Sametime
Version 9.0

Sametime 9.0 Software Development Kit
Remote Client API Guide

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

0.0.1 Table of Contents

1. Remote Client Toolkit Overview

This toolkit will enable third parties to integrate with Sametime Meetings by providing room creation and management APIs.

1.1 What's in the toolkit

The toolkit consists of the following directories and contents:

doc – consists of a PDF and ODT file with a general “how-to” guide.
Javadoc – java documentation of the APIs exposed
lib – a library of dependencies
sample – sample applications demonstrating the use of APIs and a sample customized login page
logging – containing a single logging.properties file for debugging purposes.
scripts – containing some scripts to set up the server.
server-project – contains the server side plugin sample necessary to tie into the server framework.

1.2 Dependencies

The dependencies are listed in the “lib” directory of the toolkit zip. On a high level, the external dependencies are:
Apache HttpClient 4.0.1
Apache Httpcore 4.0.1
Apache HttpMime 4.0.1
commons-logging 1.1
Apache mime4j 0.6
JSON4J

2. Meeting server setup and configuration

This section will provide an overview of the setup and configuration tasks that need to be performed on the meeting server before using the toolkit APIs. These instructions are very specific and may need to be adjusted depending on the customer's needs.

Note: It's important to have access to post 9.0-beta builds of the IBM Sametime 9.0 server, since some of the toolkit features depend on changes made after the beta release. Instructions on installing a meeting server can be found elsewhere.

2.1 Create a file based user repository

Most users of this toolkit will have their users registered in LDAP. However, in the case that this is not feasible, there is an option to create users in default wim file repository via the admin console or by editing fileRegistry.xml, or use AdminTask in wsadmin. It is important to note that since the Meeting server depends on the user's User Principal to be always mapped to the e-mail attribute, developers are required to alter the wimconfig.xml file, to ensure the mail attribute is first in the list of login properties.

2.1.1 Creating a user via the WAS admin console:

1. In the admin console, navigate to “Users and Groups” and click on “Manage Users”
2. Click on “Create” and enter the fields for a new user. Make sure the user has an e-mail address.
3. Edit the winconfig.xml file, to make sure the mail attribute as is first login property for the wim file repository in wimconfig.xml, this should be done on the deployment manager, not the application server, the file is typically localed like:

   ...\AppServer\profiles\STMAppProfile\config\cells\<cell>\wim\config

   The adapted file should look like:

```
<config:repositories xsi:type="config:FileRepositoryType"
      adapterClassName="com.ibm.ws.wim.adapter.file.was.FileAdapter"
      id="InternalFileRepository" supportPaging="false" ...>
            <config:baseEntries name="o=defaultWIMFileBasedRealm"/>
            <config:loginProperties>mail</config:loginProperties>
      <config:loginProperties>uid</config:loginProperties>
</config:repositories>
```

Note: In order for this change to take affect, all servers must be restated, specifically the deployment manager (dmgr), and the application server (STMeetingServer).

2.2 Setting up a custom login page:

Some integrators require their users to be automatically logged in, or to leverage SSO capabilities of the over all system solution. The URLs to Sametime meetings lead to a controller, which always redirects users to a configurable login page. It is the job of the login page, to collect the user's info (id, password, nickname, etc), and POST this data back to the room URL. The login page is required, because several mechanisms work together, for example, HTTP redirect headers, cookies, etc.

The default page, which ships with Sametime, prompts the user for either their login credentials, or their nick name. It can be referred to as an example of how to build a custom login page. For example, such a custom logon page, configured to be used in place of the default Sametime login page, could simply offer some temporary text, indicating the user is being transferred to a meeting room, which simultaneously doing a hidden/transparent POST of pre-determined user credentials or nick-name.

This next sections offer details of setting up a custom login page. The following steps are needed:

– Modify the configuration value with the new custom login page. Details about achieving this via scripting are documented below.
– Deploy the custom login page on the server.

2.2.1 Modifying the jython script to change configuration values:

The login page is a configuration value on the meeting server. It can be changed via a wsadmin/jython script attached with this toolkit (changeConfigLoginPage.py in the /scripts directory of the SDK).

Open changeConfigLoginPage.py in a text editor and search for the words 'meetingroomcenter.customLoginPage'. Modify the value field to be the custom login page.

Example: parms = ['meetingroomcenter.customLoginPage', ***'myloginpage.jsp***']

Save the file and copy it to the <Websphere install directory>/AppServer/bin.

2.2.2 Running the script:

Note: Usernames / passwords / paths will have to be changed to match your server.

From a command prompt, navigate to the <WAS install path>/AppServer/bin directory.

Run the following command:

wsadmin -conntype SOAP -host localhost -port 8880 -user **<username>** -password **<password>** -lang jython -f changeConfigLoginPage.py -wsadmin_classpath "C:\Program Files\IBM\WebSphere\AppServer\profiles\STMAppProfile\optionalLibraries\rtc\rtc.common.jar"

After the command executes successfully, the configuration will have been modified to point to the new

login page. Restart the server.

2.2.3 Deploying the custom login page sample

A custom login page sample is provided with the SDK under the "samples" directory. customloginpageSample.jsp and mock_post.jsp need to be deployed on the server under the stmeetings.webclient WAR file.

..\AppServer\profiles\STMAppProfile\installedApps\xyzmeetingCell\Sametime Meeting Server.ear\stmeetings.webclient.war

The purpose of the sample is to demonstrate joining a room without requiring any manual user input provided there are other ways of getting the user's information before-hand.

When the login page is invoked as part of joining a room, this sample reads some pre-defined, custom cookies and joins the room. The join is accomplished via a HTTP POST. The parameters that get posted are display name of the user, login Id and password (if the room has any).

The pre-defined cookies can be set by entering the information first in mock_post.jsp. The custom login page configuration value can also be set/unset via this JSP.

Note: This cookie mechanism is strictly for demonstration purposes only. It does not show the most effective way to integrate with Meetings. Integrators should devise their own mechanism for integrating their application with meetings.

2.3 Creating unlimited rooms:

Administrators may have the need to create unlimited rooms on the meeting server. Default policy limits number of rooms per user to 100. To do this, a new custom configuration key needs to be created on the server. The same procedure documented above for the custom login page can be used for this (see section on "Creating a custom login page").
The changeConfig.py script needs to be modified as follows:
Example: parms = ['meetingroomcenter.overrideAdminUserRoomLimit', ***'true'***]

To reset the policy limit, this configuration value can be set to false.

2.4 Editing policy for users:

The Sametime System Console (SSC) should be used to manage user policies for meetings. It is not possible to create or manage new policies without using the SSC. However, it is possible possible to modify the two pre-existing default policies.
This is done by pulling the row of policy data out of DB2 (there are only 2 rows for meeting server:

1. Default-anonymous policy for meetings
2. Default-authentic user policy for meetings

Each row is XML data (text). Once the rows is selected, the XML data can be edited, then re-inserted into the Database.
Since the XML sections and element names are not intuitive, annotated/commented versions Are provided as part of the toolkit API bundle.

Note: It is possible to inspect these changes by using the policy REST API URL, for example this can be inspected from a browser.

http://server.company.com/stmeetings/policy?format=json

3. Creating Custom Server-side Integration Points:

Some integration scenarios require server-side hooks, to trigger custom code on events, such as when users leave or join meetings. The meeting server provides a number of APIs, which are based on the eclipse-like, server-side plugin model. The allows a developer to create a new plug-in jar file, which implements certain interfaces, and plug into the server infrastructure via extension points or filters.

A sample eclipse project, containing both source and a .jar file, is provided in the server-project directory of the SDK. The source project "meeting.plugin.server.sample" is a full fledged Eclipse project and can be imported into Eclipse.

To test the sample, the meeting.plugin.server.sample.jar must be copied to the server's shared libraries directory -

.../profiles/STMAppProfile/optionalLibraries/rtc.

Note: Server will have to be restarted for this change to be picked up.

The prints from the sample will show up in the logs when a user joins/leaves a session on the meeting server. Prints will have to be enabled for com.ibm.rtc.sample.Sample via the WAS admin console to see this in action.

Sample code:

```
public final class Sample implements MessagingFilter
{

protected static final Logger logger = Logger.getLogger(Sample.class.getName());

/* (non-Javadoc)
 * @see com.ibm.rtc.polled.ext.MessagingFilter#isMessagingEnabled()
 */
public boolean isMessagingEnabled()
{
	return true;
}

/* (non-Javadoc)
 * @see com.ibm.rtc.polled.ext.MessagingFilter#filterMessagingEvent(com.ibm.rtc.polled.UpdateEvent)
 */
public boolean filterMessagingEvent(UpdateEvent updateEvent)
{
	// always set this to false, since we're just using this as a hook
	// and we don't want to alter meeting server's behavior
	boolean filterMessagingEvent = false;

	logger.entering(this.getClass().getName(), "filterMessagingEvent", updateEvent);

	// lets hook the event that a user is leaving a meeting room
	// first check for UserMap, all other events will be ignored
	if(updateEvent.getContainerName().equals("UserMap"))
	{
		// next make sure this is special case, where the user's key and value are the same
		String key = updateEvent.getKey();
		String value = null;
```

```
if(updateEvent.getValue()!=null)
{
	value = updateEvent.getValue().toString();
}
if(value!=null && key.equals(value))
{
	// next check that this is a removal all other operations will be ignored
	if(updateEvent.getOperation() == UpdateEvent.OP_REMOVE)
	{
		// USER LEAVING
		// this is to show the user's ID, room ID, and time of departure
		logger.info("user["+key+"] left the meeting room["+updateEvent.getSessionId()
+"] at["+updateEvent.getTimeStamp()+"]");

		// BEGIN CUSTOM CODE
		/*
		 * NOTE:  If there is more work to do here, it should be done on a worker

		 * this thread should not be blocked if there is any latent I/O, etc.
		 */
		// END CUSTOM CODE
	}
	else if(updateEvent.getOperation() == UpdateEvent.OP_CHANGE)
	{
		// USER JOINING
		// this is to show the user's ID, room ID, and time of departure
		logger.info("user["+key+"] joined the meeting room["+updateEvent.getSessionId()
+"] at["+updateEvent.getTimeStamp()+"]");

		// BEGIN CUSTOM CODE
		/*
		 * NOTE:  If there is more work to do here, it should be done on a worker

		 * this thread should not be blocked if there is any latent I/O, etc.
		 */
		// END CUSTOM CODE
	}
}


logger.exiting(this.getClass().getName(), "filterMessagingEvent", filterMessagingEvent);

return filterMessagingEvent;
}
```

4. SSL support in the client toolkit

The SDK support SSL out of the box when communicating with a server with a valid certificate. If a server has a self-signed certificate, a javax.net.ssl.SSLPeerUnverifiedException will be thrown with the following stack trace:

```
javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated
	at com.sun.net.ssl.internal.ssl.SSLSessionImpl.getPeerCertificates(Unknown Source)
	at org.apache.http.conn.ssl.AbstractVerifier.verify(AbstractVerifier.java:128)
	at org.apache.http.conn.ssl.SSLSocketFactory.connectSocket(SSLSocketFactory.java:339)
	at
org.apache.http.impl.conn.DefaultClientConnectionOperator.openConnection(DefaultClientConnec
tionOperator.java:123)
	at org.apache.http.impl.conn.AbstractPoolEntry.open(AbstractPoolEntry.java:147)
	at
org.apache.http.impl.conn.AbstractPooledConnAdapter.open(AbstractPooledConnAdapter.java:108)
	at
org.apache.http.impl.client.DefaultRequestDirector.execute(DefaultRequestDirector.java:415)
	at org.apache.http.impl.client.AbstractHttpClient.execute(AbstractHttpClient.java:641
```

```
)
        at org.apache.http.impl.client.AbstractHttpClient.execute(AbstractHttpClient.java:576)
        at org.apache.http.impl.client.AbstractHttpClient.execute(AbstractHttpClient.java:554)
        at com.ibm.rtc.client.http.HttpServiceWrapper.execute(HttpServiceWrapper.java:324)
        at com.ibm.rtc.client.http.ServiceBase.httpRequest(ServiceBase.java:364)
        at com.ibm.rtc.client.http.ServiceBase.httpRequest(ServiceBase.java:335)
        at com.ibm.rtc.client.http.ServiceBase.login(ServiceBase.java:207)
        .....
        .....
```

Note that this exception may also be thrown if the port is incorrect.

This exception is part of the regular SSL handshake negotiation procedures in the JVM and is external to the SDK. The server's self-signed certificate will have to be imported into the JVM keystore for SSL to work. Certificates can be imported using the JVM "keytool" utility. This is well documented in Oracle's Java documentation.

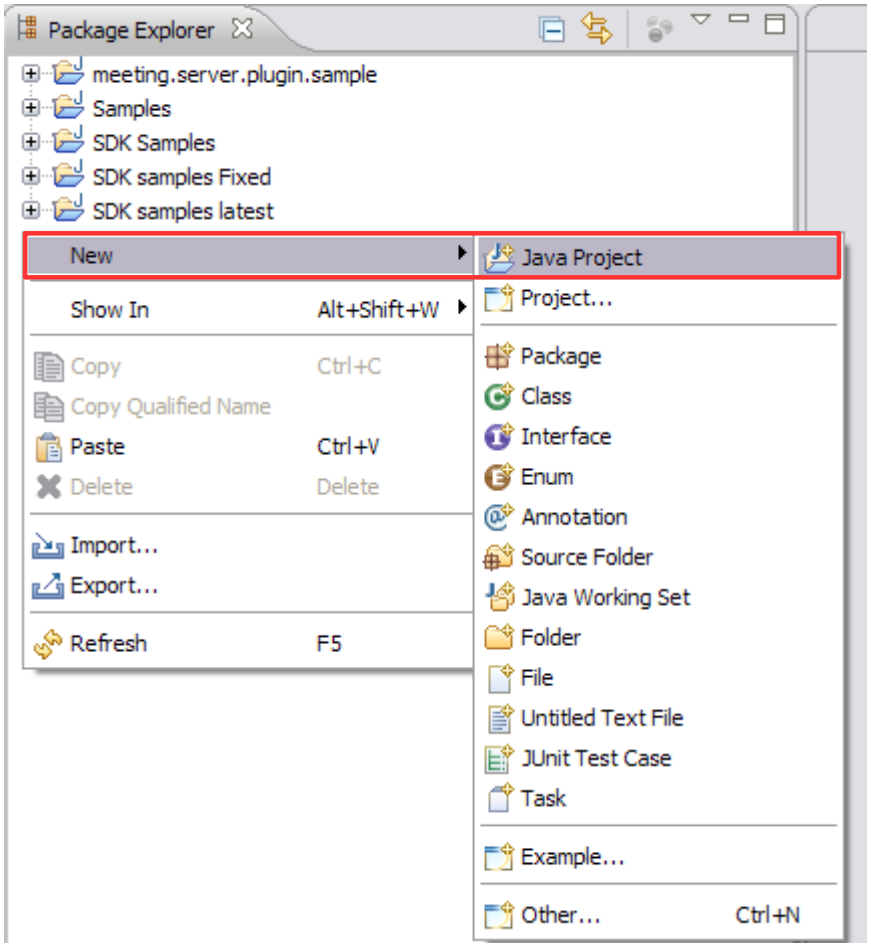
5. Samples

6.1 Running the Samples

STEP 1: DOWNLOAD SAMPLES

To download the most recent updated SDK samples please follow the link
http://www.ibm.com/developerworks/lotus/downloads/sametime_remote_client_sdk.html?ca=dre-

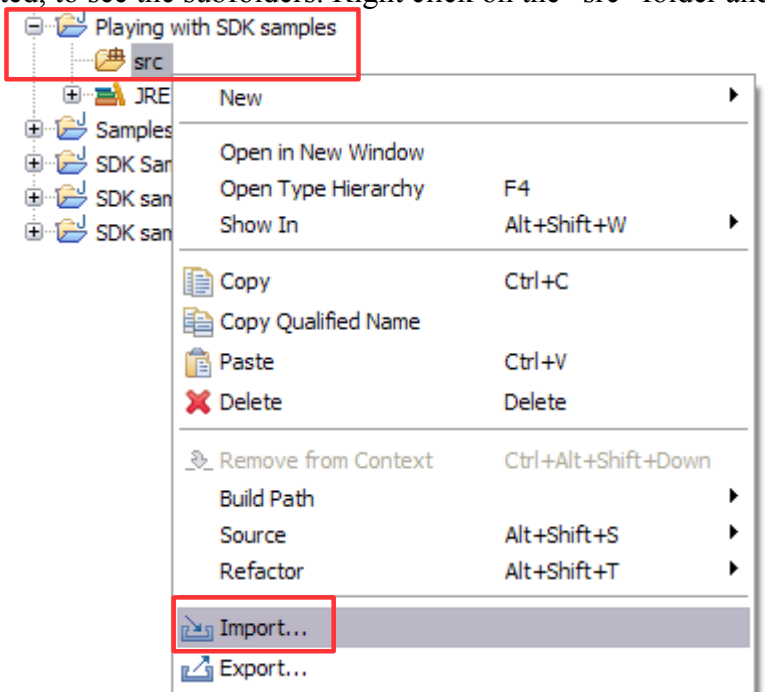
Step 2: CREATING A NEW PROJECT IN ECLIPSE



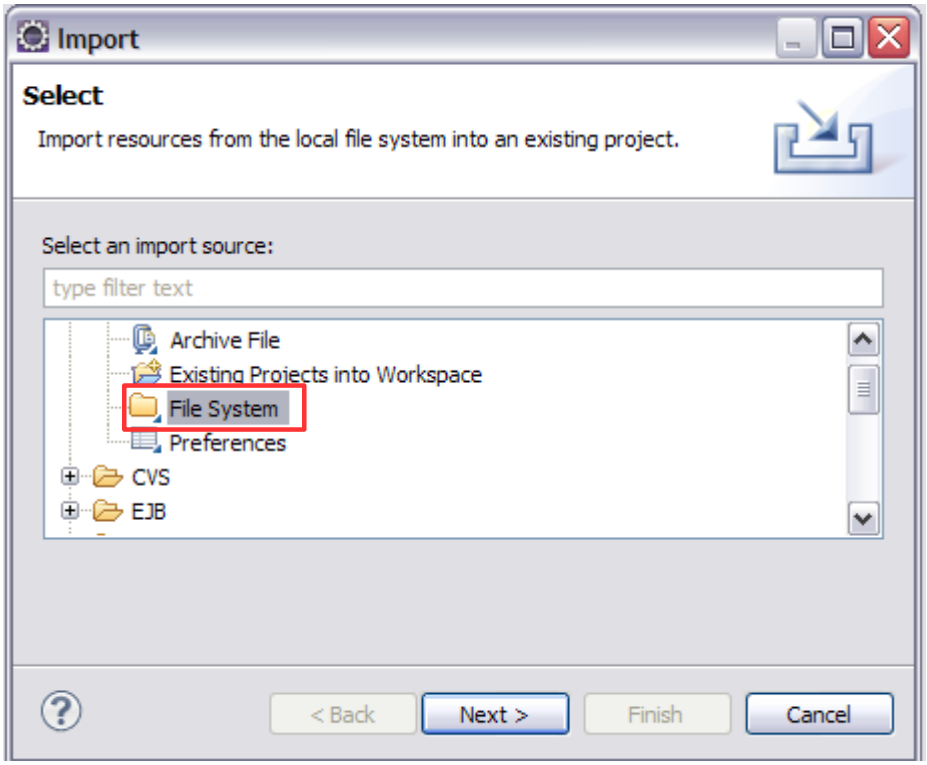
Right click on the left side of the window under the 'Package Explorer' pane to create a New project.
New → Java Project
The 'New Java Project' window pops up after clicking Java project from the previous step. Enter a project name of your choice and click Finish to create a java project.

Step 3: IMPORTING SAMPLES IN ECLIPSE

Expand the folder you created, to see the subfolders. Right click on the “src” folder and click on Import.

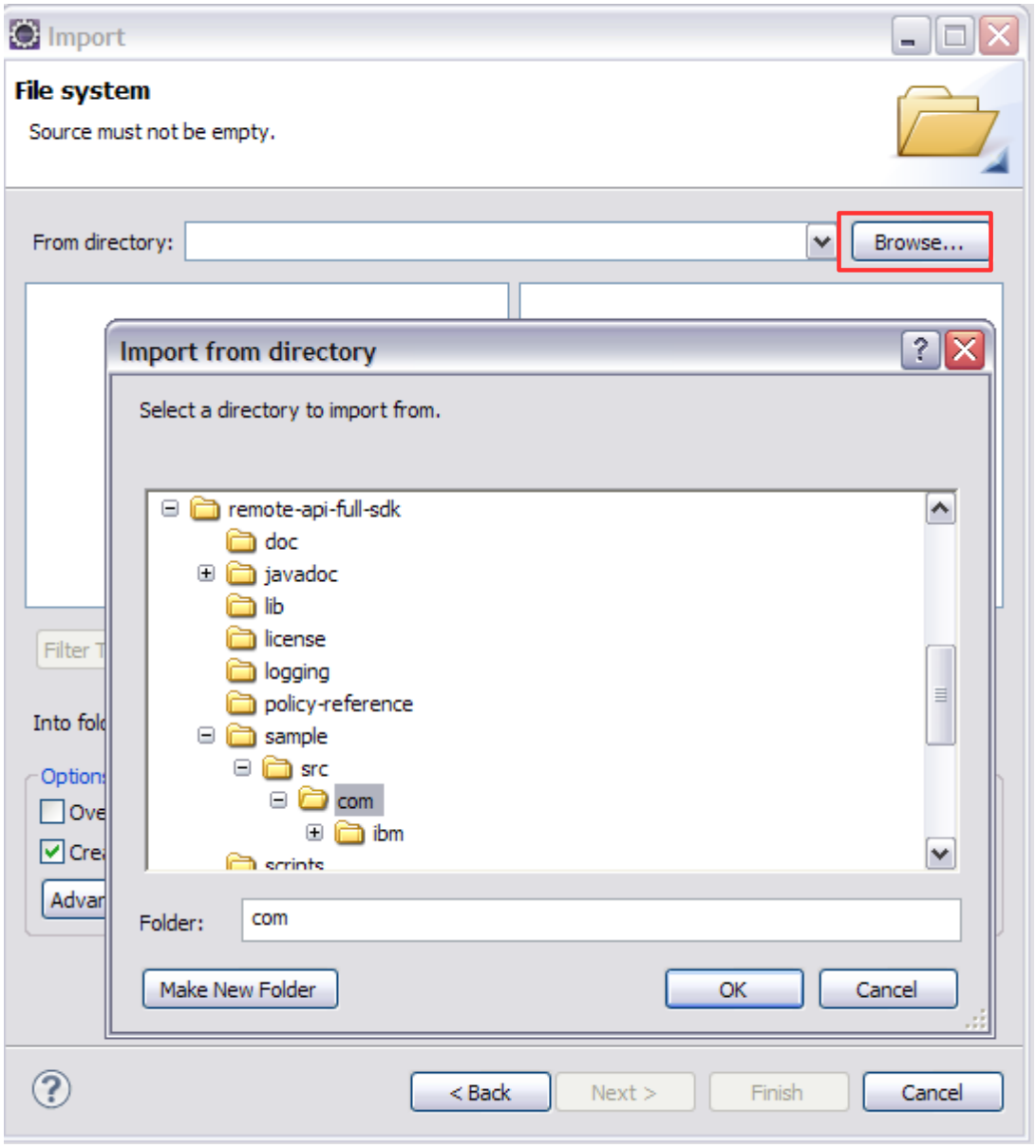


Select the option “File System” in the window 'Import' and click next.

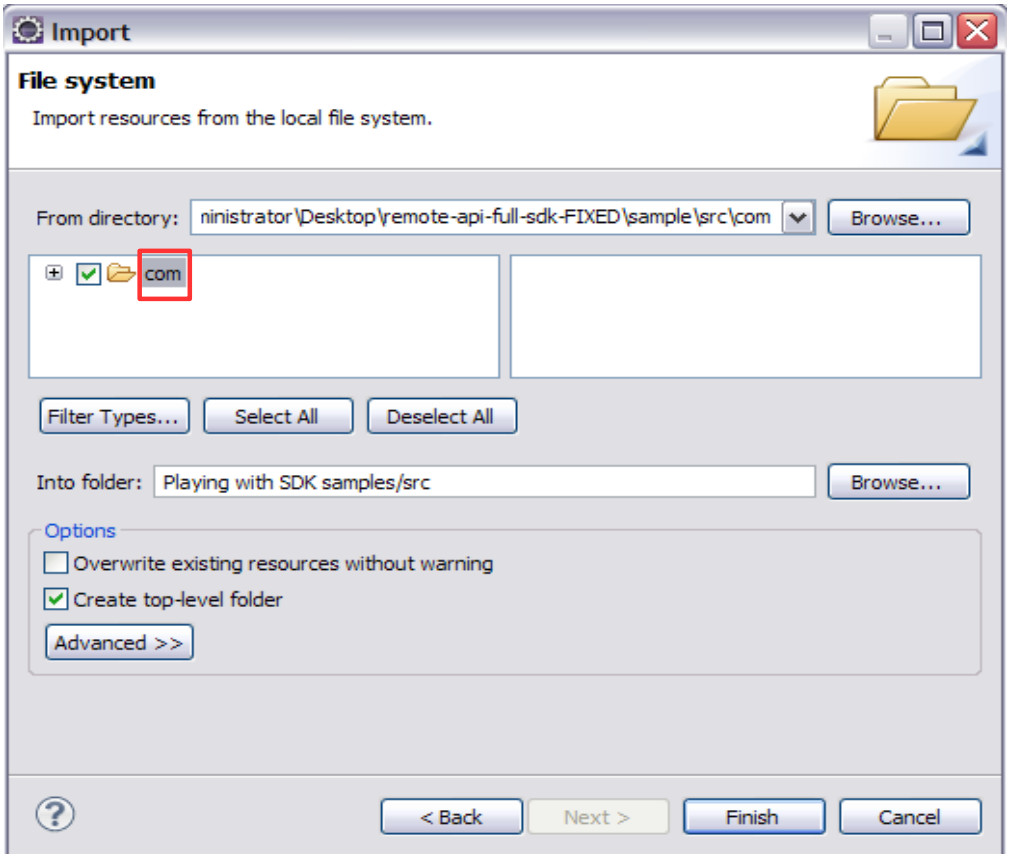


Click on Browse to import the samples from the directory where you extracted the .zip file in the first step. Navigate to the location where the extracted folder is located and expand the folder to finally reach the 'com' folder as shown below and click ok.

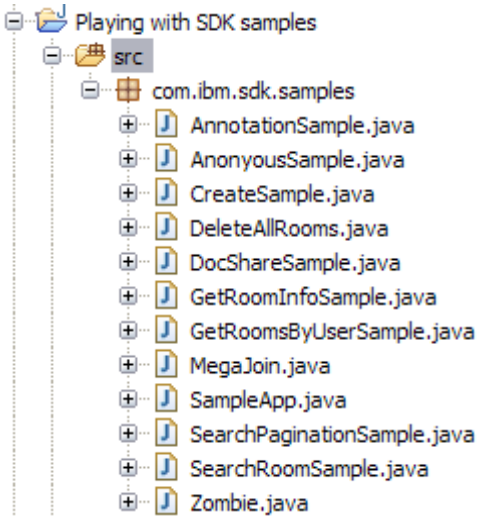
Remote-api-full-sdk → sample → src → com



On selecting the 'com' folder in the previous step the folder now appears on the left side on the window “Import”. Click on the Check button to the left of the folder and make sure the option “Create top-level folder” is checked. Then press Finish.

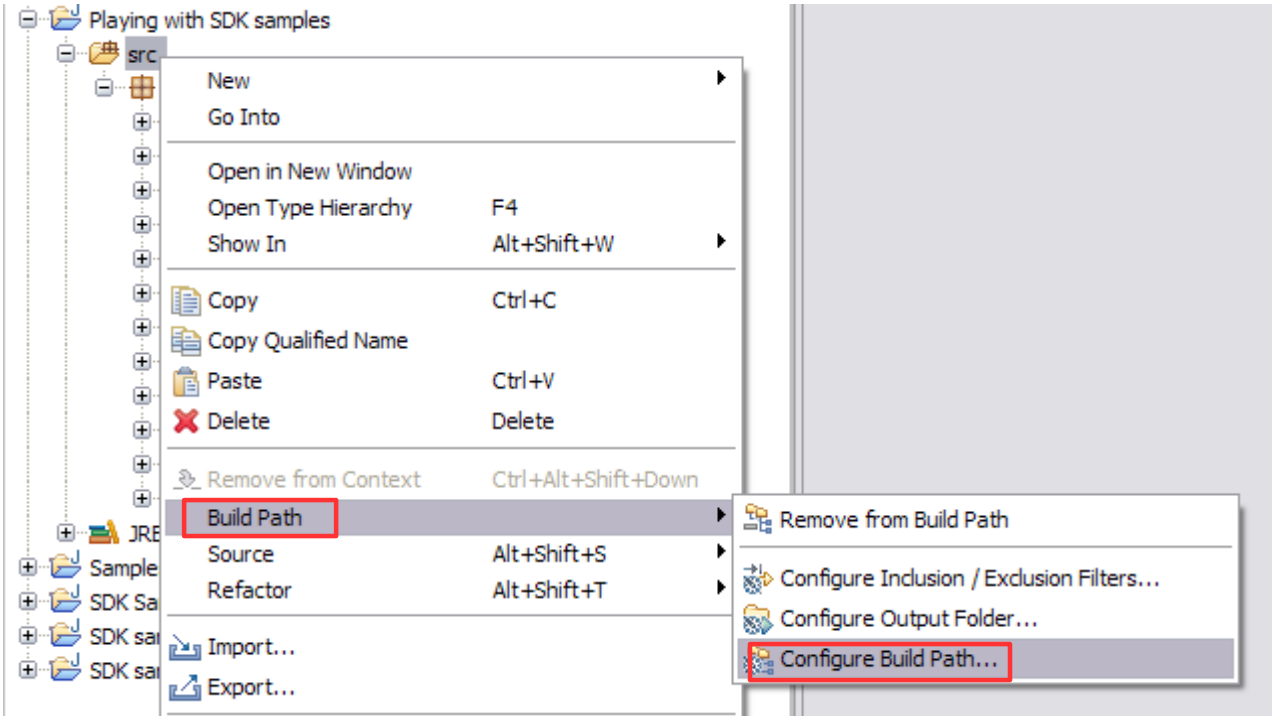


The Samples have been imported in Eclipse and now can be seen under the 'src' folder as shown below.

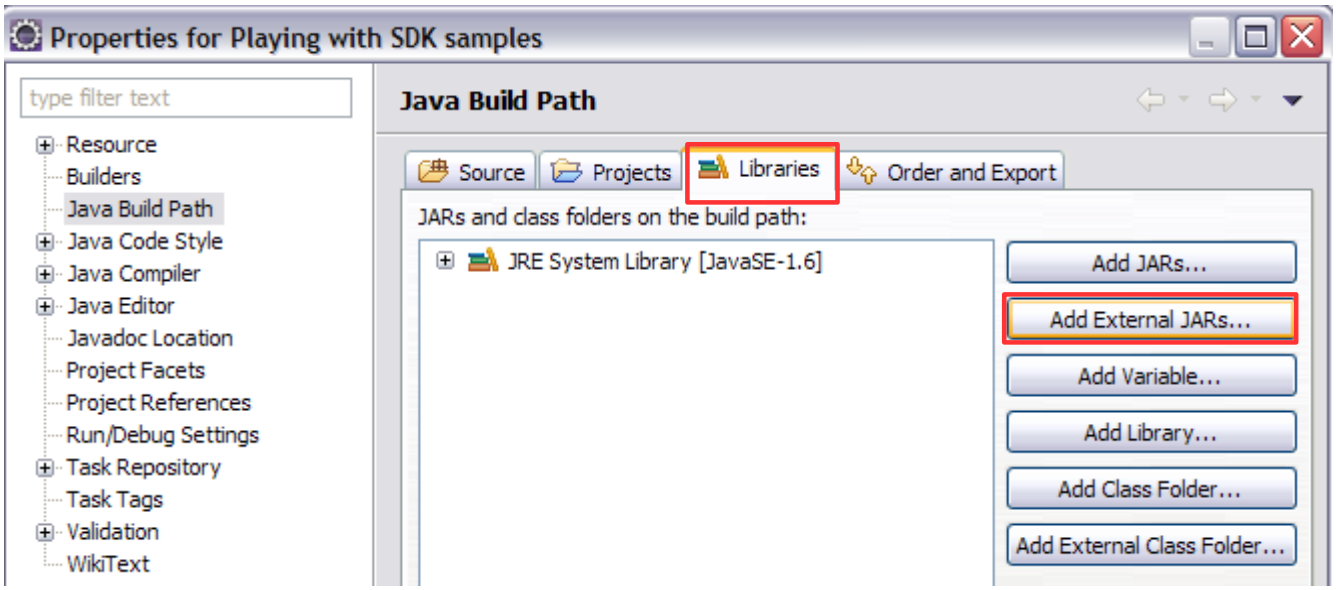


STEP 4: IMPORTING LIBRARIES

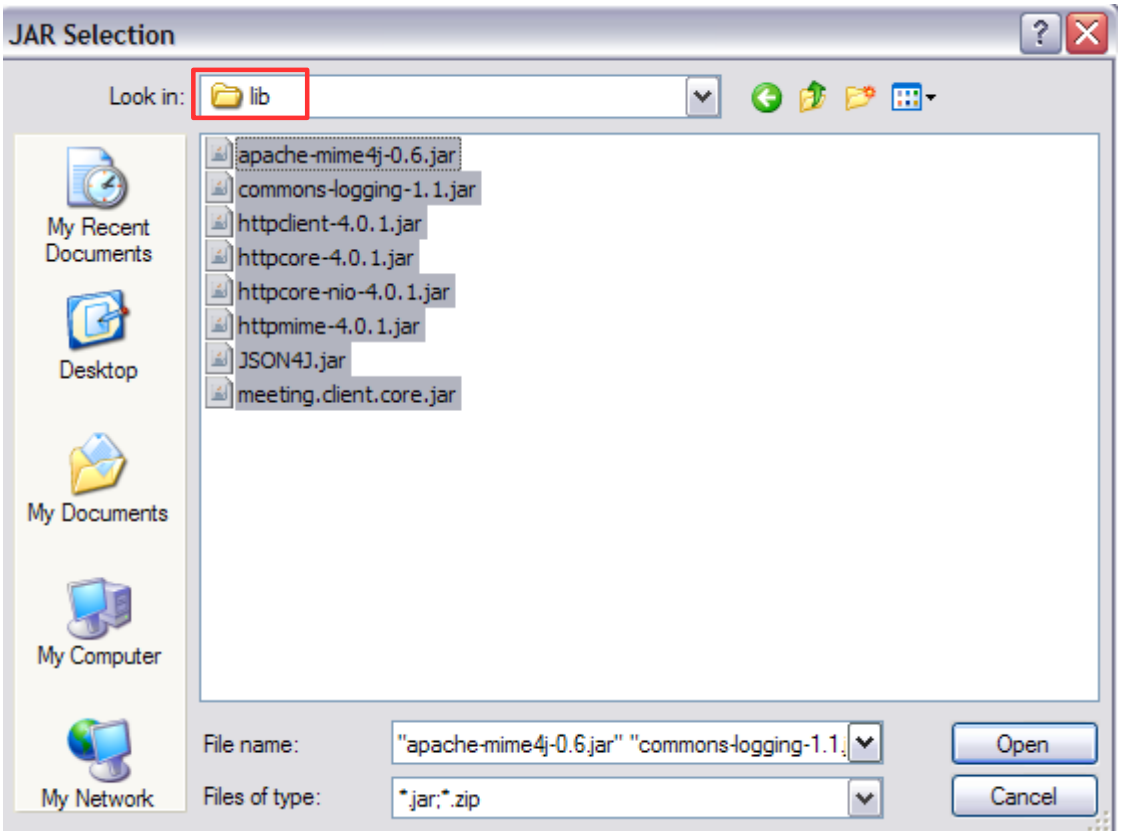
Right Click on the 'src' folder. Src → Build Path → Configure Build Path



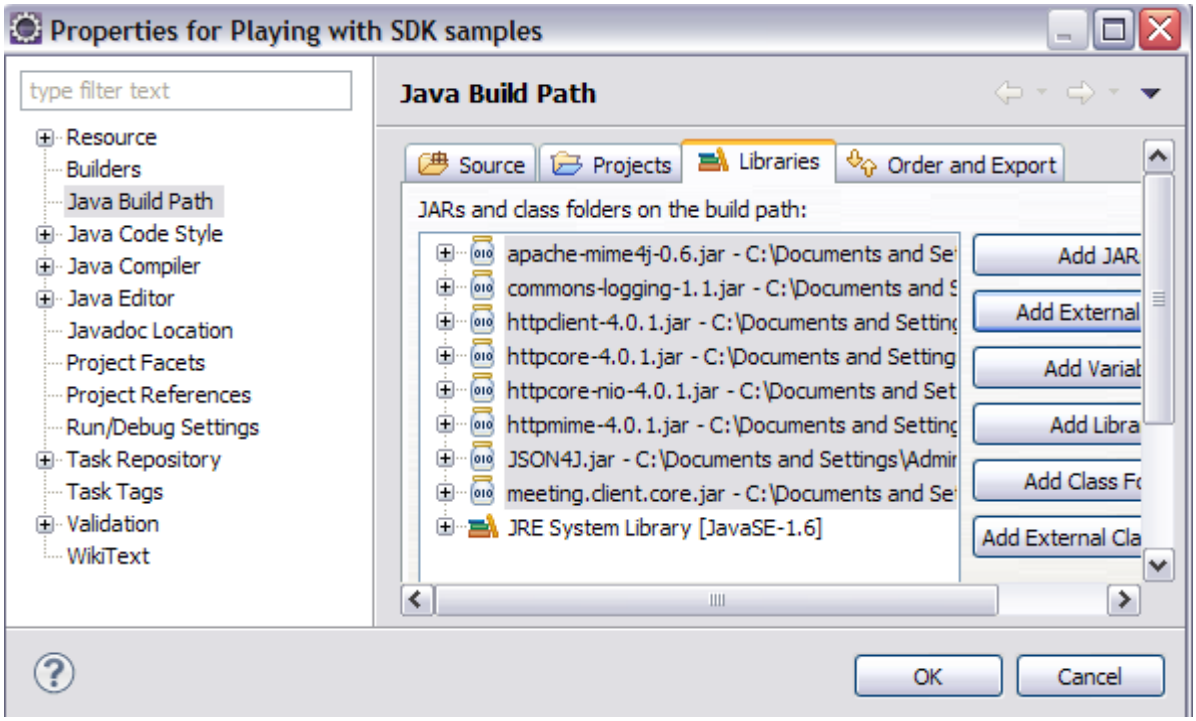
Click on 'Libraries' in the new window and then click on 'Add External JARs'



On Clicking 'Add External JARs' JAR selection window pops up where you need to navigate to the 'lib' folder inside the Remote-api-full-sdk folder. Select all files and click Open

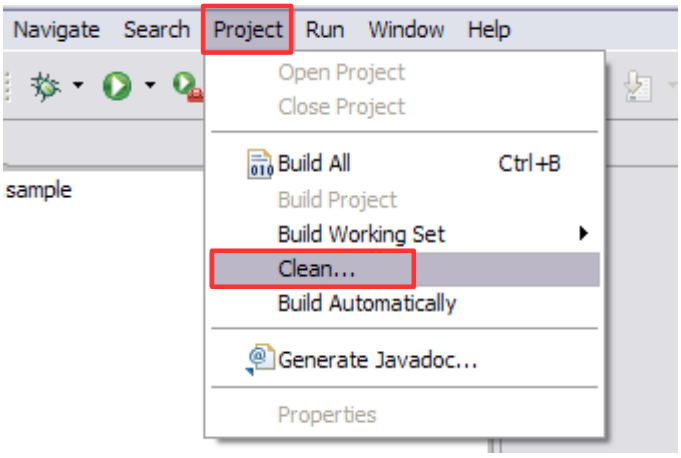


It adds all the JAR file selected in the previous step. To add these JAR files to the project click on Ok

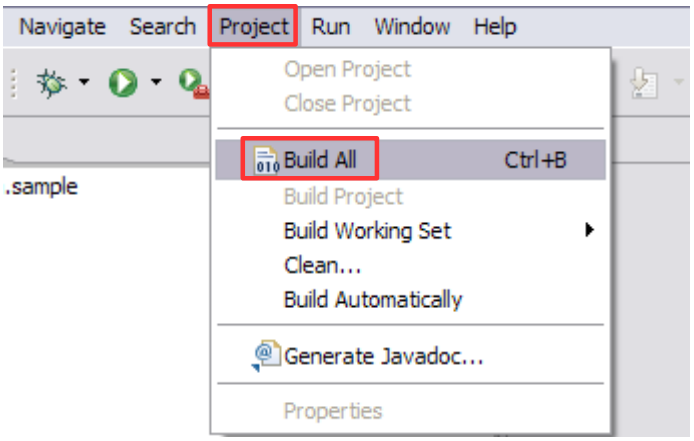


<u>STEP 5</u>: COMPLING AND RUNNING THE SAMPLES

Before Compiling the project do, Project →Clean as shown below



To compile all the projects in Eclipse, Project → Build All



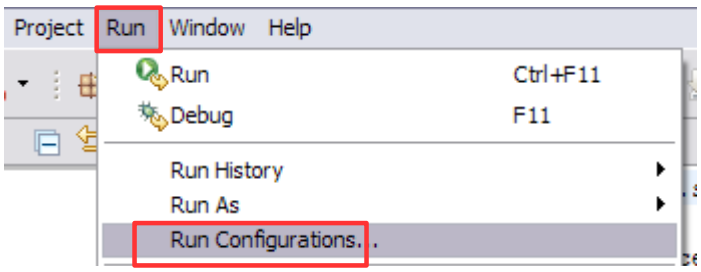
Before Executing the sample for the **FIRST** time, each sample requires arguments to be passed to it manually. Once the arguments are entered the samples can then run in the future without the need to re-enter the arguments.

The each sample may require a different combination of arguments. Enter only the arguments specified in the code for that sample, following the guidelines in the inline comments. The order of the arguments for all the sample codes is:

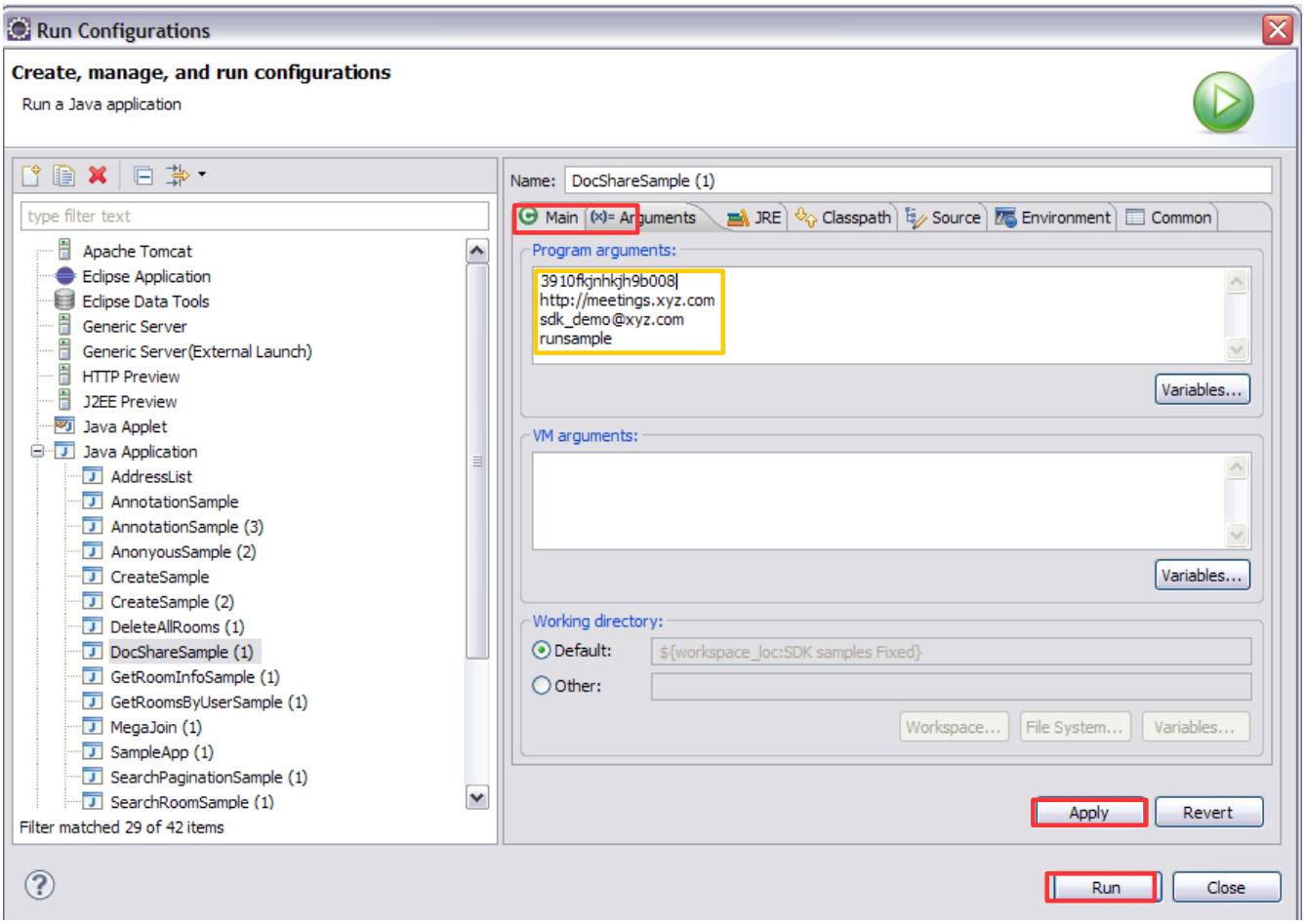
```
String Server   = args[0];
String Username = args[1];
String Password = args[2];
String Room_id  = args[3];
```

Say for example in a sample code there are only two arguments Server and Room_id then the Room_id will be passed as the second argument, String Room_id = args[1]; In case of a missing argument the next argument takes precedence in terms of the argument number passed in the sample code. A special case is the **LTPA_TEST sample** where LTPA token value is passed as an argument other than these standard arguments. There may be other samples with slightly different arguments as well.

Go to Run → Run Configurations



In the 'Run Configurations' window select the 'Arguments' tab and manually enter the arguments in the 'Program Arguments' area for the sample to run. Press Apply and then Run to execute the sample. To add a new sample to this menu you can right-click on Java Application and choose 'new'.



To run a particular sample of a project, first open the sample by expanding the Project.
Project Name → src (folder) → sample package→ sample (.java file)
In this case:

Playing with SDK samples → src → com.ibm.sdk.samples → AnnotationSample.java
double click on the file AnnotationSample.java

6.2 Sample: Creating a room

```
public class CreateSample

{
	public static void main(String[] args) throws IOException
	{
		String server = args[0];
		String username=args[1];
		String password=args[2];

		// Bind to a server.
		// note the server string, is a url string, with a port, like
http://server.company.com:9080
		RoomServices roomServices = null;
		try
		{
			roomServices = new RoomServices(server);
			roomServices.login(username,password);

			//Make up something random for the originId. It is the consumer's
responsibility to ensure
			//the originId/originType pair is unique. It will lead to
problems while searching otherwise.
			String originId = "STC"+System.currentTimeMillis();
			MeetingRoom room = new MeetingRoom("sample room");
			room.setOriginId(originId);
			room.setOriginType("myproduct");
			room.setPassword("password");
			// now lets create a room and save the roomId
			MeetingRoom meetingRoom = roomServices.createRoom(room);
			String tempRoomId = meetingRoom.getId();
			if(tempRoomId!=null)
			{
				System.out.println("Room created successfully");
			}
			System.out.println("Deleting room...");
			roomServices.deleteRoom(tempRoomId);
			System.out.println("Deleted room successfully");
			roomServices.logout();
		}
		catch (MalformedURLException e1)
		{
			e1.printStackTrace();
		}
		catch(RoomException e)
		{
			e.printStackTrace();
		}
		catch(LoginException e)
		{
			System.out.println(e.getMessage());
		}
	}
}
```

6.3 Sample: Getting room information

```
public class GetRoomInfoSample

{
	public static void main(String[] args) throws RoomException, IOException
	{
		String server = args[0];
		String username=args[1];
		String password=args[2];

		// Bind to a server.
		// note the server string, is a url string, with a port, like
http://server.company.com:9080
		RoomServices roomServices = null;
		try
		{
			roomServices = new RoomServices(server);
			roomServices.login(username,password);

			//Make up something random for the originId. It is the consumer's
responsibility to ensure
			//the originId/originType pair is unique. It will lead to problems while
searching otherwise.
			String originId = "STC"+System.currentTimeMillis();
			MeetingRoom room = new MeetingRoom();
			room.setRoomName("sample room2");
			room.setPassword("password");
			room.setOriginId(originId);
			room.setOriginType("myproduct");
			// now lets create a room and save the roomId
			room = roomServices.createRoom(room);

			String tempRoomId = room.getId();
			if(tempRoomId!=null)
			{
				System.out.println("Room created successfully");
			}

			//Find the room
			MeetingRoom myroom = roomServices.get(tempRoomId);
			if(myroom!=null)
			{
				System.out.println("Room name: " + myroom.getRoomName());
				System.out.println("Description: " + myroom.getDescription());
				System.out.println("Owner: " + myroom.getId());
				System.out.println("Permaname: " + myroom.getPermaName());
			}
			//Delete the room
			System.out.println("Deleting room...");
			roomServices.deleteRoom(tempRoomId);
			System.out.println("Deleted room successfully");
			roomServices.logout();
		}
		catch (MalformedURLException e)
		{
			e.printStackTrace();
		}
		catch(LoginException e)
		{
			System.out.println(e.getMessage());
```

```
            }
      }
}
```

6.4 Sample: Searching for a room based on originType and originId

```
public class SearchRoomSample

{
      public static void main(String[] args) throws RoomException, IOException
      {
            String server = args[0];
            String username=args[1];
            String password=args[2];

            // Bind to a server.
            // note the server string, is a url string, with a port, like
http://server.company.com:9080
            RoomServices roomServices = null;
            try
            {
                  roomServices = new RoomServices(server);
                  roomServices.login(username,password);

                  //Make up something random for the originId. It is the consumer's
responsibility to ensure
                  //the originId/originType pair is unique. It will lead to
problems while searching otherwise.
                  String originId = "STC"+System.currentTimeMillis();
                  MeetingRoom room = new MeetingRoom("sample room3");
                  room.setOriginId(originId);
                  room.setOriginType("myproduct");
                  room.setPassword("password");
                  // now lets create a room and save the roomId
                  room = roomServices.createRoom(room);
                  String tempRoomId = room.getId();
                  if(tempRoomId!=null)
                  {
                        System.out.println("Room created successfully");
                  }

                  //Find the room
                  MeetingRoom result =
roomServices.getRoomByOriginInfo("myproduct", "STC");
                  if(result!=null)
                  {
                        System.out.println("Room name: " + result.getRoomName());
                        System.out.println("Description: " +
result.getDescription());
                        System.out.println("Owner: " + result.getId());
                        System.out.println("Permaname: " + result.getPermaName());
                  }

                  System.out.println("Deleting room...");
                  roomServices.deleteRoom(tempRoomId);
                  System.out.println("Deleted room successfully");

            }
            catch (MalformedURLException e)
```

```
            {
                  e.printStackTrace();
            }
            catch(LoginException e)
            {
                  System.out.println(e.getMessage());
            }
      }

}
```

6.5 Sample: Anonymous Join, RealTimeSessionListener, and Modifying Maps

The RealTimeSession object provides the entry point into all realtime operations controlling session state.

```
public class AnonyousSample
{
      public static void main(String[] args) throws Exception
      {
            final String SERVER = args[0];            // like "http://server.company.com";
            final String USERNAME = args[1];    // like "someone@company.com";
            final String ROOMID = args[2];            // like "6f2d3d61-8998-4b32-b647-
                                                            2e737a63b38a";
            System.out.println("C roomId ["+ROOMID+"]");


            RealTimeSession room = new RealTimeSession(SERVER, ROOMID, USERNAME);
            //Note, we must set our "guest" name since we're anonymous
            room.setUser(USERNAME);

            // before we join the room, lets bind a listener to receive updates
            room.addListener(new RealTimeSessionListener()
            {
                  public void handleEvents(ArrayList<RealTimeSessionEvent> events)
                  {
                        Date date = new Date();
                        System.out.println(date.toString()+"\nE1 from server:
                              \n"+Arrays.toString(events.toArray()));
                  }
                  public void handleEvent(RealTimeSessionEvent event) {
                        System.out.println("E2 from server: \n"+event);
                  }
                  public void handleError(RealTimeSession rtSession, HttpResponseWrapper
                                                response, Exception e){
                        System.out.println("E3 from server: \n"+response);
                        e.printStackTrace();
                  }
            });

            BufferedReader console = new BufferedReader(new InputStreamReader(System.in));

            System.out.println("Press <Enter> to join the room");
            // wait for enter
            String line = null;
            do
            {
                  line = console.readLine();
            }
```

```
            while(line == null);
            System.out.println("starting");

            // now lets join the room
            System.out.println("F joining room");
            HttpResponseWrapper joinResponse = room.join();
            System.out.println("join response: "+joinResponse);

            // now lets start the updates for receiving data
            System.out.println("G starting updates");
            room.startUpdates();
            room.putInMap("UserMap",USERNAME+".handRaise", "true");


            System.out.println("Press <Enter> to leave the room");
            // wait for enter
            line = null;
            do
            {
                  line = console.readLine();
            }
            while(line == null);


            // now lets stop updates and leave
            System.out.println("I leaving the room");
            room.stopUpdates();
            room.leave();

            // log out to be clean
            System.out.println("K logging out");
      }
}
```

6. Logging and troubleshooting

Standard java logging has been used in the toolkit. A logging.properties file has been included in the “logging” directory of the tookit zip file.

In order to enable package/class level logging, please modify the “logging.properties” file in the “lib” directory.

Set the java system property

-Djava.logging.config.file=<path>/logging.properties

This should enable logging. Logs will be captured in a java.log file in the user's home directory. The location of this file can also be modified via the logging.properties file.

7. Notices

Disclaimer

Copyright

the United States, other countries, or both.
Other company, product, or service names may be trademarks or service marks of others.

Third Party Notices
For STLport
License Agreement
Boris Fomitchev grants Licensee a non-exclusive, non-transferable, royalty-free license to use STLport and its documentation without fee.
By downloading, using, or copying STLport or any portion thereof, Licensee agrees to abide by the intellectual property laws and all other applicable laws of the United States of America, and to all of the terms and conditions of this Agreement.
Licensee shall maintain the following copyright and permission notices on STLport sources and its documentation unchanged :
Copyright 1999,2000 Boris Fomitchev
This material is provided "as is", with absolutely no warranty expressed or implied. Any use is at your own risk.
Permission to use or copy this software for any purpose is hereby granted without fee, provided the above notices are retained on all copies. Permission to modify the code and to distribute modified code is granted, provided the above notices are retained, and a notice that the code was modified is included with the above copyright notice.
The Licensee may distribute binaries compiled with STLport (whether original or modified) without any royalties or restrictions.
The Licensee may distribute original or modified STLport sources, provided that:
The conditions indicated in the above permission notice are met; The following copyright notices are retained when present, and conditions provided in accompanying permission notices are met :
Copyright 1994 Hewlett-Packard Company
Copyright 1996,97 Silicon Graphics Computer Systems, Inc.
Copyright 1997 Moscow Center for SPARC Technology.
Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Hewlett-Packard Company makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Silicon Graphics makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Moscow Center for SPARC Technology makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
Copyright 2001 by STLport

For MD5 hash

For Log4J Logging

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.