Sametime
Version 8.5.2 IFR 1

Sametime 8.5.2 Interim Feature Release (IFR) 1 Software Development Kit

Meetings Compliance API Guide

Edition Notice

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

This edition applies to version 8.5.2 Interim Feature Release (IFR) 1 of IBM Lotus Sametime (program number 5724–J23) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

This document will outline events that can be captured and/or modified in a meeting room by implementing the interfaces provided.

Compliance Adapter Overview

In order for certain customers to deploy ST Meetings, we must meet a requirement for enhanced control over certain meetings features. The most obvious example being financial institutions who must comply with certain SEC regulations when using unified communication software to conduct business. In such situations, it is essential to provide the ability to record or log actions by participants – and in some cases block these actions

Most customers are already working with third party products that enforce the regulations mentioned above – it is our responsibility to provide a hook into ST Meetings for these third party products.

Main requirements

- The ability to block meeting entry based on who is already there.
- The ability to log meeting entry/exit
- Logging the start/stop of all sharing functions.
- Logging of all polls
- Logging of all sidebar tool content.
- The ability to redact or block the sidebar tool submitted content

Summary of Components

This section briefly describes some of the key components discussed in the previous section.

Meeting Interfaces and Events

Note**: This section describes some the classes/interfaces used in this API. Please see the accompanying javadoc for a complete listing/description.

Interfaces define a set of methods and parameters are passed to each method as a specific event class. The following table lists the interfaces and events:

Nearly Identical to the ST Advanced (PersistentChat) Adapter API – the Meetings version 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 (Please see the section labeled “Adapter API - Persistent Chat” in the ST Advanced spec.). The main differences will be events that can be logged/blocked will be meeting specific. The adapter interfaces will be:

- **PluginUserHandler**: Will allow the decision to be made on who can enter a room based on the users already there (but not limited to this). This will also be used for logging the user entry (as well as user leave) and any decision that was made in the first phase.
- **PluginLibraryHandler**: This adapter will allow the inspection of library entries (and deletions) so that they can be logged. Furthermore, this will also be used for when a library item or user's screen is shared.

- **PluginPollSentHandler**: Allows the inspection of the question being sent so that it can be logged + blocked based on the originator and the content of the question itself.
- **PluginChatSentHandler**: Allows the logging, filtering, and blocking of chat entries based on content and/or user.

Table listing features and associated interfaces, events, and event status

Feature	**Interface**	**Event**	**EventStatus**
User Join	PluginUserHandler	UserEvent	UserEventStatus
User Leave	PluginUserHandler	UserEvent	UserEventStatus
Chat	PluginChatSentHandler	ChatSentEvent	ChatSentEventStatus
Poll	PluginPollSentHandler	PollSentEvent	PollSentEventStatus
Library Item Shared	PluginLibraryHandler	N/A	LibraryEventStatus
Library Item Removed	PluginLibraryHandler	N/A	LibraryEventStatus
Library Item Added	PluginLibraryHandler	N/A	LibraryEventStatus

Interfaces can contain two types of methods:
- An action method which performs the action
- A corresponding response method which provides the status of the action.

For example, the **PluginUserHandler** interface contains a **userJoin(UserEvent subEvent)** as well as a matching **onUserJoin(UserEventStatus subEvent).** Action methods are passed an **Event** which contains all relevant information including an **EventStatus**. The EventStatus can be set to a failure code which allows a plug-in to block or modify the meeting event.

The PluginRegistration defines the message handler plug-in life cycle and is extended by all of the PluginHandler interfaces described above.

Some interfaces, however, only expose the onAction(actionEventStatus) method. This prevents the action from being blocked or modified. This will mainly be used for events that are meant to be logged/retained only. One example of this is the **PluginLibraryHandler** interface. This interface only exposes an **onLibraryEvent(LibraryEventStatus)** method. This method will be called whenever a library item is added, removed, or shared, but will not allow them to be blocked.

As noted earlier, each method takes a parameter that extends **EventStatus.** These classes contain fields specific to the type of event. Using the **onLibraryEvent(LibraryEventStatus)** method as an example again, you can see that it takes a **LibraryEventStatus** parameter.

```
/**
 * Used for the addition of documents and urls to a meeting, as
 * well as sharing.
 *
 */
public interface LibraryEventStatus extends EventStatus
{
        /**
         * This method is used to get the name of an item that has been added or removed
         * from the library.  These items may be files/documents or urls.
         * @return The name of the document or url
         */
        public String getEntryName();

        /**
         * Method used to determine if the item was removed/deleted.
         * @return true if the entry was deleted, false otherwise.
         */
        public boolean isRemoved();

        /**
         * Method used to determine if an item is being shared.  The item being shared may be
         * a document, url, or the user's screen.
         * @return true if an item is being shared, false otherwise.
         */
        public boolean isShared();
}
```

(To see the methods on the other **xxxEventStatus** interfaces, please see the accompanying javadoc)

The base **EventStatus** interface contains common info for all events. It also provides access to utility classes that enable info to be obtained on individual users as well determining the users who are in a meeting room:

```
public interface EventStatus
{

        public static final int EVENT_OK_TYPE = 0;
        public static final int EVENT_MODIFIED_TYPE = 1;
        public static final int EVENT_BLOCKED_TYPE = 2;

        /**
         * Every requested action is assigned a unique ID before plugins
         * are called, available by this getter
         * @return unique ID for the request
         */
        String getRequestID();

        /**
         * Getter function for the sessionID - e.g. the unique ID for the
         * Meeting Room
         * @return the SessionID for which the action is requested or applied
         */
        String getSessionID();

        /**
         * The current status of the event, e.g. EVENT_OK_TYPE,
         * EVENT_MODIFIED_TYPE, or EVENT_BLOCKED_TYPE
         * @return the event status
         */
        int getStatus();

        /**
         * Returns the type of event e.g. EventType.CHAT_TEXT_EVENT_TYPE
         * @see com.ibm.collaboration.realtime.event.EventType
         * @return the event type
```

```
         */
        int getEventType();

        /**
         * If plugins participating in the API block or modify an action, they can
         * also specify a reason for doing so which will be echoed back to system
         * users.
         * @return the reason supplied by a plugin for a block or modify, null if
         * left unset by plugins
         */
        String getReason();

        /**
         * @see
         * @return the User that initiated the action to which this event refers
         */
        User getOriginatingUser();

        /**
         * Accessor method for the singleton MeetingRoomUtil, which can be used to
         * query details about a gived session such as currently present users.
         * @return the utility class for inspecting room details
         */
        MeetingRoomUtil getMeetingRoomUtil();

}
```

Implementing Meeting Server Compliance Plugins

This section describes how to implement, package, and deploy a Meetings Compliance plugin. This section will show sample code that implements all of the available plugins.

Extension Registry Framework – How Compliance Plugins Are Discovered

The Eclipse extension registry framework, used by WebSphere Application Server, follows a plug-in system similar to a standard Eclipse plug-in. Applications are extensible by plug-ins that plug into the application's declared extension points.

The plugin.xml file, which is part of the plug-in deployment files, defines how the plug-in works with the platform runtime. This file should be located in the same package as the extension points' source code.

You can find additional information about the WebSphere Application Server implementation at: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cweb_extensions.html

Find additional information about the Eclipse Plug-in Architecture at: http://www.eclipse.org/articles/Article-Plug-in- architecture/plugin_architecture.html

Write the plugin.xml file

In order to be identified as a plug-in used by the Sametime Meeting server, the plug-in code is referenced as an extension point in the plugin.xml file. The plugin.xml file identifies the plug-in to the

environment, and what extension points it extends.

The following table lists some of the parts of the plugin.xml file and their function.

Table listing the parts of the plugin.xml file

PART	**FUNCTION**
<plugin>	Identifies the plug-in
id	the name of the primary package
name	A text name
version	Version number for this plug-in.
provider-name	Identifies who created the plug-in
<extension point>	Declares the point that is being extended.
id	The extension needs a unique ID
name	A name for the extension point you are creating.
<action class>	Within the extension tag we then specify a <action> tag with a single "class" attribute. This points to the class we defined above and tells the environment that this should be instantiated and used

The extension points for the compliance API are as follows:

- *com.ibm.meeting.compliance.plugin.user_join_handler*
- *com.ibm.meeting.compliance.plugin.poll_sent_handler*
- *com.ibm.meeting.compliance.plugin.chat_sent_handler*
- *com.ibm.meeting.compliance.plugin.libaray_event_handler*

Here is what a plugin.xml should look like for a project that extends all of the extension points above:

```
<plugin id="com.ibm.meeting.compliance.sample"
        name="Sametime Meetings Sample Compliance Plug-ins" version="1.0.0"
        provider-name="IBM">
        <extension point="com.ibm.meeting.compliance.plugin.user_join_handler"
                id="sampleJoinHandlerPlugin" name="Sample Join Handler">
                <action
                        class="com.ibm.meeting.compliance.sample.SampleUserHandler">
                </action>
        </extension>
        <extension point="com.ibm.meeting.compliance.plugin.poll_sent_handler"
                id="samplePollSentHandlerPlugin" name="Sample Poll Sent Handler">
                <action
                        class="com.ibm.meeting.compliance.sample.SamplePollSentHandler">
                </action>
        </extension>
        <extension point="com.ibm.meeting.compliance.plugin.chat_sent_handler"
                id="sampleChatSentHandlerPlugin" name="Sample Chat Sent Handler">
                <action
                        class="com.ibm.meeting.compliance.sample.SampleChatSentHandler">
                </action>
        </extension>
```

```
	<extension point="com.ibm.meeting.compliance.plugin.library_event_handler"
		id="sampleLibraryEventHandlerPlugin" name="Sample Library Event Handler">
		<action
			class="com.ibm.meeting.compliance.sample.SampleLibraryEventHandler">
		</action>
	</extension>
	<requires>
		<import plugin="com.ibm.collaboration.realtime.plugin" />
	</requires>
</plugin>
```

Continuing on with this example, the class that implements the *com.ibm.meeting.compliance.plugin_user_join_handler* extension point would look something like this (note** - this is taken from the sample code that is provided with the API):

```
public class SampleUserHandler implements PluginUserHandler
{

	private final static String PLUGIN_NAME =
		"com.ibm.meeting.compliance.sample.sampleJoinHandlerPlugin";

	public void onUserJoin(UserEventStatus status)
	{
		String sessionId = status.getSessionId();
		int stat = status.getStatus();
		String uId = status.getOriginatingUser().getId();
		String ip = status.getOriginatingUser().getIp();
		String meetingRoomName = null;
		String meetingRoomOwner = null;
		String callInfo = null;
		boolean isManager = false;

		try
		{
			meetingRoomName = status.getMeetingRoomUtil().getMeetingRoomName(sessionId);
			meetingRoomOwner = status.getMeetingRoomUtil().getMeetingRoomOwner(sessionId);
			callInfo = status.getMeetingRoomUtil().getMeetingCallInfo(sessionId);
			isManager = status.getMeetingRoomUtil().isManager(sessionId, uId);
		}
		catch(PluginException e)
		{
			e.printStackTrace();
		}

		if(status.getEventType() == EventType.USER_JOIN_EVENT_TYPE)
		{
			System.out.println("OnUserJoin user[" + uId + "] ip[" + ip + "]" + "session[" +
				sessionId +"]" + "room name[" + meetingRoomName + "]" + "Result[" + stat +
					"] reason[" + status.getReason() + "]");

			System.out.println("room owner[" + meetingRoomOwner + "] call Info[" + callInfo +
				"] is manager[" + isManager + "]");
		}
		else
		{
			System.out.println("OnUserLeave user[" + uId + "] ip[" + ip + "]" + "session[" +
				sessionId +"] Result[" + stat + "] reason[" + status.getReason() + "]");
		}
	}

	public void userJoin(UserEvent event)
	{
		User u = event.getOriginatingUser();
		String sessionId = event.getSessionId();
		String currId = u.getId();

		System.out.println("userJoin for u.id = " + currId + " in sess = " + sessionId );
```

```
                List<String> currentUsers = null;

                try
                {
                        currentUsers = event.getMeetingRoomUtil().getCurrentUsers(sessionId);
                }
                catch (PluginException e)
                {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                Iterator<String> iterator = currentUsers.iterator();

                System.out.println("Current Users in Session:");

                while(iterator.hasNext())
                {
                        System.out.println("User: " + iterator.next());
                }


                if(u.getId().contains("adams100"))
                {
                        event.setStatus(EventStatus.EVENT_BLOCKED_TYPE, "No Adams100!");
                }
                else
                {
                        event.setStatus(EventStatus.EVENT_OK_TYPE);
                }

        }

        public void customPropertiesChanged(Properties properties) {
                // TODO Auto-generated method stub

        }

        public void init(Properties properties, PluginStartupCallback callback) {
                System.out.println("Enter SampleUserHandler.init");

                if (callback != null)
                {
                        callback.pluginStarted(PLUGIN_NAME);
                }

        }

        public void terminate() {
                // TODO Auto-generated method stub

        }

}
```

Development Dependencies and Deployment

Jars needed for development

- meetings.compliance.jar: This jar contains the interfaces/utilities needed to create a meetings compliance adapter.
- meetings.compliance.samples: This jar contains sample adapters, including the **SampleUserHandler** in the previous section. The source code for the samples is also packaged with the API.

Location for jars containing compliance adapters:
Jars that contain sample adapters, including the jar that includes the sample adapters, need to be placed in a specific location on the Sametime Meeting Server. This location is *<Path to Websphere>/AppServer/profiles/<Meeting Server Profile>/optionalLibraries/rtc*. The locations in the path noted with <...> may have different values depending on OS, deployment type, etc. Similarly, the output for the sample adapters can be found in the server's SystemOut.log file, which can be found at *<Path to Websphere>/AppServer/profiles/<Meeting Server Profile>/logs/STMeetingServer*.

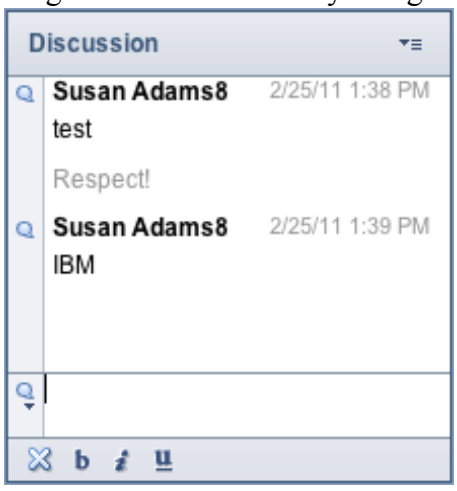
Additional Notes

- Library Events: As stated earlier, these events cannot be blocked. All library actions (adding/deleting docs and urls, sharing docs and urls) are captured as events for logging/retention only. The files that are added cannot be inspected. Further control over adding library content is available via Meeting Server policies.

- Polls: Polling questions can be inspected before they are sent out. The question can be modified or the poll can be blocked (not sent). In either case, the sender will not receive any special message indicating that the poll has been modified or blocked. However, the reason can be logged via the compliance api.

- In addition to modifying the question, the api also allows the modification of the open response flag. When sending a poll, the originator has the option to allow open ended responses. The compliance api has the ability to inspect whether or not open responses are allowed, and disable them.

- Chats: Chat entries in the discussion area can be blocked or modified. When the entry is blocked, the chat message will not appear in the discussion area and the sender will not receive any special error message. However, the reason can be logged via the compliance api. In the case where the chat entry is modified, the modified entry will appear in the discussion area. In addition to the chat entry, the sender will see a message above the chat entry that gives the reason for modification.



- In the discussion area above, the user attempted to enter a chat message that simply read “ibm”. Using the sample adapters that ship with the compliance api, “ibm” was changed to “IBM”. Also,

above that line, the sender was shown the reason for the modification. NOTE** It is up to the implementer of the compliance adapter to fill in the reason via the api.

Notices

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

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

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

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

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

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

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

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

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

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

IBM Corporation
5 Technology Park Drive

Westford Technology Park
Westford, MA 01886

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

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

Trademarks

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

IBM

AIX

DB2

DB2 Universal Database Domino

Domino

Domino Designer

Domino Directory

i5/OS

iSeries

Lotus

Notes

OS/400

Sametime

System i

WebSphere

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

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

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

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

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

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

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

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