Introduction to Bluetooth and J2ME Part 2

By Jason Lam · December 27, 2004 · 3 Comments

In Part 1 we went over the basics of Bluetooth technology and some possible development opportunities Bluetooth gives us. Part 2 of this article goes into a little more detail how to create a Bluetooth server and client.

About the Example

In keeping things simplistic as possible proper handling of multiple slaves/clients is not implemented essentially it only works with one server and one client. As well the sample is just that sample code, it is coded from the perspective of getting the two nodes to communicate with each without any real effort in “proper” coding technique/design. Remember the focus is to get you started with the actual Java Bluetooth API.

Before we continue you should download the JSR 82 Java Doc API from http://jcp.org/aboutJava/communityprocess/final/jsr082/ . Familiarize yourself with the available classes and get feel where things are. We won’t go into every single class or explain every single detail mainly because the Java Doc does a good job of this already.

This demo includes three source files / classes:

- BluetoothEchoDemo.java – The main driver that allows you to start a client or server
- EchoClient – The client class
- EchoServer – The server class

This example uses RFCOMM for transferring data the other two options are L2CAP and OBEX. In short RFCOMM is used for streaming data, L2CAP is for packet data and OBEX is for object data.

Client Overview

Let us start with the client, the first thing we need to do is to determine what Bluetooth devices

and services are there around us. First get the local device and obtain the discovery agent.

```
LocalDevice localDevice = LocalDevice.getLocalDevice();

discoveryAgent = localDevice.getDiscoveryAgent();
```

Now to start searching for Bluetooth devices within the area use the startInquiry method from the DiscoveryAgent class.

```
discoveryAgent.startInquiry(DiscoveryAgent.GIAC,this);
```

DiscorveryAgent.GIAC stands for “The inquiry access code for General/Unlimited Inquiry Access Code” refer to the Java Docs for the other inquiry access code definitions.

Now with the use of the Discovery Listener you have available to you 4 call back methods. As defined in the Java Doc they are:

Method Name	Description
void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)	Called when a device is found during an inquiry.
void inquiryCompleted(int discType)	Called when an inquiry is completed.
void servicesDiscovered(int transID, ServiceRecord[] servRecord)	Called when service(s) are found during a service search.
void serviceSearchCompleted(int transID, int respCode)	Called when a service search is completed or was terminated because of an error.

That is it for the discovery part! It really is that simple. Now with these calls being triggered you can obtain device information, service information and with that make the appropriate data transmission/communication with the Bluetooth server. (see source code)

In the deviceDiscovered method we obtain some device info which are more or less self-explanatory. Except for the Major and Minor classification, these two attributes let you know what type of device it is. The following is an incomplete sample of the Major and Minor definitions, for a complete list visit www.bluetooth.org . When you run the program you will notice it outputs Major 512 and Minor 4, according to the chart it is a Phone / Celluar.

Major Class	Minor Class	Major Class Description	Minor Class Description
256	4	Computer	Desktop
256	8	Computer	Server
256	12	Computer	Laptop
256	20	Computer	PDA
512	4	Phone	Cellular
512	8	Phone	Household cordless
512	12	Phone	Smart phone
1536	32	Imaging device	Camera
1536	64	Imaging device	Scanner
1536	128	Imaging device	Printer

In the servicesDeiscovered method you can obtain the respective URL needed to open a connection to the available service(s).

```
for(int i=0;i   serviceUrl = servRecord[i].getConnectionURL(0,false);
}
```

Now that you have the url, you can send data through the standard Generic Connector.

```
String msg = "Say Hello World";
conn = (StreamConnection)Connector.open(serviceUrl);
OutputStream output = conn.openOutputStream();
output.write(msg.length());
output.write(msg.getBytes());
output.close();
```

Server Overview

In the server class we need to initialize once again but this time instead of searching for devices we need to setup a server

```
private static String serverUrl = "btspp://localhost:" +
BluetoothEchoDemo.RFCOMM_UUID + ";name=rfcommtest;authorize=true";
.
.
.
conn = null;
localDevice = LocalDevice.getLocalDevice();
localDevice.setDiscoverable( DiscoveryAgent.GIAC );
notifier = (StreamConnectionNotifier)Connector.open(serverUrl);
.
.
.
```

Now make a connection with the same Generic Connector you use when making HTTP calls. The url definition is as follows:

```
scheme://host:port;parameters
```

Name	Description
scheme	Connection type such as http, in the case of Bluetooth you use btspp for RFCOMM or btl2cap for L2CAP
host	The address to connect to or if you are setting up a server then you put in localhost
port	The port of the client connections and for servers it describes the UUID more about this later.
parameters	Specify option parameters ie: like given the service name name=rfcommtest

Now wait for a client response, this pauses the thread until it receives something.

```
conn = notifier.acceptAndOpen();
```

Once a client responds read the data in and send back a message

```
// Read Data Transmission
String msg = BluetoothEchoDemo.readData(conn);
System.out.println("Received Message from Client: " + msg);
// Send Back a Message
msg = "Hello Back from Server";
output = conn.openOutputStream();
output.write(msg.length()); // length is 1 byte
output.write(msg.getBytes());
output.close();
```

Running the Example

Download the source and compile from here.

You will need to invoke the Sun WTK twice because the outputs are displayed in the consoles. Start one instance and select Server, then start a second instance and select Client (Yes you need to start 2 instances of the WTK not just simply hit the "run" button twice on the same instance because the console of the first instance will not show). Like HTTP calls you may need to answer yes for connections being made.

In the end you will see the following in the server console:

```
Starting Echo Server
Server Running...
Received Message from Client: Say Hello World
```

In the client console you will see:

Judge Says Gmail Might Be Breaking The Law

Google continues to fight for right to scan your emails

3 Comments

BlackBerry Posts $965 Million Quarterly Loss

A dismal financial report from a company on the brink

```
Major Device Class: 512 Minor Device Class: 4
Bluetooth Address: 000060854FBF
Bluetooth Friendly Name: WirelessToolkit
InquiryCompleted
ServicesDiscovered
SERVICE_SEARCH_COMPLETED
Service URL:
btspp://000060854FBF:1;master=false;encrypt=false;authenticate=false
Hello Back from Server
```

Summary and What is Missing

You should be able to make a basic RFCOMM communication between a Bluetooth client and server. However, even though we went through a lot detail there are still details that were left out or not mentioned. Now that you have a better understanding of Bluetooth with J2ME you need to dive deeper into the following:

- An understanding of Service Discovery Database (SDDB)
- A more in depth look at UUIDs and other Data Elements
- Understand what a Service Record is and what it does
- Understanding the relationship between SDDB, Service Record and Data Elements

As a recap the example in this article the primary goal is to give you an introduction to setting up the communication between a master and slave. Rather then jumping into a full example and getting over whelmed this sample code layouts out the bare bones. Along with more research into the APIs and review of more complete samples of Bluetooth you will be on your way to making some exciting new Bluetooth games/aplications. More complete examples are available with developments kits such as:

- BluetoothDemo comes with the WTK 2.2 Beta (Sun)
- BluePad comes with the SDK from SonyEricsson
- BlueToothCar comes with the SDK from SonyEricsson
- Samples that comes with the Rococo Software SDK

Source Code

Download Source to Bluetooth Example (Click Here)

Development Kits

http://java.sun.com (Wireless Toolkit 2.2 Beta includes JSR 82)
http://forum.nokia.com (Nokia Developer Suite 2.2 includes JSR 82)
http://developer.sonyericsson.com (J2ME SDK 2.1.4 Beta includes JSR 82)
http://www.atinav.com
http://www.rococosoft.com
http://www.esmertec.com
http://www.smartnd.com
http://www.oi-us.com

Resources

http://jcp.org/aboutJava/communityprocess/final/jsr082/
http://www.bluetooth.com
http://www.bluetooth.org
http://opensource.nus.edu.sg/projects/bluetooth/
http://bluez.sourceforge.net/
http://benhui.net

Jason is a wireless and open source developer enthusiast who enjoys creating synergy and sharing knowledge in the software development world. To learn more about him visit his personal site at http://www.jasonlam604.com/

Share on Facebook

RELATED ITEMS BLUETOOTH

Jason Lam

About Jason Lam
Jason is a wireless and open source developer enthusiast who enjoys creating synergy and sharing knowledge in the software development world. To learn more about him visit his personal site at http://www.jasonlam604.com/
View all posts by Jason Lam →

Top Rated White Papers and Resources

Help Desk Software in the Cloud [Sign Up Free]

THERE ARE *3* COMMENTS. ADD YOURS.

chadngm
February 13, 2011 at 5:15 pm

0 0

hi,
try the following url :

http://www.pudn.com/downloads161/sourcecode/mobile/j2me/detail727017.html

(for the EchoClient, there is a bug in html code. But you can copie it from source code of the page)

Enjoy

Reply

Guest
January 17, 2011 at 4:51 am

0 0

Even i couldn't download the source code..
Pls send me the link soon;.....

Reply

manoj
October 4, 2010 at 2:26 pm

0 0

hi,

I can't download the source code with given link.please send me the working link to download the soure code immediately.

Thank you.

Reply

WHAT DO *YOU* THINK? *RESPOND.*

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

You may use these HTML tags and attributes: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

Post Comment

NEXT ARTICLE »

WebProNews
WebProNews is your comprehensive resource for news, information, and tips related to online business.

DevWebPro
DevWebPro is dedicated to bringing you the best developer information on the net.

GetIP
A free application that helps users find out the details of their Internet connection, as well as that of any other IP address or domain.

PRtracking
A free automated PageRank checking service.

Twellow
Twellow is a directory of public Twitter accounts, with hundreds of categories and search features to help you find people who matter to you.

Company
Corporate
Advertising
Sitemap
Newsletters
Privacy Policy
About Us
Contact Us

iEntry Network
Web Developers
IT Managers
Small Business Owners
eBusiness Management
Software
Gamers

Advertising
Why Advertise?
Who's Advertising?
Testimonials
Newsletter Samples
Ad Specs
Contact

Get to know us

iEntry Network, a business-to-business Web media services company, provides your pathway to over 6 million IT professionals, small business owners and ecommerce entrepreneurs, marketing professionals, industry bloggers, and Web-savvy media consumers.