

NFC

Introduction

NFC (*Near Field Communication*) est une technologie de communication sans fil (courte portée et haute fréquence à 13,56 MHz) qui permet de transmettre de petites quantité de données sur des distances d'environ quelques cm (5 à 10 cm max). Cette technologie est une extension de la norme ISO/CEI 14443 utilisant la radio-identification (RFID).

Il existe trois modes de fonctionnement :

- le mode [émulation de carte](#) : l'équipement se comporte comme un *tag* passif et autorise une lecture par un lecteur externe.
- le mode lecteur : la lecture et l'écriture d'un *tag* passif suivant différentes technologies.
- le mode pair à pair (P2P) : l'échange de données (cf. [Android Beam](#))

NDEF (*NFC Data Exchange Format*) définit le format d'échange des données.

L'API 9 (Android 2.3) a introduit le support [NFC](#). Android propose une [API spécifique pour chaque technologie](#).

Principe

Lorsque le capteur NFC de l'appareil mobile détecte un *tag*, un *intent* (un message asynchrone) sera diffusé.

Trois *intent* sont définis par le système par ordre de priorité :

- `ACTION_NDEF_DISCOVERED` : un *tag* contenant des données NDEF a été détecté et cet *intent* est utilisé pour lancer une activité qui utilise un `intent-filter` de ce type afin de traiter ces données (*intent* le plus prioritaire).
- `ACTION_TECH_DISCOVERED` : cet *intent* est diffusé si le précédent n'a pas été géré. Il est aussi lancé si le *tag* ne possède pas de type MIME reconnu, d'URI ou de données NDEF.
- `ACTION_TAG_DISCOVERED` : cet *intent* est diffusé si les deux précédents n'ont pas été gérés.

AndroidManifest.xml

Avant toute chose, il faut donner une autorisation matérielle pour le capteur NFC, indiquer la version minimale du SDK (API 9 ou API 10 qui prend en charge complétement la lecture/écriture ou API 14) et prévenir d'une utilisation obligatoire avec `uses-feature` .

Il faut ensuite déclarer les `intent-filter` . Si l'activité filtre pour l'*intent* ACTION_TECH_DISCOVERED, il faudra créer un fichier de ressources XML `nfc_tech_filter.xml` (dans `res/xml/`) qui précise les (une, deux ou trois) technologies prises en charge parmi la liste suivante :

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <tech-list>
        <tech>android.nfc.tech.IsoDep</tech>
        <tech>android.nfc.tech.NfcA</tech>
        <tech>android.nfc.tech.NfcB</tech>
        <tech>android.nfc.tech.NfcF</tech>
```

```

<tech>android.nfc.tech.NfcV</tech>
<tech>android.nfc.tech.Ndef</tech>
<tech>android.nfc.tech.NdefFormattable</tech>
<tech>android.nfc.tech.MifareClassic</tech>
<tech>android.nfc.tech.MifareUltralight</tech>
</tech-list>
</resources>

```

Remarque : il est également possible de spécifier plusieurs ensembles de listes `<tech-list>`.

Le fichier `AndroidManifest.xml` sera alors :

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplicationnfc">

    <uses-permission android:name="android.permission.NFC" />
    <uses-sdk android:minSdkVersion="10"/>
    <uses-feature
        android:name="android.hardware.nfc"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data android:mimeType="text/plain" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.nfc.action.TECH_DISCOVERED"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
            <meta-data android:name="android.nfc.action.TECH_DISCOVERED"
                android:resource="@xml/nfc_tech_filter" />
        </activity>
    </application>
</manifest>

```

Remarque : la version minimum du SDK (`minSdkVersion`) peut être indiquée dans le fichier `build.gradle`.

Tests

Android fournit tout d'abord la classe [NfcAdapter](#) qui représente le lecteur NFC de l'appareil mobile. On le récupère en appelant la méthode `getDefaultAdapter()`.

On va lire un badge [MifareClassic](#).

```
public class MainActivity extends AppCompatActivity
{
    private final String TAG = "NFC";
    private NfcAdapter nfcAdapter = null;
    private PendingIntent nfcPendingIntent = null;
    private boolean nfcEstPresent = false;
    private boolean nfcEstActif = false;
    private TextView texteMessageTypeNFC;
    private TextView texteMessageIdNFC;
    private TextView texteMessageDonneesNFC;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        texteMessageTypeNFC = (TextView) findViewById(R.id.texteMessageTypeNFC);
        texteMessageIdNFC = (TextView) findViewById(R.id.texteMessageIdNFC);
        texteMessageDonneesNFC = (TextView) findViewById(R.id.texteMessageDonneesNFC);

        texteMessageTypeNFC.setText("Scannez un badge ...");

        nfcAdapter = NfcAdapter.getDefaultAdapter(this);
        if(nfcAdapter == null)
        {
            Toast.makeText(getApplicationContext(), "Pas de NFC !", Toast.LENGTH_LONG).show();
            nfcEstPresent = false;
            nfcEstActif = false;
        }
        else if (nfcAdapter != null && !nfcAdapter.isEnabled())
        {
            Toast.makeText(getApplicationContext(), "NFC désactivé !", Toast.LENGTH_LONG).show();
            nfcEstPresent = true;
            nfcEstActif = false;
        }
        else
        {
            Log.v(TAG, "NFC ok");
            nfcEstPresent = true;
            nfcEstActif = true;
        }
    }

    @Override
    protected void onResume()
    {
        super.onResume();

        if(nfcEstPresent && nfcEstActif)
        {
            nfcPendingIntent = PendingIntent.getActivity(this, 0, new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
            nfcAdapter.enableForegroundDispatch(this, nfcPendingIntent, null, null);
        }
    }
}
```

```

        // ou :
        //demarrer();
    }

}

@Override
protected void onPause()
{
    super.onPause();

    if(nfcEstPresent && nfcEstActif)
    {
        nfcAdapter.disableForegroundDispatch(this);
    }
}

@Override
public void onNewIntent(Intent intent)
{
    String action = intent.getAction();

    if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action) || NfcAdapter.ACTION_TECH_DISCOVERED.
equals(action) || NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action))
    {
        Log.v(TAG, "action := " + intent.getAction());

        if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action))
        {
            // cf. intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
        }
        else if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action))
        {
            // cf. intent.getParcelableArrayExtra(NfcAdapter.EXTRA_TAG);
        }
        else if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(action))
        {
            Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);

            String type = "";
            texteMessageTypeNFC.setText(type);
            texteMessageIdNFC.setText("");
            texteMessageDonneesNFC.setText("");

            for (String tech : tagFromIntent.getTechList())
            {
                Log.v(TAG, "type := " + tech);
                if(tech.equals("android.nfc.tech.NfcA"))
                {
                    type += " NFC-A ISO 14443-3A";
                }
                if(tech.equals("android.nfc.tech.NfcB"))
                {
                    type += " NFC-B ISO 14443-3B";
                }
                if(tech.equals("android.nfc.tech.NfcV"))
                {
                    type += " NFC-V ISO 15693";
                }
                if(tech.equals("android.nfc.tech.IsoDep"))
                {

```

```

        type += " ISO-DEP ISO 14443-4";
    }
    if(tech.equals("android.nfc.tech.MifareClassic"))
    {
        Toast.makeText(getApplicationContext(), "Badge Mifare", Toast.LENGTH_LONG
).show();
        type += " MifareClassic";
        String informations = lireMifare(tagFromIntent);
        texteMessageDonneesNFC.setText(informations);
    }
}
if(!type.isEmpty())
{
    texteMessageTypeNFC.setText(type);
    String uuid = ByteArrayToHexString(intent.getByteArrayExtra(NfcAdapter.EXTRA_
ID));
    texteMessageIdNFC.setText("UUID : " + uuid);
}
}

private String lireMifare(Tag tag)
{
    MifareClassic mif = MifareClassic.get(tag);

    String informations = "";
    int type = mif.getType();
    switch (type)
    {
        case MifareClassic.TYPE_CLASSIC:
            informations += "Type : MIFARE Classic\n";
            break;
        case MifareClassic.TYPE_PLUS:
            informations += "Type : MIFARE Plus\n";
            break;
        case MifareClassic.TYPE_PRO:
            informations += "Type : MIFARE Pro\n";
            break;
        case MifareClassic.TYPE_UNKNOWN:
            informations += "Type : MIFARE Classic compatible\n";
            break;
    }

    int size = mif.getSize();
    informations += "Taille : " + size + "\n";

    int nbSecteurs = mif.getSectorCount();
    informations += "Nb secteurs : " + nbSecteurs + "\n";

    int nbBlocs = mif.getBlockCount();
    informations += "Nb blocs : " + nbBlocs + "\n";

    try
    {
        mif.connect();
        if (mif.isConnected())
        {
            for(int i=0; i< nbSecteurs; i++)
            {

```

```

        boolean isAuthenticated = false;

        if (mif.authenticateSectorWithKeyA(i, MifareClassic.KEY_MIFARE_APPLICATION_DIRECTORY))
        {
            isAuthenticated = true;
        }
        else if (mif.authenticateSectorWithKeyA(i, MifareClassic.KEY_DEFAULT))
        {
            isAuthenticated = true;
        }
        else if (mif.authenticateSectorWithKeyA(i,MifareClassic.KEY_NFC_FORUM))
        {
            isAuthenticated = true;
        }
        else
        {
            //Log.d("TAG", "Autorisation := refusée");
        }

        if(isAuthenticated)
        {
            int index = mif.sectorToBlock(i);
            byte[] block = mif.readBlock(index);
            String s_block = MainActivity.ByteArrayToHexString(block);
            if(block_index < 10)
                informations += "bloc " + index + " : " + s_block + "\n";
            else
                informations += "bloc " + index + " : " + s_block + "\n";
        }
        else
            informations += "Aucune autorisation\n";
    }
}

mif.close();
}
catch (IOException e)
{
    e.printStackTrace();
}
return informations;
}

private void demarrer()
{
    final String[][] techList = new String[][] {
        new String[] {
            NfcA.class.getName(),
            NfcB.class.getName(),
            NfcF.class.getName(),
            NfcV.class.getName(),
            IsoDep.class.getName(),
            MifareClassic.class.getName(),
            MifareUltralight.class.getName(),
            Ndef.class.getName()
        }
    };
}

nfcPendingIntent = PendingIntent.getActivity(this, 0, new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

```

```

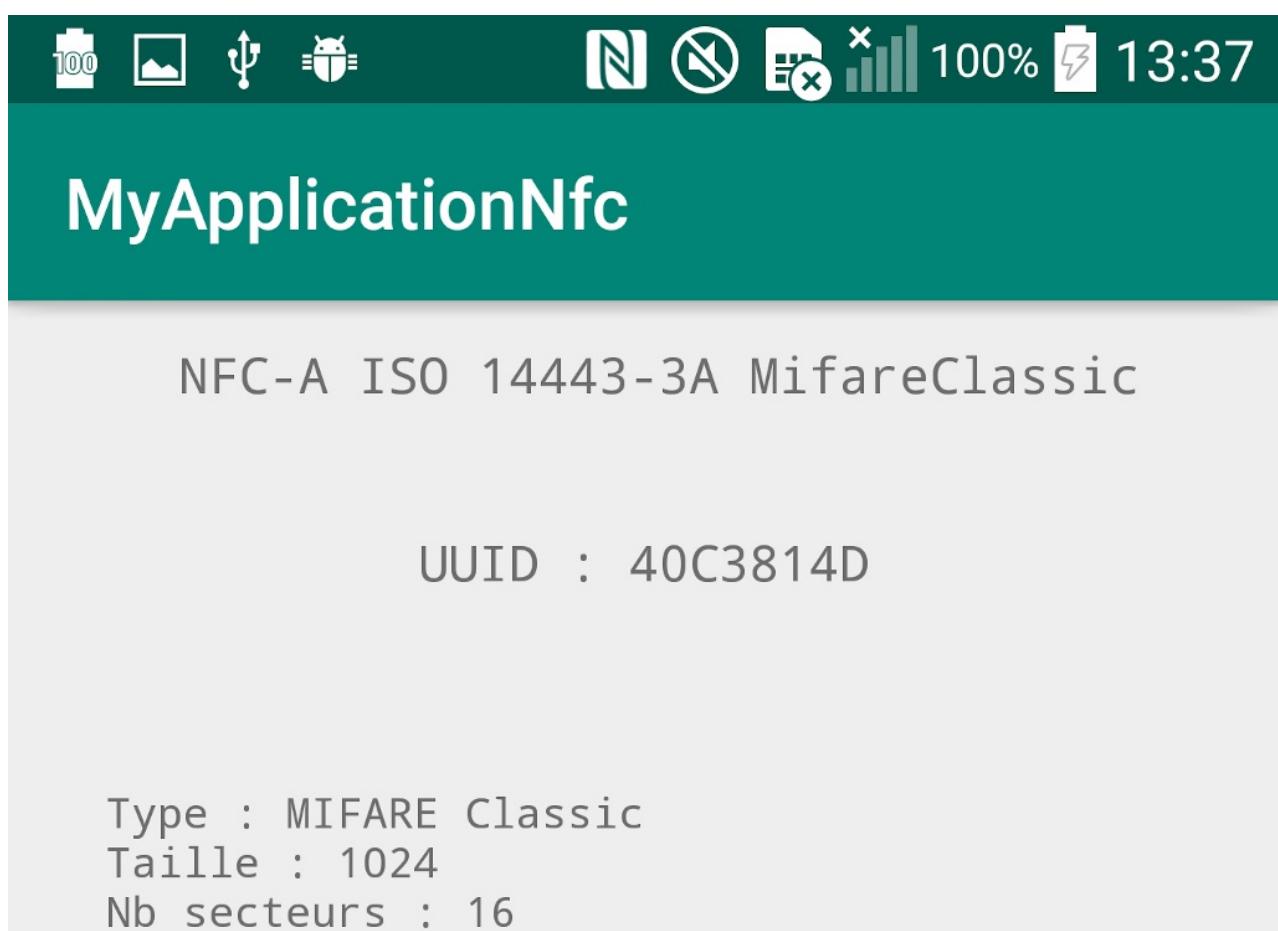
IntentFilter filter = new IntentFilter();
filter.addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
filter.addAction(NfcAdapter.ACTION_TECH_DISCOVERED);
filter.addAction(NfcAdapter.ACTION_TAG_DISCOVERED);

nfcAdapter.enableForegroundDispatch(this, nfcPendingIntent, new IntentFilter[]{filter}, techList);
}

private static String ByteArrayToHexString(byte[] inarray)
{
    int i, j, in;
    String[] hex = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F"};
    String out = "";

    for (j = 0; j < inarray.length; ++j)
    {
        in = (int) inarray[j] & 0xff;
        i = (in >> 4) & 0x0f;
        out += hex[i];
        i = in & 0x0f;
        out += hex[i];
    }
    return out;
}
}

```



```
Nb blocs : 64
bloc 0 : 40C3814D4F880400000000000000000000000000000000
bloc 4 : 0000000000000000000000000000000000000000000000000
bloc 8 : 0000000000000000000000000000000000000000000000000
bloc 12 : 0000000000000000000000000000000000000000000000000
bloc 16 : 0000000000000000000000000000000000000000000000000
bloc 20 : 0000000000000000000000000000000000000000000000000
bloc 24 : 0000000000000000000000000000000000000000000000000
bloc 28 : 0000000000000000000000000000000000000000000000000
bloc 32 : 0000000000000000000000000000000000000000000000000
bloc 36 : 0000000000000000000000000000000000000000000000000
bloc 40 : 0000000000000000000000000000000000000000000000000
bloc 44 : 0000000000000000000000000000000000000000000000000
bloc 48 : 0000000000000000000000000000000000000000000000000
bloc 52 : 0000000000000000000000000000000000000000000000000
bloc 56 : 0000000000000000000000000000000000000000000000000
bloc 60 : 0000000000000000000000000000000000000000000000000
```



Pour écrire des données sur un badge [MifareClassic](#), on pourra procéder de la manière suivante :

```
private void ecrireMifare(Tag tag, int numeroSecteur, String message)
{
    MifareClassic mif = MifareClassic.get(tag);
    try
    {
        mif.connect();
        if (mif.isConnected())
        {
            if (mif.authenticateSectorWithKeyA(numeroSecteur, MifareClassic.KEY_DEFAULT))
            {
                if (!message.isEmpty() && message.length() <= MifareClassic.BLOCK_SIZE)
                {
```

```
        byte[] datas = message.getBytes();
        mif.writeBlock(mif.sectorToBlock(numeroSecteur), datas);
    }
}
mif.close();
}
catch (IOException e)
{
    e.printStackTrace();
}
}
```

© Thierry Vaira <tvaira@free.fr>