Managing Chat Sessions

Chat Sessions are created within IM Sessions and are used to manage and participate in person-to-person chats and chat rooms. All text-based instant message chats are handled using the IChatSession interface, which offers methods for sending text or data messages and inviting new participants into a chat.

You can attach a Chat Listener to a Chat Session to listen to the messages associated with it. Handling Chat Sessions is particularly useful for integrating text messaging within your own applications. Using a Chat Session, you can create a chat room for multiplayer games, or integrate person-toperson messaging within a mobile social networking application.

Starting or Joining a Chat Session

A Chat Session represents the conduit through which all instant messaging communication with a target user passes, so you can only maintain a single Chat Session per contact per IM Session.

New Chat Sessions are created through an IM Session, using the getChatSession or createChatSession methods.

If a Chat Session already exists for a given contact, retrieve it by passing in the username of the person with whom you wish to converse, as shown in the following snippet. If there is no active Chat Session with the specifi ed user, this method returns null.

```
IChatSession cs = imSession.getChatSession(targetContactEmailAddress);
```

If you haven't established a Chat Session with a particular user, create one using the createChatSession method, passing in the target contact's username. If the IM Session is unable to create a new Chat Session, this method will return null.

```
IChatSession chatSession = imSession.createChatSession(targetContactEmailAddress);
```

The following pattern checks to see if there is an existing Chat Session with a target user before creating a new one if necessary:

```
IChatSession chatSession = imSession.getChatSession(targetContactEmailAddress);
if (chatSession == null)
chatSession = imSession.createChatSession(targetContactEmailAddress);
```

Group Chat Sessions are also represented using the IChatSession interface, but they're handled a little differently. Group chat functionality is explored in more detail later in this chapter.

Sending Instant Text Messages

Once you have an active Chat Session, use the sendChatMessage method to send messages to the contact(s) in that session, as shown in the following code snippet:

```
chatSession.sendChatMessage("Hello World!");
```

The message text specifi ed will be transmitted to all the contacts involved in the current Chat Session.

Receiving Instant Text Messages

To listen for incoming messages, implement the IChatListener interface, overriding its newMessageReceived handler. You can register this interface with either a specifi c Chat Session or the more generic IM Session using the addRemoteChatListener method.

The following snippet shows the skeleton code for creating and registering a new Chat Listener interface for both a specifi c Chat Session and an IM Session. Note that the IChatListener interface includes a Stub class that you should extend when creating your own Chat Listener implementation.

```
IChatListener chatListener = new IChatListener.Stub() {
public void newMessageReceived(String from, String body) {
// TODO Handle incoming messages.
}
// Required group chat implementation stubs.
public void convertedToGroupChat(String oldJid,
String groupChatRoom,
```

```
long groupId) {}
public void participantJoined(String groupChatRoom, String nickname) {}
public void participantLeft(String groupChatRoom, String nickname) {}
public void chatClosed(String groupChatRoom) throws RemoteException {}
public void chatRead(String arg0) throws RemoteException {}
};
// Add Chat Listener to the chat session.
chatSession.addRemoteChatListener(chatListener);
// Add Chat Listener to the instant messaging session.
imSession.addRemoteChatListener(chatListener);
```

Chat Listeners registered with an IM Session receive every message received by any Chat Session associated with that session, so the message handling here should be fairly generic. In contrast, listeners registered to a single Chat Session are only notifi ed of messages and events relevant to that specifi c session.

Chat Rooms and Group Chats

Chat rooms are an excellent way to encourage a sense of community within a collaborative or multiuser application.

The GTalk Service supports chat rooms and group chats. They are managed using the same IChatSession interface used for simple P2P Chat Sessions.

To create a new chat room, use the createGroupChatSession method on an IM Session, passing in a nickname for the room and a list of users to invite, as shown in the following snippet:

```
String nickname = “Android Development”;
String[] contacts = { “bill”, “fred” };
imSession.createGroupChatSession(nickname, contacts);
```

Alternatively, you may want to join group chats that others have invited you to. Use the IGroupChatInvitationListener interface to listen for group chat invitations. Each invitation includes the address and password needed to join an existing chat room.
To join an existing chat room, use the joinGroupChatSession method from an active IM Session, passing in the address of the room you want to join, a nickname for you to identify it, and the password

required to join, as shown in the following snippet:

```
imSession.joinGroupChatSession(address, nickname, password);
```

The following skeleton code shows how to register a Group Chat Invitation Listener on an active IM Session to listen for, and accept, invitations to join chat rooms.

```
IGroupChatInvitationListener listener = new IGroupChatInvitationListener.Stub() {
public boolean onInvitationReceived(GroupChatInvitation _invite)
throws RemoteException {
String address = _invite.getRoomAddress();
String password = _invite.getPassword();
String nickname = _invite.getInviter();
imSession.joinGroupChatSession(address, nickname, password);
return true;
}
};
try {
imSession.addGroupChatInvitationListener(listener);
} catch (RemoteException e) { }
```

Managing Group Chat Sessions

You can get a list of participants in a Chat Session using the getParticipants method. You can also send text or data messages to each chat member as you would in a normal chat, as well as invite new members using inviteContact. The leave method lets you exit a chat room and end the session.

As with normal chats, you can listen to chat room messages by implementing and registering an IChatListener. As well as listening for chat messages, you can react to people joining or leaving the room.

The following skeleton code shows the implementation of a Chat Listener highlighting the group chat event handlers:

```
IChatListener groupChatListener = new IChatListener.Stub() {
// Fired when a one-to-one chat becomes a group chat.
```

```
public void convertedToGroupChat(String oldJid,
String groupChatRoom,
long groupId) throws RemoteException {
// TODO Notify user that the conversation is now a group chat.
}
// Fired when a new person joins a chat room.
public void participantJoined(String groupChatRoom, String nickname)
throws RemoteException {
// TODO Notify user that a new participant has joined the conversation.
}
// Fired when a participant leaves a chat room.
public void participantLeft(String groupChatRoom, String nickname)
throws RemoteException {
// TODO Notify user a chat participant left.
}
// Fired when the group chat is closed
public void chatClosed(String groupChatRoom) throws RemoteException {
// TODO Close the chat.
}
public void chatRead(String arg0) throws RemoteException { }
public void newMessageReceived(String from, String body) { }
};
```