Using Bluetooth OBEX Client with Authenticator in Java ME

Overview

This code example describes how to use OBEX(Object Exchange Protocol) protocol service via bluetooth as client-side chat with authentication process. We must include `Authenticator` (see code in MessengerAuthenticator.java) interface implementation, that let us use authentication. This interface provides a way to respond to authentication challenge and authentication response headers through `onAuthenticationChallenge()` and `onAuthenticationResponse()` methods.

`Authenticator` implementation appended to Obex_messenger_client snippet. Also we need to correct `servicesDiscovered()` method. After we got `ClientSession object`, we must create a `MessengerAuthenticator` instance that implements the `Authenticator` interface and set it as Authenticator of `ClientSession` by calling `setAuthenticator()` method.

Then create Authentication `HeaderSet` object that will be signaling to server that we want authenticate him after `connect()` method calling.

After that authentication process calls `onAuthenticationChallenge()` on server and `onAuthenticationResponse()` on our client and compares authentication data. If it's ok, server may authenticate us and our `onAuthenticationChallenge()` method will be called where we must return the correct `PasswordAuthentication` object instance. And if it's ok we can chat.

Preconditions

Bluetooth must be enabled on device before using.

It can't be used independently from server. For testing use Bluetooth OBEX server with authenticator (look at "See also" part)

Source file: BtObexCliAuthenticatorMidlet.java

```
//practically all code is the same as in Obex_messenger_client snippet
//with some difference in servicesDiscovered() method
    /**
    * From DiscoveryListener.
    * Called when service(s) are found during a service search.
    * This method provides the array of services that have been found.
    * Trying to connect to if Messenger service has been found.
    * @param transID the transaction ID of the service search that is
    * posting the result
    * @param service a list of services found during the search request
    */
    public void servicesDiscovered(int transID, ServiceRecord[] serviceRecords){
        for(int i = 0; i < serviceRecords.length; i++) {
            DataElement rcvObject;

            //getting ServiceClassID as DataElement object
            rcvObject = serviceRecords[i].getAttributeValue(
                    ServiceRecord.ID_ServiceClassIDList);

            if(rcvObject!=null) {
                //rcvObject presents as enumeration of DataElement
                //getting enumeration...
                Enumeration enumeration = (Enumeration)rcvObject.getValue();

                for (Enumeration e = enumeration; e.hasMoreElements() ;) {
                    DataElement subDataElement = (DataElement)e.nextElement();
                    //...and converting every element of enumeration to UUID
                    UUID uuid = (UUID) subDataElement.getValue();
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    //comparing needed UUID with present
                    if(serviceClassIDstring.equals(uuid.toString())) {
                        //if UUIDs are the same then trying to connect
                        //getting the connection string for connection
                        String connectionString =
                            serviceRecords[i].getConnectionURL(
                                ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
                        printToForm(connectionString);

                        try {
                            //getting the client connection object
                            connectionSession = (ClientSession) Connector.open(
                                                connectionString,
                                                Connector.READ_WRITE );

                            //creating authenticator for OBEX connection session
                            MessengerAuthenticator authenticator =
                                        new MessengerAuthenticator();
                            connectionSession.setAuthenticator(authenticator);

                            //creating authenticator header
                            //defining that we want to authenticate the server
                            HeaderSet authHead =
                                    connectionSession.createHeaderSet();
                            authHead.createAuthenticationChallenge(
                                        "MessengerClient", true, true);

                            //trying to connect
                            HeaderSet headerSet =
                                    connectionSession.connect(authHead);

                            if (headerSet.getResponseCode() ==
                                    ResponseCodes.OBEX_HTTP_OK) {
                                printToForm("Request OK");
                            } else {
                                printToForm("Request Failed. code: " +
                                        headerSet.getResponseCode());
                                connectionSession.close();
                                return;
                            }

                            //if connection established we add some commands
                            //for sending message and disconnecting from
                            //server
                            frmMain.addCommand(cmdPut);
                            frmMain.addCommand(cmdDisconnect);
                            //TimerTask object creating for periodic
                            //asynchronous sending GET requests to server
                            ObexClientTimerTask timerTask =
                                    new ObexClientTimerTask(this);
                            //timer starting
                            timer.schedule((TimerTask)timerTask, 1000);
                            return;
                        } catch (IOException ex) {
                            printToForm(ex.toString());
                        }
                    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
                }
            }
        }
    }
```

Source file: ObexClientTimerTask.java

```
import java.util.TimerTask;

/**
 * A task object that can be scheduled for one-time
 * or repeated execution by a Timer
 * Used in BtObexCliAuthenticatorMidlet object for sending GET requests
 * to OBEX Server
 */
public class ObexClientTimerTask extends TimerTask {
    /**
     * This midlet used for calling GetMessageOnTimer() which create requests
     * to OBEX server
     */
    BtObexCliAuthenticatorMidlet parentMidlet;

    /**
     * Constructor
     * @param parent is BtObexCliAuthenticatorMidlet. Used for calling
     * GetMessageOnTimer() function in run().
     */
    public ObexClientTimerTask(BtObexCliAuthenticatorMidlet parent) {
        parentMidlet = parent;
    }

    /**
     * From TimerTask.
     * The action to be performed by this timer task.
     */
    public void run() {
        //calling function that create GET reqest to OBEX Server
        parentMidlet.GetMessageOnTimer();
        }
}
```

Source file: MessengerAuthenticator.java

```
import javax.obex.Authenticator;
import javax.obex.PasswordAuthentication;

/**
 * Instance of this object provides a way to
 * respond to authentication challenge and authentication
 * response headers
 */
```


(C) Copyright Nokia 2014. All rights reserved.

```
public class MessengerAuthenticator implements Authenticator {

    /**
     * Constructor
     */
    public MessengerAuthenticator() {
    }

    /**
     * From Authenticator.
     * Function called when a client or a server receives an authentication
     * challenge header. It should respond to the challenge with a
     * PasswordAuthentication that contains the correct user name and password
     * for the challenge.
     * @param description the String that describes of which user name and
     * password should be used. Not used
     * @param isUserIdRequired is boolean. true if the user ID is required.
     * false if the user ID is not required. Not used
     * @param isUserIdRequired is boolean. true if full access to the server
     * will be granted; false if read only access will be granted. Not used
     */
    public PasswordAuthentication onAuthenticationChallenge(String description,
                                                            boolean isUserIdRequired,
                                                            boolean isFullAccess) {
        PasswordAuthentication passAuthentication =
            new PasswordAuthentication("user".getBytes(), "user".getBytes());
        return passAuthentication;
    }

    /**
     * From Authenticator.
     * Called when a client or server receives an authentication response
     * header. This method will provide the user name and expect the correct
     * password to be returned.
     * @param userName is byte[].
     * the user name provided in the authentication response. Not used
     */
    public byte[] onAuthenticationResponse(byte[] userName) {
        String result = "user";
        return result.getBytes();
    }
}
```

Postconditions

MIDlet implements client-side of Obex chat with Authenticator.

See also

- Using Bluetooth OBEX Client in Java ME
- Using Bluetooth Obex server authentication in Java ME

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.