving mode
autoArchiveArg	Integer	Mandatory where autoArchive is not *off*	10	If autoArchive = ”byDays”, number of days to keep; if autoArchive = “byLines”, number of lines to keep

Updating a persistent chat

To update the properties of a persistent chat, use HTTP PUT with these parameters:

Table listing parameters for updating a persistent chat.

Name	Type	Required	Default	Description
entry	String	Optional		A text entry to add to the persistent chat – nothing is added if the parameter is not present
title	String	Optional	current value	The chat title
description	String	Optional	current value	The chat description
password	String	Optional	current value	Password to the chat
createCommunity	Boolean	Optional	current value	Creates a community also if set to true
tags	String[]	Optional	current value	Tags to be associated with the chat
owner	User[]	Optional	current value	The owners/managers of the chat room
invitees	User[]	Optional	current value	The members of the chat room
access	ChatAccessMode	Optional	current value	Access Level to the chat room
emailNotify	Boolean	Optional	current value	Send a notification email to members
invitationAllowed	Boolean	Optional	current value	Can participant invite others
autoHide	ChatArchivingMode	Optional	current value	Automatic hiding mode
autoHideArg	Integer	Mandatory where autoHide is not *off*	current value	If autoHide = ”byDays”, number of days to keep; if autoHide = “byLines”, number of lines to keep
autoArchive	ChatArchivingMode	Optional	current value	Automatic archiving mode
autoArchiveArg	Integer	Mandatory where autoArchive is not *off*	current value	If autoArchive = ”byDays”, number of days to keep; if autoArchive = “byLines”, number of lines to keep
archived	Boolean	Optional		Archive or unarchive the chat room. Note that this is the same functionality as in the UI to set the archived flag for the chat room. It does not perform a database archive.

Deleting a persistent chat

To remove a chat, use HTTP DELETE, and this will remove the chat specified by the path or ID.

Access Control

The following URI is used to determine whether a set of users have access to a chat room:

```
/stadvancedapi/api/chat/access/<chatid>
```

To determine whether a set of users have access to a chat room, use HTTP GET with these parameters:

Table listing parameters for setting authenticated users.

Name	Type	Required	Default	Description
users	User[]	Optional	The authenticated user	The users whose access is being queried.

Note that this API does not take into consideration whether the user knows the password to a chat room. The properties returned by the access api are:

Table listing properties returned by access api.

Name	Type	Element	Description
hasAccess	User	<sta:hasAccess>	This tag will be returned for each user that has access to the chat room. The tag will be returned as a subtag to the <sta:person> tag.
noAccess	User	<sta:noAccess>	This tag will be returned for each user that does not have access to the chat room

Chat search

The following URI is used for a persistent chat search:

```
/stadvancedapi/api/chat/all
```

Performing a filtered search of persistent chats

To search the available chats, use HTTP GET with the parameters below:
The result of the search will be a list of persistent chats similar to what would be returned by a folder query. The search can be executed either for specific tags, for text in the chat description, or for my owner.

Table listing parameters for searching available chats.

Name	Type	Required	Default	Description
tags	String[]	Optional		A list of tags - will return chats that contain all the searched tags
searchText	String	Optional		Will search in the chat description for this text
owner	User	Optional		Will search for chats with this owner
manager	User	Optional		Will search only chat rooms where this user is the manager, either explicitly or implicitly Note that this parameter cannot be used in conjunction with the owner parameter. If both are used, the manager parameter will be ignored.
participant	User	Optional		Will search only chat rooms where this user can be a participant, either explicitly or implicitly. Otherwise, if set to *anonymousAccess* only chat rooms allowing anonymous will be retuned. If set to *authenticatedAccess* only chat rooms which require authentication will be returned. Note that this parameter cannot be used in conjunction with the manager

				parameter. If both are used, the participant parameter will be ignored.
includeArchived	ArchiveType	Optional	NoArchived	Determines whether or not to return chat rooms that archived
hasJoined	User	Optional		Will return only chat rooms where this user has joined the chat room at least once Note that this parameter cannot be used in conjunction with the owner or manager parameters. If either the manager or owner parameters are used, the participant parameter will be ignored.

Community

Communities are identified by their name. The URI to access them is:

```
/stadvancedapi/api/community/<community_name>
```

The properties of communities are:

Table listing properties of communities.

Name	Type	Element	Description
Id	String	<id>	The community id
Name	String	<title>	The community title
description	String	<subtitle>	The community description
Managers	User[]	<sta:managers>	Managers of the community
Members	User[]	<sta:members>	Members of the community
group		<sta:group>	This tag is only included if the user is a group
Type	CommunityType	<sta:communityType>	The community type

Retrieving the properties of a community

Use HTTP GET to retrieve the properties of a community specified by *community_name* in the URI.

Retrieving a list of communities

To retrieve a list of communities with which a user is associated with, use HTTP GET with the following URI and parameter. Note to retrieve the entire list of communities, *community_name* in the URI is omitted as illustrated below.

```
/stadvancedapi/api/community
```

Table listing parameter for retrieving a list of communities.

Name	Type	Required	Default	Description
types	CommunityType[]	Optional		If omitted, returns communities of all types. Otherwise returns the communities of the specified types.

Creating a community

To create a community, use HTTP POST with the following parameters. This creates the community specified as *community_name* in the URI. Since the name is part of the URI when including extended ASCII characters or any other special characters (e.g., space), the URI will need to be properly encoded.

Table listing parameters for creating a community.

Name	Type	Required	Default	Description
description	String	Optional		The community description
manager	User[]	Mandatory		Managers of the community
members	User[]	Mandatory when type is other than *open*		Members of the community
type	CommunityType	Mandatory		One of open, private, restrictedPublishers, restrictedRecipients

Updating a community

To update a community specified as *community_name* in the URI, use HTTP PUT with the following parameters:

Table listing parameters for updating a community.

Name	Type	Required	Default	Description
name	String	Optional	Current value	The community name
description	String	Optional	Current value	The community description
managers	User[]	Optional	Current value	Managers of the community
members	User[]	Optional	Current value	Members of the community
type	CommunityType	Optional	Current value	One of open, private, restrictedPublishers, restrictedRecipients

Deleting a community

To remove a community specified by the *community_name* in the URI, use HTTP DELETE.

Alerts

The REST URI to broadcast an alert to a community is:

```
/stadvancedapi/api/communityalerts
```

Sending an alert

To send an alert, use HTTP POST with the following parameters:

Table listing parameters for sending an alert.

Name	Type	Required	Default	Description
community	String	Mandatory		The name of the community to which the message is broadcast
message	String	Mandatory		The message to send in the alert

Chat FAQs

Frequently asked questions (FAQs) contain a question and answer that can be edited and rated. FAQs can be added and removed from chats. The properties of FAQs are:

Table listing properties of FAQs.

Name	Type	Element	Description
Id	String	<id>	
Name	String	<title>	
Question	String	<subtitle>	
Answer	String	<sta:answer>	
Author	User	<author>	

Retrieving a FAQ entry

To retrieve a specific FAQ, use HTTP GET, specifying the FAQ's ID in the URI:

```
/stadvancedapi/api/faq/<faq_id>
```

Retrieving a list of FAQs

To retrieve a list of all FAQs, use HTTP GET with the following URI:

```
/stadvancedapi/api/faq/all
```

To retrieve all FAQs associated with a specific chat, use HTTP GET with the above URI with the following parameter. Note that this returns the question and the ID of each FAQ, and the complete data associated with a specific FAQ can be retrieved by means of the FAQ ID:

Table listing parameter for retrieving FAQs.

Name	Type	Required	Default	Description
chatid	String	Mandatory		The id of the chat from which the FAQ is retrieved

Creating a FAQ entry

To create a FAQ with the specified ID, use HTTP POST with the following parameters:

Table listing parameter for creating FAQs.

Name	Type	Required	Default	Description
chatid	String	Mandatory		The id of the chat to which the FAQ is added
question	String	Mandatory		The FAQ question
answer	String	Mandatory		The FAQ answer

Updating a FAQ entry

To update the properties of a FAQ, i.e. its question and answer, use HTTP PUT with the following parameters:

Table listing parameter for updating FAQs.

Name	Type	Required	Default	Description
question	String	Mandatory		The FAQ question
answer rating	String Integer	Mandatory Optional		The FAQ answer The rating for this FAQ; if not present, no rating is assigned

Retrieving a FAQ entry

To remove a specific FAQ, use HTTP DELETE, specifying the FAQ's ID in the URI.

SkillTaps

Skilltaps, or Community FAQs, are rated answers to a question sent to a community. The URI used to address Skilltaps is:

```
/stadvancedapi/api/skilltap
```

Skilltap properties are provided as one entry per Skilltap:

Table listing properties of Skilltap.

Name	Type	Element	Description
Id	String	<id>	http://.../stadvancedapi/api/skilltap/<id>
Question	String	<title>	The question
Requester	User	<author>	The person asking the question
timeRequested	Date	<published>	The time of asking the question
timeLastAnswered	Date	<updated>	The time of the last response
Response	String	<sta:text>	contained in <sta:skilltapResponse>
Rating	Double	<sta:rating>	contained in <sta:skilltapResponse>

Retrieving a list of skilltaps

To get a list of Skilltaps for a given community, use a GET request with the following URI:

```
/stadvancedapi/api/skilltap/<community>
```

Since the name is part of the URI when including extended ASCII characters or any other special characters (e.g., space), the URI will need to be properly encoded.

The following optional parameter may be used:

Table listing parameter for retrieving Skilltaps of a given community.

Name	Type	Required	Default	Description
user	User	Optional		If not specified, retrieve the Skilltaps asked by all users. Otherwise retrieve only those asked by the specified user.

To get a list of Skilltaps across all communities, use a GET request with the following URI:

```
/stadvancedapi/api/skilltap/all
```

The following optional parameter may be used:

Table listing parameter for retrieving Skilltaps across all communities.

Name	Type	Required	Default	Description
user	User	Optional		If not specified, retrieve the Skilltaps asked by all users. Otherwise retrieve only those asked by the specified user.

Rating a skilltap response

To add a rating for a Skilltap response, use HTTP PUT with the following URI:

```
/stadvancedapi/api/skilltap/<responseid>
```

The following parameter is required:

Table listing parameter for rating a Skilltap response.

Name	Type	Required	Default	Description
rating	Integer	Mandatory		The rating for the skilltap response: a number between 1 (poor) and 5 (excellent)

Location Service

The location of a user is determined by the last known location heartbeat received for that user. To interrogate the location of a list of users, the following URI may be used:

```
/stadvancedapi/api/location/
```

Location properties are provided as one entry per user:

Table listing properties of Location Service.

Name	Type	Element	Description
User	String	<id>	/stadvancedapi/api/location?users=<user>
User	User	<author>	The user whose location is provided
activeTimestamp	String	<published>	timestamp of last heartbeat received for this user indicating active status (in the Sametime sense)
Timestamp	String	<updated>	timestamp of last heartbeat received for this user
Location	String	<title>	the location string for this user from last heartbeat
Status	String	<sta:status>	the last reported status for this user (0=unknown, 1=inactive, 2=active)

Retrieving the location information for a list of users

To retrieve the locations of a list of users, use HTTP GET with the parameters:

Table listing parameter for retrieving the location information.

Name	Type	Required	Default	Description
users	User[]	Mandatory		The list of users from which to retrieve location information

Submitting location information

To submit location information, use HTTP PUT with the following parameters:

Table listing parameter for submitting location information.

Name	Type	Required	Default	Description
subnet	String	Mandatory		The subnet id of the router where users will connect. Usually in the form 9.xx.xx.0
netmask	String	Mandatory		The netmas of the router where users will connect. Usually in the form 255.255.255.0

macaddr	String	Mandatory		The mac address of the router where users will connect. Usually in the form 00000C07AC01.
city	String	Optional		The city for this location. Although city is optional, one of city, state or country must be specified.
state	String	Optional		The state for this location. Although state is optional, one of city, state or country must be specified.
country	String	Optional		The two digit country code for this location. Although country is optional, one of city, state or country must be specified. The country codes conform to the ISO standard (http://www.iso.org/iso/english_country_names_and_code_elements)
zipcode	String	Optional		The zip code for this location.
timezone	String	Optional		The time zone for this location. The time zone format string is based on the java TimeZone format (e.g. "Eastern Time (GMT -05:00)")

Deleting location information

To delate a location information, use HTTP DELETE with the below parameters. Note if any user location records are using postal code information, then the delete command will fail.

Table listing parameter for deleting location information.

Name	Type	Required	Default	Description
subnet	String	Mandatory		The subnet id of the router where users will connect. Usually in the form 9.xx.xx.0
netmask	String	Mandatory		The netmas of the router where users will connect. Usually in the form 255.255.255.0
macaddr	String	Mandatory		The mac address of the router where users will connect. Usually in the form 00000C07AC01.
city	String	Optional		The city for this location. Although city is optional, one of city, state or country must be specified.
state	String	Optional		The state for this location. Although state is optional, one of city, state or country must be specified.
country	String	Optional		The two digit country code for this location. Although country is optional, one of city, state or country must be specified. The country codes conform to the ISO standard (http://www.iso.org/iso/english_country_names_and_code_elements)
zipcode	String	Optional		The zip code for this location.
timezone	String	Optional		The time zone for this location. The time zone format string is based on the java TimeZone format (e.g. "Eastern Time (GMT -05:00)")

Chat Creation Workflow

If the workflow feature has been enabled in the Sametime Advanced Server administration settings,, then chats require an additional approval step after creation. The REST URI to access chat creation workflow information is:

```
/stadvancedapi/api/workflow/chatqueue
```

The properties of chat creation workflow are (one entry per chat in the queue):

Table listing properties of chat creation workflow.

Name	Type	Element	Description
Id	String	<id>	The chat id
Title	String	<title>	The chat name
Creator	User	<author>	The person who created the chat
description	String	<content>	The chat description

Retrieving a list of chats in the approval queue

To get a list of the chats currently in the approval queue, use HTTP GET.

Approving or rejecting chats in the approval queue

To approve or reject chats that are in the approval queue, use HTTP PUT with the following parameters:

Table listing parameter for approving or rejecting chats in the approval queue.

Name	Type	Required	Default	Description
approved	String[]	Mandatory		A list of chat ids to be approved; may be empty
rejected	String[]	Mandatory		A list of chat ids to be rejected; may be empty

Note that both parameters are required, but each can be an empty list. If a chat does not exist or the approval process fails for one of them, processing will continue with the next chat in the list. The status returned from this API will be success if at least one approval / rejection was successful.

Community Creation Workflow

If the workflow feature has been enabled in the Sametime Advanced Server administration settings, then communities will require an additional approval step after creation. The REST URI to access community creation workflow information is:

```
/stadvancedapi/api/workflow/communityqueue
```

The properties of community creation workflow are (one entry per community in the queue):

Table listing properties of community creation workflow.

Name	Type	Element	Description
Id	String	<id>	The chat id
Title	String	<title>	The chat name
Creator	User	<author>	The person who created the chat
description	String	<content>	The chat description

Retrieving a list of communities in the approval queue

To get a list of the communities currently in the approval queue, use HTTP GET.

Approving or rejecting communities in the approval queue

To approve or reject communities that are in the approval queue, use HTTP PUT with the following parameters:

Table listing parameter for approving or rejecting communites in the approval queue.

Name	Type	Required	Default	Description
approved	String[]	Mandatory		A list of chat ids to be approved; may be empty
rejected	String[]	Mandatory		A list of chat ids to be rejected; may be empty

Note that both parameters are required, but each can be an empty list. If a community does not exist or the approval process fails for one of them, processing will continue with the next community in the list. The status returned from this API will be success if at least one approval / rejection was successful.

Licensing

The REST URI to access a license is:

```
/stadvancedapi/api/license/DEFAULT
```

The properties of a license are:

Table listing properties of a license.

Name	Type	Element	Description
Id	String	<id>	License ID
count	Integer	<tcount>	Total number of licenses
available	Integer	<available>	Number of available licenses
anonymous	Boolean	<anonymous>	If anonymous users allowed
Everyone	Boolean	<everyone>	If everyone may access this feature
description	String	<description>	Description of the license

All REST API calls identify the license using the *license* part of the URI, and follow the usual rules for REST calls.
Similarly, the REST URI to access a licensed feature is:

```
/stadvancedapi/api/license/DEFAULT/ALL
```

In order to use the licensing APIs, the caller must have the AdminUsers role in the IBM Sametime Advanced Application. The only exception to this is a GET request that does not include a user parameter.
The properties of the license feature are:

Table listing properties of the license feature.

Name	Type	Element	Description
Id	String	<id>	License feature ID
Licenceid	String	<license>	License ID
Count	Integer	<tcount>	Total number of licenses
Available	Integer	<available>	Number of available licenses
anonymous	Boolean	<anonymous>	If anonymous users allowed
Everyone	Boolean	<everyone>	If everyone may access this feature
description	String	<description>	Description of the feature

Again, manipulating the license features uses the usual REST mechanisms.
Managing the assignment of licenses can be either per user or per group of users. The feature is assigned or revoked to users and groups by means of the following parameters:

Table listing parameters for managing the assignment of licenses.

Name	Type	Required	Default	Description
group	String	Optional		

user	User[]	Optional		

Monitoring

Persistent Chat Monitoring

Monitoring of chat rooms is available for a given time span (between specified start and finish dates), with the results grouped by period (days, weeks, months, quarters or years). The REST URI for monitoring chat rooms is:

```
/stadvancedapi/api/monitoring/chats
```

Monitoring properties are provided as follows, one entry per monitoring period:

Table listing properties of monitoring.

Name	Type	Element	Description
Id	String	<id>	id
Name	String	<title>	Descriptive title for the results
Total	Integer	<sta:totalChatRooms>	Number of chat rooms (at end of period)
Active	Integer	<sta:activeChatRooms>	Number of active (ie updated) chat rooms
ActiveUsers	User[]	<sta:activeUsers>	Active users count (ie count of users who contributed entries during the period)
Entries	Integer	<sta:chatRoomEntries>	Total number of chat room entries

Retrieving monitoring information for persistent chats
To get the monitoring information for chats, use HTTP GET with the following parameters:

Table listing parameters for retrieving monitoring information of persistent chats.

Name	Type	Required	Default	Description
startDate	Date	Mandatory		Start of the monitoring period
endDate	Date	Optional	Current date	End of the monitoring period
period	MonitoringPeriod	Optional	day	Period which each monitoring entry covers
pageIndex	Integer	Optional	0	The page of results to return
elementsPerPage	Integer	Optional	100	Number of entries per results page
chatId	String	Optional		Id of the chat
folderId	String	Optional		Id of a parent folder; monitoring information is limited to chats in this folder and its subfolders

User monitoring

Monitoring of users is available, with the results grouped by user. The REST URI for monitoring users is:

```
/stadvancedapi/api/monitoring/users
```

Monitoring properties are provided as follows, one entry per user:

Table listing properties of user monitoring.

Name	Type	Element	Description
Id	String	<id>	id
Name	String	<title>	Descriptive title for the results
User	Integer	<sta:user>	The user
ChatsOwned	Integer	<sta:chatRoomsOwned>	Number of chat rooms owned
ChatsJoined	Integer	<sta:chatRoomsJoined>	Number of chat rooms joined
ChatsInMyChats	Integer	<sta:chatsInMyChats>	Number of chat rooms in My Chats

Retrieving monitoring information for users
To get the monitoring information for users, use HTTP GET with the following parameters:

Table listing parameters for retrieving monitoring information of users.

Name	Type	Required	Default	Description
userIds	String[]	Optional	Current user	Users to be monitored
startDate	Date	Mandatory		Start of the monitoring period
endDate	Date	Optional	Current date	End of the monitoring period
period	MonitoringPeriod	Optional	day	Period which each monitoring entry covers
pageIndex	Integer	Optional	0	The page of results to return
elementsPerPage	Integer	Optional	100	Number of entries per results page

Community monitoring

Monitoring of communities is available for a given time span (between specified start and finish dates), with the results grouped by period (days, weeks, months, quarters or years). The REST URI for monitoring communities is:

```
/stadvancedapi/api/monitoring/communities
```

Monitoring properties are provided as follows, one entry per monitoring period:

Table listing properties of community monitoring.

Name	Type	Element	Description
Id	String	<id>	Id
Name	String	<title>	Descriptive title for the results
TotalCommunities	Integer	<sta:total>	Total number of communities that exist at the time the request was issued
TotalNumberCommunitiesCreated	Integer	<sta:created>	Total number of communities created during the monitoring period

Retrieving monitoring information for communities
To get the monitoring information for communities, use HTTP GET with the following parameters:

Table listing parameters for retrieving monitoring information of communities.

Name	Type	Required	Default	Description
startDate	Date	Mandatory		Start of the monitoring period
endDate	Date	Optional	Current date	End of the monitoring period
period	MonitoringPeriod	Optional	day	Period which each monitoring entry covers
pageIndex	Integer	Optional	0	The page of results to return
elementsPerPage	Integer	Optional	100	Number of entries per results page

Search

Search is available for a given string across a range of resource types. The REST URI for search is:

```
/stadvancedapi/api/search
```

The atom data returned is in the format defined for the various elements (i.e., folders, chats, FAQs etc), with one entry per item found.

Performing a search

To perform a search, use HTTP GET with the following parameters:

Table listing parameters for performing a search.

Name	Type	Required	Default	Description
text	String	Mandatory		The text to search for
scope	SearchScope	Optional	all	The scope to be used for the search

.

Chapter 5. Server extension reference

It is possible to extend the IBM Sametime Advanced 9.0 platform by implementing server extensions. The following extension points are available:

User information (directory service) extension point

A plugin for this extension point may be implemented in order to provide extra user information from an additional repository whenever an LDAP query is made. Here are the extension details from the extension's plugin.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<plugin id="com.ibm.rtc.extensions"
       name="SametimeAdvancedExtensions"
       version="1.0.0"
       provider-name="IBM">

   <extension-point id="UserInformationProvider"
                    name="UserInformationProvider"
                    schema="schema/UserInformationProvider.exsd">
   </extension-point>

</plugin>
```

Providing an extension to supply additional user information

To provide an extension to supply additional user information, you must supply an implementation of the com.ibm.rtc.extensions.UserInformationProvider interface. The main details of this interface are as follows:

```
public interface UserInformationProvider {
 public UserInformation getUserInformation( final String loginId,
                                            final String vmmId,
                                            final String
emailAddress,
                                            final String displayName
);
}
```

There is a single method here to be implemented in the extension, getUserInformation(). This method passes in the login id, vmm id and email address of a user and allows the extension to return additional information in the form of a UserInformation object. The details which this object allows you to provide include:

- business address
- title
- phone number
- a picture

For further information, see the com.ibm.rtc.extensions.UserInformation class.
In addition to the extension implementation class, you must also supply a plugin.xml file which specifies the details of your extension. This should take the following form:

```
<plugin id="com.mycompany.extensions.userinformationprovider"
   name="MyUserInformationProvider"
   version="1.0.0"
   provider-name="">

   <extension
   id=" com.mycompany.extensions.userinformationextension "
      name="MyUserInformationProviderExtension"
      point="com.ibm.rtc.extensions.UserInformationProvider">

      <implementation
         class="com.mycompany.extensions.MyUserInformationProviderImpl"/>

   </extension>

</plugin>
```

Chapter 6.Compliance API

Sametime Advanced Compliance Overview

Some customers desire or require advanced control and archiving over all of the messaging and communications in their enterprise -- for example, financial institutions needing to comply with SEC regulations. The levels of policy enforcement, archiving, and auditing needed for such regulation compliance significantly exceed the base functionality included in email and unified communication software. A number of add-on solutions have been created by third party vendors which provide the additional functionality to cater to those compliance needs. The vendors use terms like "advanced IM management" when labeling their add-on products -- this spec will use the narrower term "compliance product" to describe these third-party offerings.

Although the core Sametime instant messaging product has had companion compliance products from these vendors for some time now, the components which comprise Sametime Advanced had been lacking some of the API hooks to permit this kind of integration. Sametime Advanced 9.0 adds the appropriate hooks, "compliance adapter APIs", to enable third party vendors to create the adapter pieces which will integrate their compliance products with Advanced.

Audience: this API guide is intended for developers and managers either at third party compliance product vendors interested in building compliance products for Sametime Advanced, or at institutions that create their own archiving or compliance solutions. It also serves as a general overview of the range of archiving/compliance functionality offered by Sametime Advanced.

Functional Areas of Compliance Products

The following list gives a brief description of main functions of the compliance products that require APIs from Advanced, with examples of the new additions in Sametime Advanced to meet those needs. More thorough descriptions of the API additions are provided later.

Archiving: the complete capture and reconstruction of dialogs or messages sent in the system. Advanced's tools for Broadcast Communities, like BC Announcements, previously had no archiving.

Disclaimers and User Text Customization: compliance products will optionally inject a disclaimer message stating that chat contents are archived and subject to review. Such functionality will be exposed for Advanced Persistent Chats.

Real-time Content Based Alerting & Blocking: filters can be established to alert managers and/or block entirely a communication which matches the filter -- for example, one vendor's product can be configured to block messages containing credit card or social security numbers. The new API supports this in Persistent Chats and Broadcast Community functions.

File Transfer Control and Virus Scanning: the ability to block, modify, or virus scan a file being appended to a communication. Current Advanced features permitting file transfer or posting, for example Persistent Chat rooms, have not previously had these capabilities.

Ethical Boundary Enforcement: also known by terms like "chat walls" in vendor literature, this is the ability to perform communication blocking based on the participants themselves. For example, a financial institution which has both an investment banking group and a group of securities analysts may forbid chat communications between those two groups. Both Broadcast Communities and Persistent Chat features will add API for such controls.

Some features of compliance products can be built out as additional functionality as long as basic adapter capabilities above are provided. For example, features which forward captured material into an email system or those which present usage statistics are both possible now that the basic archiving portions of the API are provided.

Assumptions

Sametime UserID

Most of the actions that need to be processed by a compliance product will need to consider the sending user requesting that action and the set of potential recipients. Where this spec indicates that such IDs are used, the Advanced server will use an LDAP attribute for each user. By default, the attribute used will be the first one assigned in the directory configuration "Login properties" field under the Federated Repositories Configuration tab, in the Secure Administration section of the Integrated Solutions Console -- this is frequently configured to be an email address.

Bootstrapping, In-process API

The connection to the compliance adapter in the Sametime Advanced server will utilize the extension point loading scheme provided by Websphere Application Server. When the Advanced server initializes, it will make a request to WAS for classes implementing the extensions, then load the returned specified adapter classes into its JVM process space and register implementations of listener APIs with corresponding event producers.

The extension loading paradigm is designed to support multiple classes implementing the same handler, however the 9.0 version of Sametime Advanced restricts its support to only one class of a handler type being loaded and used. Future versions of the Advanced server might support chaining of multiple instances of the same handler, potentially from different vendors and/or having different purposes, but this is not yet available in 9.0.

The overall set of compliance functions will be broken into a handful of Java interfaces with a naming pattern PluginXyzHandler. Implementation of interfaces is optional -- a compliance product can implement the adapter interfaces for only the actions it wishes to influence. Any interfaces not implemented are treated by the Advanced server as "everything approved".

Multithreaded and unsynchronized

The Advanced server will make multiple simultaneous calls to adapter functions (such as announcement(announcementEvent)) on separate threads; each thread is handling a different Advanced activity. It is important to understand that the Advanced server is blocking on its adapter call to effectively suspend the activity pending the compliance adapter's approval. For system performance, the adapters must handle the separate calls concurrently and ensure there is no synchronization that will serialize the accesses. Archiving calls are also implemented as blocking calls to maintain a uniform programming model; a compliance or archiving tool may need to implement additional queuing to ensure performance. The expectation is that a large fraction of activities will require minimal processing; any subset that requires more intensive processing must not hinder the completion of the others.

Adapter API - Broadcast Communities

Broadcast Communities Recap

The basic Broadcast Community is a collection of users with some common interest who wish to communicate via messages that are delivered to the collection members. The Open BC is the most liberal form of community -- anyone who is a Sametime user can join, anyone joined can initiate message broadcasts, and any message is delivered to all users who subscribe to that BC and are currently logged in. Sametime Advanced also provides three levels of community membership control: Private, Restricted Publisher, and Restricted Recipient. This spec assumes that constraints imposed by compliance products will need to work in all of the above. While we expect customers will use Private communities where ethical walls are needed, the compliance API gives compliance products an overruling authority to permit or deny participation in a BC.

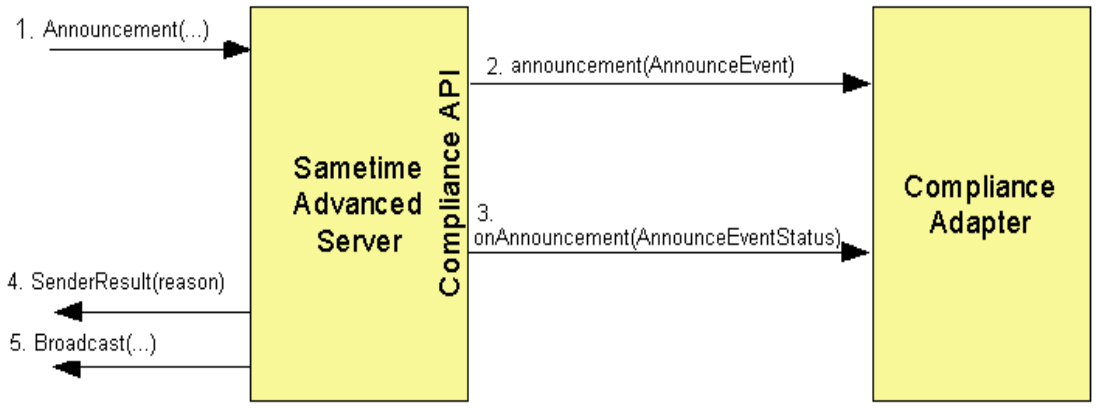
The participants in a Broadcast Community have four tools for connecting with their BC peers:
Broadcast Announcements: real-time alerts can be sent to channel subscribers
Instant Poll: subscribers can create and respond to real-time polls
Skill Tap: subscribers can find and interact with experts; expert responses can be saved and retrieved
Broadcast Chat: invite members of a broadcast community to an online group conversation

Compliance in Broadcast Communities - Use Case and Basic Architecture

An Announcement sent in an Open BC is the most fundamental operation exercising the BC pub-sub infrastructure. Any member of the BC can create the announcement, and on publication the other users who subscribe to that community and are also currently logged in will be shown an alert dialog containing the announcement text. The compliance product has two concerns for the operation: whether the content of the broadcast violates established corporate policy, and for possible future audit, which users received the message.

A third general concern is ethical wall enforcement – which users should be permitted communication via the BC. This is not handled on a per-message basis but rather at the community subscribe level and will be discussed shortly. In this first example, therefore, all of the users logged into the BC have passed any ethical wall test and are permitted to communicate with each other.

The figure below outlines the basic architecture and flow for a broadcast operation through the compliance APIs for Bcs.



The initiation of an Announcement in a BC is handled in the Advanced system as a publish event on the BC publish-subscribe bus. The first objective of the compliance adapter API then is to interrupt the

process before that broadcast, permitting an attached compliance product an opportunity to modify the message or to block the broadcast altogether. The compliance adapter can also submit an informational message which will be sent back to the user who initiated the broadcast.

The following details apply to the steps shown in the figure:

1. The user-initiated Announcement function arrives as a Publish command for the BC pub-sub bus. The parameters include message-text, sender-ID, the target BC, tool-name (Announcement). The Advanced server generates a GUID for this message and assigns it as message-ID.

2. The compliance API checks to see whether a compliance adapter is registered which handles announcements (PluginBroadcastHandler), and, finding one, forwards the details of the Announcement as as a call to the method **announcement(announcementEvent)** on that adapter. The Sametime Advanced server will withhold the broadcast from target recipient users until interactions with the compliance adapter have completed. The announcementEvent encapsulates the details of the incoming request and also the results from the compliance adapter in writable fields like status and reason. The adapter can potentially modify the text of the announcement or block the action entirely. The event fields and the server behavior are discussed in more detail shortly.

The next steps will always be after step 2, but may be re-ordered with respect to each other as needed to permit performance tuning or better user experience. Future versions of the compliance API may permit multiple adapters, in which case all adapters will complete step 2 before the the API will enter these subsequent steps.

3. The compliance API will call out to the adapter a second time to indicate the final result -- in this case, to the onAnnouncement(announcementEventStatus) method. The general pattern is that the xxEventStatus object contains the same details as the xxEvent object, but they are now immutable in the EventStatus version. This call is made to allow the compliance adapter to archive the final result details and reflects the actual deliveries to be performed in steps 4 and 5.

4. If there is any message which the compliance adapter wants to echo back to the originating user (set in step 2), it is delivered back to that user's client. Depending on the particular action, there may be data packets sent to multiple users and filtered at the client to only be shown to the one user -- in other words, if there is content which has been redacted because it is deemed inappropriate to broadcast, then for maximum security that content should not be included in the reason message.

5. If the broadcast action is permitted by the adapter consulted in step 2, then the actual broadcast will be sent to recipient users, with any modifications specified by the adapter in step 2.

Details of the announcementEvent and resulting Advanced behaviors

announcementEvent fields
- originatingUser: identity of the user that initiated the action, an object with two subfields: ID (String) and isAnonymous (boolean)
- sessionID: broadcast community ID
- requestID: a unique ID assigned to this requested action
- text: initially the text entered by the user, but writable to permit adapters to modify for compliance
- status: writable field for compliance result, one of EventStatus.EVENT_OK_TYPE, EVENT_MODIFIED_TYPE, or EVENT_BLOCKED_TYPE

reason: writable string field where compliance adapter inserts a message which will be displayed only to the originatingUser
sessionDetailUtil: a utility class for all events for retrieving name and user lists

Note that in version 9.0 BCS users are never anonymous, but Persistent Chat does permit anonymous users. The same User object is used for all events, so the ID of the user attempting the broadcast is contained in getOriginatingUser().getId().

An instance of the SessionDetailUtil class is accessible from all events and can provide details about the Broadcast Community's (or Persistent Chat Room's) name and users . In 9.0, the sessionID of a BCS is based on the user-specified community name – however, future versions of Advanced may move to a GUID, so the recommended practice is to use the SessionDetailUtil.getName(). Also, in 9.0 the history of BCS users is **not** maintained in the Advanced server, so getHistoricalUsers is unimplemented for broadcast communities; a compliance product would need to maintain that history if their compliance options relied upon it. Finally, a SessionDetailUtil object obtained from a BCS event cannot be used for Persistent Chat details and vice versa.

The message-text field is originally populated with the content submitted by the user, but can be written with a modified value by the compliance adapter. If the compliance adapter wants to enforce a text modification the adapter must set the status field to MODIFIED.

At the conclusion of step 2, if the status is MODIFIED, then the Advanced server will inspect the other event fields and react as follows:

text: The text field will be read and used as the body of the announcement broadcast to the other users
reason: if non-null, the string is echoed back to the user who originated the action -- may be in a pop-up dialog or incorporated into other existing UI. Messages here may include indications of modifications. **null** indicates that no message should be sent to the originating user

If an adapter has determined that the action should be blocked, the status must be set to EVENT_BLOCKED. The reason field is inspected and follows the same behavior as the modified case -- if non-null, it is echoed back only to the originating user (step 4). Step 5 is eliminated because of the BLOCKED status -- no transmission of the original broadcast is made.

At the conclusion of step 2, if the status remains EVENT_OK then the Advanced server will ignore the other fields for its processing. The original text that had been submitted is broadcast to all eligible users in the BC, the same as if no adapter was present for the action.

Some compliance products can be configured to send a general notification – for example, a notice that usage of the tool is monitored and subject to review, which is sent on a user's first usage of the tool each day. This kind of message can be sent to an originatingUser by setting the reason field and setting the MODIFY status -- the text field will be used by Advanced for the broadcast but could be left unchanged by the adapter for this kind of general notification purpose.

Enforcing Ethical Walls in Broadcast Communities

A UserJoinEvent is created and sent to the PluginBroadcastHandler.userJoin() method for each community to which a user is establishing a new subscription. When a user is logging into a new session of the Sametime client, a new subscription is established for each community that they had

already chosen in the list of BCs in their Broadcast Communities panel. Any new community that they join during a session – adding a new community to their list – will also create a userJoin call.

The compliance tool can set the status field of the event to block the user from joining – this might be done to enforce an ethical wall. A reason message should be provided in the case of this block, so that the user understands their attempts to use the community will not function.

A second call to onUserJoin is made which reflects the decision of the adapter and provides an archive opportunity for tracking which users are in the room. When a user leaves a community either by dropping one from their list or simply logging out, another call is made to onUserJoin with eventType set to USER_LEAVE_EVENT_TYPE so that the exit can be tracked.

For performance purposes, only the first subscription attempt during a session is presented to the userJoin function for the opportunity to block. When the user logs in, both the userJoin and onUserJoin will be called. If, in the course of that session, the user drops a community from their list and then rejoins that same community in the same session, the onUserJoin will be called to reflect a drop and another join, but the userJoin (with the opportunity to deny entry) will not be called again.

Tool-Specific Functions and Consideration

Announcements: the basics section above covers all details concerning Announcements sent on a BC as long as the option to also start a Broadcast Chat has not been selected. If the user has enabled the option "Start a Broadcast Chat in addition to sending the Announcement", the compliance API will present the requested action to the compliance adapter as a Broadcast Chat request.

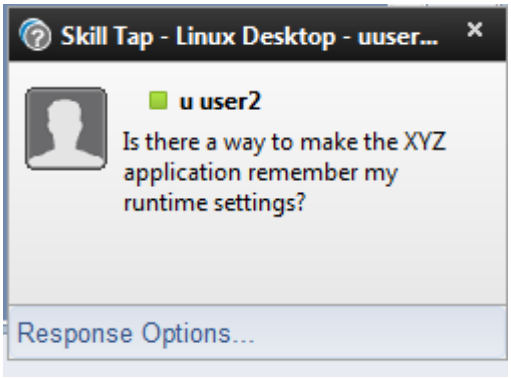
Broadcast Chat: the Sametime Advanced server is involved in the initial distribution of the invitation, and the request to send this will be handled much like the basic Announcement: the request details are passed. After the invitation broadcast has been made, the Advanced server is not involved with the Join action of the individual users – these are handled in the Sametime system as standard N-Way chats, and a compliance product would need to handle them as such.

Instant Poll: the initiation is similar again to the basic Announcement except that the choices on which the recipients will vote are included and demarcated. The InstantPollEvent has a String field for the question, and a String array field for the answer set. In both cases the compliance adapter has the opportunity to revise the text of either, or block the action entirely.

The connections established for both the voting and the result-checking will not be subject to compliance adapter delivery constraints-- the compliance product needs to produce its block in the first stage before the initial message is sent (any user that can see the question can vote and see results).

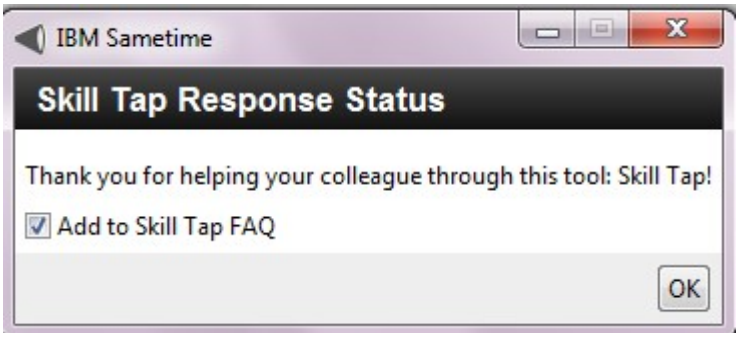
Skill Tap: the Skill Tap feature has two main function areas: a broadcast function which starts with a question being sent and permits recipients to reply, and a set of FAQ functions which allow FAQs for the Broadcast community to be created, viewed and edited. These areas have some overlap, because the final stage of the broadcast activity allows the answering party to submit their answer as a FAQ, but other FAQ activity like editing and deleting can be performed later, independent of any broadcast.

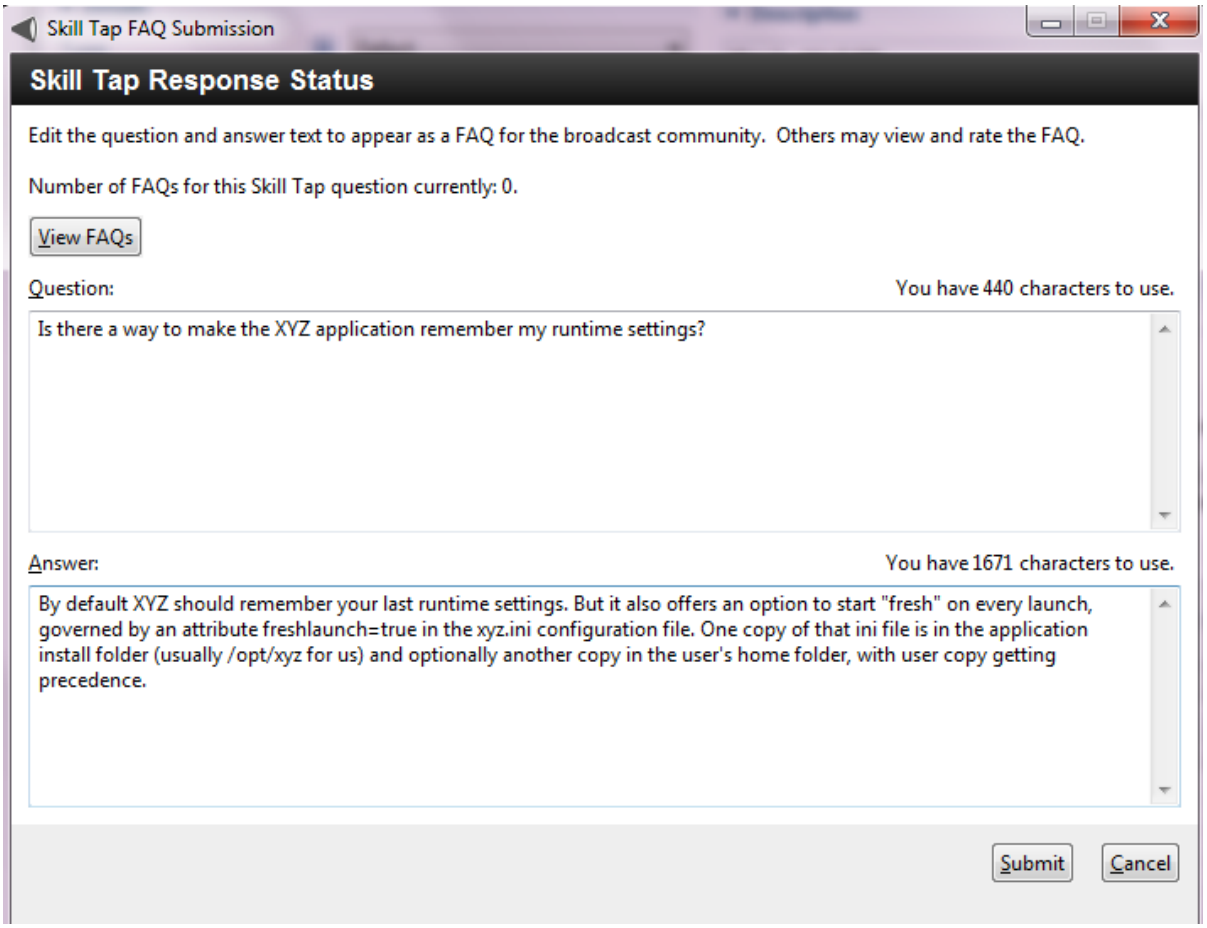
Starting with the broadcast activity, a member of a BC initiates a Skill Tap by asking a question that will be sent out to all logged in recipients. This broadcast of the initial question is identical to a basic Announcement from a compliance perspective, with the same concerns -- text content and the set of permitted recipients. The same details sent to the compliance adapter for Announcement processing will also be sent for a Skill Tap initiation broadcast. The first picture below shows the recipients' view of an alert bubble notification initiating a Skill Tap:



The second picture shows the control dialog presented to a recipient if they press the Response Options button on the notification, and this permits the recipient to enter a second stage of the Skill Tap, live communication between the asking party and this recipient. The two live communication options, "Individual Response" and "Broadcast Chat" are forms of standard chats, and therefore the opportunity to govern or archive must be handled by the compliance product attached to the Sametime chat (community) server. The "Update Status" and "FAQ Watch" functions will not convey new text and are not routed into a compliance adapter.

If an answering user has responded, when that user closes their chat they will be asked whether they would like to submit their answer as a FAQ for the community; the first picture shows the dialog that presents this, and the second picture is the FAQ creation window opened if the user chooses that option:





The FAQ creation window permits the answering user to modify the text of the question as well as the answer, and then store these into a database of FAQ entries for that community. Because the text is open for editing, the Advanced server will call a compliance adapter with the ID of the answering user plus both text pieces to permit the compliance adapter to make modifications or block the entry entirely. Any FAQ creation capability in Advanced will be likewise sent to the compliance adapter for processing.

BC Compliance in Clustered Deployments

In a clustered Advanced deployment, each user logging into Advanced Broadcast Communities will create a set of subscriptions that are maintained at one node of the cluster. All of those subscriptions are at a single node – the user is effectively logged into the BCS feature at a single node. When a user performs a publish action to a BC such as sending an Announcement, the BCS software will send the announcement to other users in the community also on that node, and will also forward a message to other nodes so that the publish can be sent to users on those nodes also in the community.

In the case that a compliance product is in use, it would be deployed on all the nodes in the system. During the publish activity, the compliance API will call the adapter at the node on which that user's subscriptions reside. The results of that compliance call are put into effect before the message is sent to the other nodes, as well as to the users on the local node. The adapters on the other nodes are not notified of the action, only the adapter on the node where the user is logged in.

Adapter API - Persistent Chat

Persistent Chat Recap

The Persistent Chat room is a group chat which records the transcript of messages posted by users and displays the most recent of these when a user enters the room. The whole transcript can be accessed, and search tools are supplied to help find entries from a specific time or user, or with a specified search string. The Persistent Chat permits users to post files, which are archived by the server and can be downloaded by any user visiting the room.

Compliance tools will want to control access in the same way that they can for regular chats, but the consideration for chat wall constraints must extend to those users that have been in the room in the past.

Note that even in clustered deployments, an instance of a Persistent Chat room is anchored to a single node. There is no need for the compliance events to be propagated from node to node.

Compliance in Persistent Chat

The retention of transcript data in the Persistent Chat room is a significant difference from the Broadcast Community, and in order to preserve chat wall constraints, a room must become off-limits to a user from a chat wall constrained group as soon as a user from the other side of that chat wall has entered the room. A compliance tool will need the appropriate hooks to enforce that join restriction. Even in cases where chat wall restrictions are obeyed, a compliance tool will potentially supply some of the same functions used in BCs and regular chats, such as content-based filtering, posted file controls, and their own transcript archiving.

Common Interfaces

Like the Broadcast tools, the Persistent Chat APIs will use two communication phases to the adapters, following the pattern action(actionEvent) for a decision phase, followed by onAction(actionEventStatus) for reporting the final result for archiving.

The xxEventStatus has the same fields as the xxEvent, but those in xxEventStatus are immutable. Like the BC versions, all include status and reason fields.

PersistentChat permits anonymous users, so a simple User object is introduced which has an isAnonymous flag. A second field, userId, will be a system user ID for regular users, but for anonymous users it will instead be the arbitrary display name entered by the guest when they joined the room.

A PluginException class wraps exceptions generated during processing -- it extends Exception and adds no function, simply calling super() with any message and/or cause parameters supplied.

The SessionDetailUtil class will give user history for a persistent chat room:

List getCurrentUsers(String sessionID) throws PluginException- returns a list of User objects for users who are currently in the chat room

List getHistoricalUsers(String sessionID) throws PluginException - returns a list of User objects, returning all users who have ever been in the chat room

String getSessionName(String sessionID) returns the name of the Persistent Chat Room

Join Permission: Persistent Chat rooms can already be set to permit only certain users or groups to join, and any incoming action to join a room will go through an access control calculation in Sametime Advanced before permission is granted. The compliance adapter API functions involved are:

PluginUserJoinHandler

void userJoin(UserJoinEvent event) - called when a user is trying to join a chat room

void onUserJoin(UserJoinEventStatus) - called when a user is joining or leaving a chat room

Some compliance vendors inject disclaimer messages when a user joins the room (e.g. "Note business guidelines apply to chat room use, and room content is actively monitored for compliance"). This can be achieved by setting status MODIFIED and putting the disclaimer text into the reason field.

Note that the 9.0 version of Sametime Advanced will not consult the compliance adapter when calculating lists of Persistent Chat rooms shown to users browsing or searching available rooms – we expect room managers will use the existing Sametime ACL functions which will filter those results. Still, the compliance adapter will always be consulted and have final say for the user trying to enter a room.

PChat Archiving and Filtering: Sametime Advanced does offer retrieval of transcript data, however some compliance vendors offer features that create "tamper-proof" logs of communication activity which require a real-time echo of every transcript entry out to the compliance API at the time it is made. Also, permitting the compliance product to inspect and possibly block or modify a transcript entry requires that the line be presented to the compliance API and that the Advanced server withhold publication to the room until all compliance adapters have returned their judgement. These functions are combined in the same two-phase way as the BC case above.

PluginChatTextHandler

void chatText(ChatTextEvent event)

void onChatText(ChatTextEventStatus)

The ChatTextEvent and ChatTextEventStatus include a text field initially populated with the user's attempted post text. If status is set to MODIFIED, the revised text that has been written into the

event field will be used for the final post to the room. For MODIFIED or BLOCKED the reason field is echoed to originatingUser.

PChat File Posting: these functions permit an adapter to log and manipulate files posted to the Persistent Chat room -- manipulation can include virus scanning or other inspection, and the interface permits the adapter to return a potentially modified file or block the post entirely. The transaction is similar to the text archive/filter just above.

PluginFilePostHandler
    void filePost(FilePostEvent event)
    void fileDelete(FilePostEvent event)
    void onFilePost(FilePostEventStatus)
    void onFileDelete(FilePostEventStatus)

The FilePostEvent.getFile() returns a java.io.File object -- actually a path+filename descriptor to a file on a system (e.g. on disk). If a modification is needed, the adapter should overwrite the file and set the MODIFIED status. A separate set/getFileName is used for the file name as it will be presented in the UI for the room -- this is subject to compliance checking like any text. Note that the filename portion of the getFile is not equal to getFileName (the former is mangled with roomID and GUID).

FAQ Creation, Edit, Delete: the Question and Answer fields are available for inspection, and may be modified by the compliance adapter in the create and edit cases.

PluginAddFAQHandler
    void addFAQ(AddFAQEvent event)
    void editFAQ(AddFAQEvent event)
    void deleteFAQ(AddFAQEvent event)
    void onAddFAQ(AddFAQEventStatus)
    void onEditFAQ(AddFAQEventStatus)
    void onFAQDelete(AddFAQEventStatus)

Room and Transcript Deletion: PChat can offer deletion of portions of a transcript, or an entire room. Transcript deletions are based on start and end times; times are not modifiable, but the action can be disallowed.

PluginChatDeleteHandler
    void chatRoomDelete(ChatRoomDeleteEvent)
    void chatTextDelete(ChatTextDeleteEvent)
    void onChatRoomDelete(ChatRoomDeleteEventStatus event)
    void onChatTextDelete(ChatTextDeleteEventStatus)

Adapter Initialization

The Advanced adapter handling will follow the scheme established by the Sametime Gateway Server for initialization. An adapter registered for a given extension point must call pluginStarted to signal it is ready to accept operations.

The interface includes Properties objects which in 9.0 are left null – future versions of Advanced will potentially add integration functionality which will pass properties from Sametime

configuration databases to the adapters. For 9.0 the adapters will need to deploy their own configuration utility – for example, and editable properties file which is copied to the server and read by the adapter on initialization.

PluginRegistration (package com.ibm.collaboration.realtime.plugin)
- void init(Properties customProperties, PluginStartupCallback callback) - the custom properties object is unused in 9.0 and will be null. The callback should be immediately called when the plugin is ready to be started.
- void customPropertiesChanged(Properties newProperties) - currently unused. Will be called once custom properties are implemented.
- void terminate() - not implemented in 9.0, reserved for future use

PluginStartupCallback (package com.ibm.collaboration.realtime.plugin)
- void pluginStarted(String pluginName)

Internationalization

Localization of result-text is left to the compliance adapter configuration. In the 9.0 version, Sametime Advanced does not have per-user or per-action locale information available to forward to the adapter.

Chapter 7. Samples

This section describes the samples that are included in the SDK. These are working examples, designed to illustrate the various functionalities of IBM Sametime Advanced 9.0 in a variety of environments.

J2EE Customer Support Call Management Application - MetaSoft Customer Support

The sample support call management application (MetaSoft) has been provided to illustrate how a J2EE application can be integrated with the Sametime Advanced SDK.

Overview

Metasoft -support call management application has been implemented as a J2EE application. This application supports the following concepts:

- A *support call*: a call with a state (OPEN, ASSIGNED, WAITING FOR CUSTOMER, CLOSED), history (plain text), contact (the *contact person* in the client *company*), assignee (person working on the call), general technical area (the technical area the problem belongs to), technical sub-area and a follow-up date.
- A *contact person*: a customer who can open support calls. It has a name (first, last), phone number, email and belongs to a company
- A *company*: a client company. Attributes: name, address, main phone number, website url, geographic area (AM, EMEA, AP), country, and the name of their commercial rep in this company.

Integration Points into Sametime Advanced SDK

This section describes the integration point with the Sametime Advanced SDK, how and where it can be used to leverage the STA features in the J2EE customer support call management application.

A chat room & community will be associated with every technical area:

- Adding / suppressing a technical area will trigger the creation / deletion of the associated chat room & community.
- The chat rooms for technical areas will be organized in folders: one main folder for all technical areas, one sub-folder for the main technical area, and one sub-folder for the sub-area; an example could be /Technical/UI/Accessibility
- Support engineers will have the possibility to see the latest updates to the chat room, make an update (ask a question)
- They will also have the possibility to send an alert with a "hot" question - i.e. when a customer is on th e phone with an urgent problem and needs an answer quickly ...

A chat room & community will be associated with every client company.

- Adding / removing a client company in the company panel, will trigger the creation / deletion of the associated chat room & community.
- The chat rooms for client companies will be organized in folders: one main folder for all client, one sub-folder for the region (AM, EMEA, AP), and one sub-folder for the country (i.e. Customers/EMEA/Germany/BASF). Folders may have to be created if a company in a new country is added.
- Support engineers will have the possibility to see the latest updates to the chat room, make an update (ask a question)

- They will also have the possibility to send an alert with a "hot" question - i.e. when a customer is on the phone with an urgent problem and needs an answer quickly ...

For details on how to install and configure the MetaSoft sample application, please refer the the Readme.doc document that is packaged with the application.

LDAP Adaptor Sample

This sample extends the UserInformationProvider extension point to return user information from a Sametime LDAP. This extension point is dealt with in more detail in Chapter 5. It allows more detailed information to be retrieved from a user profile including picture data which will then be used by the Sametime Advanced Server in persistent chats. The information retrieved will appear at the top of the chat room window when a participant is selected from the participants list.
In the sample the UserInfo service is on the Sametime server is used to retrieve all information on the user. This service returns an xml document of all of the user information in the ldap. This document is parsed in the sample application and added to a UserInformation object to be returned.
To connect to the sametime user info service you need a URL in the following format:
http://hostname/servlet/UserInfoServlet?operation=3&userid=<*username*>
In the sample the hostname of the server is read in from `ldapadapter.properties.`
`Operation=3` signifies that we want to retrieve all information available about the user. This operation returns a XML document containing the business card of the requested user. The XML document takes this form:

```
<?xml version="1.0" encoding="UTF-8"?>
   <userinfo>
         <user id="joe soap/">
           <field name="MailAddress" type="text/plain">soapy@mail.com</field>
           <field name="Telephone" type="text/plain">353-1-815-5680</field>
           <field name="ImagePath" type="text/plain">http://somepictureURL</field>
           <field name="Company" type="" error="UNAVAILABLE"/>
           <field name="Name" type="text/plain">Joseph Soap</field>
           <field name="Location" type="" error="UNAVAILABLE"/>
         </user>
</userinfo>
```

The sample application then parses the xml and adds the properties to a UserInformation object.
The application can be run from the command line or added as a plugin to the Sametime Advanced server.
To extend the UserInformationProvider extension point you need to implement the extension point interface... *com.ibm.rtc.extensions.UserInformationProvider*.
The interface contains only one implementable method public UserInformation getUserInformation(final String loginId, final String vmmId, final String emailAddress, final String displayName). You can then connect to any Ldap you wish to retrieve your data from. You also must include a plugn.xml in this form:

```
<plugin id="com.mycompany.extensions.userinformationprovider"
   name="MyUserInformationProvider"
   version="1.0.0"
   provider-name="">
   <extension
   id=" com.mycompany.extensions.userinformationextension "
      name="MyUserInformationProviderExtension"
      point="com.ibm.rtc.extensions.UserInformationProvider">
      <implementation
         class="com.mycompany.extensions.MyUserInformationProviderImpl"/>
   </extension>
```

```
</plugin>
```

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

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.