How to obtain mobile IP address in Java ME

Introduction

In style of a wired host computer, a mobile phone does expose limited information to Internet servers. These include User-Agent tag and an IP address. This basically means a mobile device does actually not be treated any differently than it would be a home computer accessing the remote server.

For wired host computers IP address can be very useful for web services, in attempt to geolocate the computer and thus providing targeted local area information based on where the accessing computer has been detected. This has been quite successful, even the commonness of NAT (Network Address Translation) boxes and IP addresses created dynamically.

Locating mobile phones based on their IP address, on the other hand, is not as functional. First of the reasons for the difficulties on an accurate geolocation for mobiles are their IP being changing in somewhat fast manner during network requests (mainly HTTP).This can result in web service recognizing different address for the same mobile device even a fast pace in executing queries. Secondly the same or similar IP address may be shown towards the server from mobile devices located even very far from each others, so they may still share same IP-address space.

This simply means that the mobile device IP address will not provide any very pinpoint information of their whereabouts. Opposing to the mobile devices, wired computers IP addresses stay stationary with very high accuracy from the web service host point of view. Geolocation services may however implement reported cell tower IDs in together with the strength of radio signals that the mobile device receives from different cell towers to increase accuracy.

Obtaining the IP address by the method on how presented next in this Wiki article, does not try to compete with the GPS positioning (Global Positioning System), but exposes some possible good use cases. Most definitely the mobile device IP address bases on the NAT (Network Address Translation) performed by the gateway of the service provider. This will result a "masquerading" IP address, for which reason the mobile device is not visible to the outside world with that IP. Such IP address is called "private".

This will result e.g. so that the mobile device web browser in a "masqueraded network" can browse a website outside, but this does not work the other way. It depends on the network administrator if the translation table entries will be configured for permanent use, often referred to "static NAT" or port forwarding, thus allowing traffic from outside network to access into that "hidden" network.

Request IP address in J2ME

The example MIDlet works the way, that when started it will first ask the user permission:

```
"Allow application to use network and send or receive data?"
```

If the user either selects "Once" or "Allow for this session" we will further continue trying to open a connector using ServerSocketConnection:

```
ssc = (ServerSocketConnection) Connector.open("socket://:1234");
```

Note: On a PC mobile emulator or device where GPRS has been idling, you may be returned with the localhost IP address of 127.0.0.1. This is called "loopback". Depending on the settings on your mobile, if the GPRS has been set to "use when available", the network provider IP address should be returned with high probability. If available, WIFI network connection would work similar way.

With the ServerSocketConnection the mobile device will basically become a client and would be waiting for the server to respond.

In the code is commented out another query that could be also performed, namely the System.getProperty("microedition.hostname"). If you want to see results for that, and maybe find useful, uncomment these lines:

```
...
 //private String msg2;
```


(C) Copyright Nokia 2014. All rights reserved.

```
 //msg2 = System.getProperty("microedition.hostname");
 //form.append("Hostname by 'getproperty' is:\n");
...
```

On devices that support this it should return the host name of the device, most likely it will give string "localhost" as return value.

Code example

```
package MyIpAddress;

import java.io.IOException;
import javax.microedition.io.Connector;
import javax.microedition.io.ServerSocketConnection;
import javax.microedition.midlet.*;;
import javax.microedition.lcdui.*;

public class MyIPaddress extends MIDlet implements CommandListener{

    private Display display;
    private Form form;
    private Command exit;
    private String msg;
    private String msg2;
    private Alert modalAlert;

    public void startApp() {
        display = Display.getDisplay(this);
        exit = new Command("Exit", Command.EXIT, 1);
        form = new Form("MyIPaddress");
        form.addCommand(exit);
        form.setCommandListener(this);
        display.setCurrent(form);
        form.append("Requesting your IP now...\n");

        try {
            ServerSocketConnection ssc = null;
            try {
               ssc = (ServerSocketConnection) Connector.open("socket://:1234");
            } catch (IOException ex) {
                modalAlert = new Alert("Error", "Problem with connecting!",null,
AlertType.ERROR);
                modalAlert.setTimeout(4000);
                display.setCurrent(modalAlert);
                form.append("\nNo IP address received!\n");

                ex.printStackTrace();
            }
                msg = ssc.getLocalAddress();
                form.append("Success! Your IP is:\n");
                form.append(msg);
                msg2 = System.getProperty("microedition.hostname");
                form.append("\nHostname is:\n");
                form.append(msg2);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        } catch (IOException ex) {
                modalAlert = new Alert("Error", "Exiting application!",null,
AlertType.ERROR);
                modalAlert.setTimeout(4000);
                display.setCurrent(modalAlert);
                ex.printStackTrace();
        }
    }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
        notifyDestroyed();
  }

  public void commandAction(Command c, Displayable s){
        String label = c.getLabel();
        if(label.equals("Exit")){
        destroyApp(false);
   }
  }

}
```

Screenshot - Example run

Device: N97

(C) Copyright Nokia 2014. All rights reserved.

Important note

This MIDlet performs simple try-catches and also informs user only very briefly. This to keep the code close to short as possible, still providing at least some basic error handling. You may enhance your own code with even further check's and responses, if you like. In example, when the user does not allow connection to be established, the MIDlet will currently cleanly exit with message "Application error", "Application will be closed".

--Maveric 19:33, 26 May 2010 (UTC)

(C) Copyright Nokia 2014. All rights reserved.