



MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide

Document Type:	Manual
Reference:	MAN_NFC_1109-278 Version 0.3 (11446)
Release Date:	Oct. 3, 2011
File Name:	MAN_NFC_1109-278 MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide v0.3.pdf
Security Level:	General Business Use

MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide

General Business Use

Page : 2/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

Disclaimer

This document is licensed under the Creative Commons Attribution 3.0 license (<http://creativecommons.org/licenses/by/3.0/>). (You may use the content of this document in any way that is consistent with this license and if you give proper attribution (<http://www.open-nfc.org/license.html#attribution>)).

Copyright © 2011 Inside Secure

Open NFC and the Open NFC logo are trademarks or registered trademarks of Inside Secure.

Other brand, product and company names mentioned herein may be trademarks, registered trademarks or trade names of their respective owners.

**MAN_NFC_1109-278 Open NFC -
Android Bluetooth WiFi Handover - user
guide**

General Business Use

Page : 3/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

History

Version	Date	Comments
0.1	Sept. 30, 2011	Draft
0.3	Oct. 3, 2011	Add Security issue for handover wifi connection

Summary of Contents

1	Introduction.....	6
2	Use of NFC handover module.....	7
3	Frequently Asked Questions (FAQ)	13

Reference Documents

- [1] NFC Forum Connection Handover Technical Specification V1.2. 2010-07-07. NFC Forum, Inc.

1 Introduction

NFC handover protocol enables two NFC-enabled devices, via initial NFC exchanges of carriers' information (Bluetooth/Wifi) by one simple touch, to establish a connection using other wireless communication technologies, such as WiFi or Bluetooth, which allows significantly extending the range and capacity of wireless transmission.

NFC handover module has been integrated in the OpenNFC middleware according to the standard *Connection Handover Technical Specification NFC Forum* [1]. This documentation gives a brief introduction on how to develop NFC handover application based on the OpenNFC middleware.

NFC handover module which has been integrated in the OpenNFC middleware includes the following functionalities: 1) Exchange of candidate carrier information between two NFC enabled devices; 2) Requester and Selector definition; 3) Carrier Priority Negotiation between the two devices. All this information will then be returned to the application layer.

The following section will give a detailed introduction on how to fetch this information for potential use.

2 Use of NFC handover module

This section gives a brief introduction on how to develop NFC handover application based on the NFC handover module integrated in the OpenNFC middleware.

The application developers are recommended to read [1] to be familiar of several related notation of NFC handover procedure.

NFC handover module allows exchanging and feedback the carrier information (*Wifi/Bluetooth*) of the corresponding device, the Requester and Selector definition (*whether the current device is Requester or Selector*) and Carrier Priority to use (*which carrier is priority for the future connection*).

Note that the establishment of future connections (*Wifi, Bluetooth connection or other transport connection*) is done at the application layer according to the feedback of the carrier information instead of being done directly in the NFC handover module; the reason is that such way allows satisfying wider user cases and gives more flexibility for the application scenarios (i.e. application may flexibly switch between Wifi and Bluetooth connection according to signal, battery or other conditions).

Three steps should be followed to develop the NFC handover application.

- 1) Some parameters should be sent to the OpenNFC middleware by `Broadcast` to launch the NFC handover procedure. The parameters include:
 - Number of available carriers, which represents the number of carriers that are allowed to use. This parameter is defined by user, the value of this parameter is limited to: "1" (one carrier is allowed, Bluetooth or Wifi); and "2" (both Bluetooth and Wifi are allowed).
 - Priority carrier, which represents the priority carrier that the user defines. The value of this parameter should be either "bluetooth" or "wifi" (case insensitive).
 - A random generated UUID in String format should be sent to the corresponding device by NFC touch, which allows establishing the potential Bluetooth connection with the corresponding device.
 - Two parameters: SSID and random generated security key should be sent to the corresponding device by NFC touch, which allows validating and establishing the potential wifi connection.

These parameters should be inserted in an `Intent`, which will be sent to OpenNFC middleware. Example code is as follows.

Create a new `Intent intentToBeReturned` which includes all the mentioned parameters .

```
Intent intentToBeReturned = new Intent().
    setAction("org.opennfc.intent.action.HANDOVER_MSG_RETURNED");
intentToBeReturned.putExtra("FROM_APP_NBCARRIER", nbCarrier);
intentToBeReturned.putExtra("FROM_APP_PRIORITYCARRIER", priorityCarrier);
intentToBeReturned.putExtra("FROM_APP_UUID", uuidString); //Generated by
UUID.randomUUID().toString();
intentToBeReturned.putExtra("FROM_APP_SSID", ssid);
intentToBeReturned.putExtra("FROM_APP_KEY", securityKey); //Should be Auto-
generated
```

`intentToBeReturned` is inserted in another `Intent "intentToBeSent"` which is to be sent to OpenNFC middleware by `Broadcast`.

MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide

General Business Use

Page : 8/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

Note that the `intentToBeSent` must setAction as `"org.opennfc.intent.action.HANDOVER_START"`, which is also configured in OpenNFC middleware to receive the broadcast message from application layer. Example code is as follows.

```
Intent intentToBeSent = new  
Intent().setAction("org.opennfc.intent.action.HANDOVER_START")  
.putExtra("INTENT", intentToBeReturned);
```

- 2) After the touch of two NFC enabled device, the feedback (*carrier information of corresponding device, Requester/ Selector definition Carrier Priority*) is then returned to application by another Broadcast from NFC handover module.

The feedback information is inserted in `intentToBeReturned` by NFC handover module. When `intentToBeReturned` is received at application layer, we can get all the feedback information by the method `"getStringExtra (String NAME)"`

Here is the attribute **NAME** corresponding to the feedback parameters:

TO_APP_RESULT: represents the handover result. `"HANDOVER_OK"` means the handover is finished with success. If handover procedure fails, the error will be returned in this attribute.

TO_APP_STATUS: return whether the current device is a `"REQUESTER"` or `"SELECTOR"`

TO_APP_NBCARRIER: number of the available carriers with which to establish the future connections (`"0"`: no carrier is available; `"1"`: only one carrier based connection (either WiFi or Bluetooth connection) is available between the two devices; `"2"` two carrier based connection (WiFi and Bluetooth connection) are available between the two devices).

TO_APP_PRIORITYCARRIER: priority carrier. If `"wifi"` is returned (case insensitive). WiFi connection should be priority to be established. If `"bluetooth"` is returned (case insensitive). Bluetooth connection should be priority to be established.

TO_APP_SSID: the SSID set by the corresponding device if the corresponding device serves as an access point.

TO_APP_KEY: the security key set by the corresponding device, this information is useful in case if the current device will connect to the corresponding device which serves as a WiFi access point.

TO_APP_BTNAME: the Bluetooth name of the corresponding device.

TO_APP_UUID: the UUID generated by the corresponding device, this information is useful in case if the current device will connect to the corresponding device which serves as Bluetooth server.

TO_APP_BTADDRESS: the Bluetooth device address of the corresponding device.

Here is a completed example at the application layer to launch the handover procedure and get the handover feedback information:

```
package org.opennfc.handover.app;
```

```
import java.util.UUID;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.content.BroadcastReceiver;
```

```
import android.content.Context;
```

```
import android.content.Intent;
```

```
import android.content.IntentFilter;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
import android.widget.TextView;
```

```
public class HandoverAppActivity extends Activity {  
    /** Called when the activity is first created. */
```

```
    private BroadcastReceiver receiver;
```

MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide

General Business Use

Page : 9/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final Button b1 = (Button) findViewById(R.id.button1);
    b1.setText("Start Handover");
    b1.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            b1.setEnabled(false);
            System.out.println("Click");

            String nbCarrier = "2";
            String priorityCarrier = "bluetooth";
            String ssid = "InsideShare";
            String securityKey = "password";// TODO: to be generated by
//
random function

            UUID uuid = UUID.randomUUID();
            String uuidString = uuid.toString();

            Intent intentToBeReturned = new Intent()

                .setAction("org.opennfc.intent.action.HANDOVER_MSG_RETURNED");
            intentToBeReturned.putExtra("FROM_APP_NBCARRIER",
nbCarrier);

            intentToBeReturned.putExtra("FROM_APP_PRIORITYCARRIER",
priorityCarrier);

            intentToBeReturned.putExtra("FROM_APP_UUID", uuidString);
            intentToBeReturned.putExtra("FROM_APP_SSID", ssid);
            intentToBeReturned.putExtra("FROM_APP_KEY", securityKey);

            Intent intentToBeSent = new Intent().setAction(
"org.opennfc.intent.action.HANDOVER_START").putExtra(
                "INTENT", intentToBeReturned);
            sendBroadcast(intentToBeSent);
        }
    });

    IntentFilter filter = new IntentFilter();
    filter.addAction("org.opennfc.intent.action.HANDOVER_MSG_RETURNED");

    receiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            TextView tv = (TextView) findViewById(R.id.textView1);
            b1.setEnabled(true);
            if ("org.opennfc.intent.action.HANDOVER_MSG_RETURNED"
                .equals(intent.getAction()) == true) {
                if (intent.getStringExtra("TO_APP_RESULT")
                    .equalsIgnoreCase("HANDOVER_OK")) {
                    String SSID = intent.getStringExtra("TO_APP_SSID");
```

MAN_NFC_1109-278 Open NFC - Android Bluetooth WiFi Handover - user guide

General Business Use

Page : 10/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

```
String Key = intent.getStringExtra("TO_APP_KEY");
String BTLocalName = intent
    .getStringExtra("TO_APP_BTNAME");
String BTUUIDList = intent
    .getStringExtra("TO_APP_UUID");
String BtAddress = intent

    .getStringExtra("TO_APP_BTADDRESS");
String NbCarrier = intent

    .getStringExtra("TO_APP_NBCARRIER");
String PriorityCarrier = intent

    .getStringExtra("TO_APP_PRIORITYCARRIER");
String Status =
intent.getStringExtra("TO_APP_STATUS");
+ "-- "
    System.out.println(" Status: " + Status + "-- "
        + SSID + "-- " + "-- " + Key + "-- "
        + BTLocalName + "-- " + BTUUIDList
        + BtAddress
        + " Available carrier number is : "
        + NbCarrier + " *** Priority carrier is: "
        + PriorityCarrier);
tv.setText(Status + "-- " + SSID + "-- " + Key
    + "-- " + BTLocalName + "-- " +
    + "-- " + BtAddress
    + "**** Available carrier number is : "
    + NbCarrier + " *** Priority carrier is: "
    + PriorityCarrier);
} else {
    System.out.println(" ERROR Handover result");
}
tv.setText(intent.getStringExtra("TO_APP_RESULT"));
}
}
};
registerReceiver(receiver, filter);
}
protected void onDestroy() {
    unregisterReceiver(receiver);
    super.onDestroy();
}
}
```

3) After fetching the NFC feedback parameters, available carrier based connection can then be established based on these exchanged information.

- Bluetooth connection: the returned UUID will be used to establish the Bluetooth connection. A good tutorial for Bluetooth connection is in the Android developer Framework Topics

<http://developer.android.com/guide/topics/wireless/bluetooth.html>

- WiFi connection: We define that the “SELECTOR” always serves as an Access point, and the “REQUESTER” always try to actively establish the WiFi connection with the “SELECTOR”. “SELECTOR” can set an AP using the method “[setWifiApEnabled](#)”. “REQUESTER” can use the method “enableNetwork (networkId, true)” and “reconnect()” to connect to the “SELECTOR”.

REQUESTER: to actively connect to Access Point:

```
mWifiManager = (WifiManager) mContext
    .getSystemService(Context.WIFI_SERVICE);
WifiConfiguration config = new WifiConfiguration();

config.SSID = convertToQuotedString(ssidName);
config.preSharedKey = convertToQuotedString(securityKey); //securityKey in String
config.hiddenSSID = true;
config.status = WifiConfiguration.Status.ENABLED;
config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);
config.allowedProtocols.set(WifiConfiguration.Protocol.RSN);
int networkId = mWifiManager.addNetwork(config);
mWifiManager.enableNetwork(networkId, true);
mWifiManager.saveConfiguration();
mWifiManager.reconnect();

static String convertToQuotedString(String string) {
    return "\"" + string + "\"";}
```

SELECTOR: Servers as an access point, wait to be connected by REQUESTER.

Note that the method “[setWifiApEnabled](#)” is hidden by Android, apply the following code to use this method from application layer.

```
mWifiManager = (WifiManager) mContext
    .getSystemService(Context.WIFI_SERVICE);
WifiConfiguration config = new WifiConfiguration();
config.SSID = convertToQuotedString(ssidName);
config.preSharedKey = securityKey;
config.status = WifiConfiguration.Status.ENABLED;
config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
config.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);
config.allowedProtocols.set(WifiConfiguration.Protocol.RSN);

Method method = mWifiManager.getClass().getMethod
    ("setWifiApEnabled", WifiConfiguration.class, boolean.class);
return (Boolean) method.invoke(mWifiManager, config, enabled);
```

**MAN_NFC_1109-278 Open NFC -
Android Bluetooth WiFi Handover - user
guide**

General Business Use

Page : 12/13

Date : Oct. 3, 2011

Ref. : MAN_NFC_1109-278 v0.3(11446)

3 Frequently Asked Questions (FAQ)

- **I have the returned error: “Handover Error: handoverPairingStartError: 22.”**
 - Two NFC enabled devices should touch to finish the handover procedure, if one of the NFC device removes away too quickly, this error can be happened. It is enough to re-launch the handover procedure.

- **Bluetooth should be turned on before the handover procedure?**
 - If the user add “Bluetooth” in the candidate carrier list (which means if “2” is set for “[FROM_APP_NBCARRIER](#)”, or “Bluetooth” is set for “[FROM_APP_PRIORITYCARRIER](#)”), Bluetooth should be turned on before the handover procedure.
Error “Handover Error: Please make sure that Bluetooth is on” will be returned if Bluetooth is off.