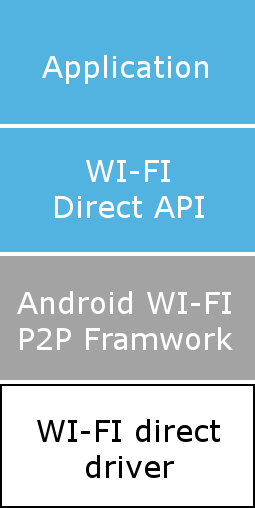
# Android WI-FI P2P

Since Android 4.0, devices with appropriate hardware are allowed to connect directly to each other over WI-FI P2P without an access point between them. Android P2P framework complies with the WI-FI Alliance’s WI-FI Direct certification program. With the usage of this API you are able to discover and connect to other devices when they support WI-FI P2P. According to documentations the advantage of WI-FI P2P beside Bluetooth or similar connection types is a fast connection across distances much longer than others. This allows applications a fast exchange of data between multiple users, which could be useful for applications such as multiplayer games, photosharing applications and in general, all applications which are relying on a fast connection between a long distance.

The main parts of the Androids WI-FI P2P APIs offer following methods and features.

* Methods for discovering, requesting and connecting to peers.
* Listeners which allows your application to get notified about the results of important WI-FI P2P methods, like discover, request, connect.
* Intents which notifies your application of specific events detected by the WI-FI P2P framework.

The picture below describes a typical Android WI-FI P2P architecture.



**Android Application and API**

**WI-FI direct package**

**Available since Android 4.0. Preinstalled with the WI-FI P2P packages and drivers.**

**Chip vender**

Figure 1: Android WI-FI P2P architecture

## Android API Description

In general the “WifiP2pManager” class provides all methods which allow your application to interact with the WI-FI hardware on your device. This class offers following actions to establish a connection:

|  |  |
| --- | --- |
| Method | Description |
| initialize() | This method registers the application with the WI-FI hardware. It is necessary to call this method before any other “WifiP2pManager” method. |
| connect() | This method starts a P2P connection with another device. |
| cancelConnect() | Cancels all running connection processes. |
| requestConnectInfo() | Requests a device connection Info |
| createGroup() | Creates a P2P group with the caller device as Group owner. |
| removeGroup() | Removes the current P2P group. |
| requestGroupInfo() | Requests the P2P a group info |
| discoverPeers() | Calling this method initiates a peer discovery. |
| requestPeers() | Requests the current list discovered peers. |

All the “WifiP2pManager” methods let you pass in a listener object. These listeners can be notified by the WI-FI P2P framework about the status of a call. The below table describes the available listeners and the corresponding “WifiP2pManager” methods which use the listeners.

|  |  |
| --- | --- |
| Listeners | Associated methods |
| WifiP2pManager.ActionListener | connect(), cancelConnect(), createGroup(), removeGroup(), discoverPeers() |
| WifiP2pManager.ChannelListener | initialize() |
| WifiP2pManager.ConnectionInfoListener | requestConnectInfo() |
| WifiP2pManager.GroupInfoListener | requestGroupInfo() |
| WifiP2pManager.PeerListListener | requestPeers() |

Furthermore, Androids WI-FI P2P APIs offers intents which notify your application when certain WI-FI P2P events happen, for example when new peers are discovered or the WI-FI state changes. If you want to use these intents it is necessary to create a broadcast receiver in your Android application that handles the specific intents. The following Code example shows how to implement a simple a broadcast receiver class in your application.

/\*\*  
 \* A BroadcastReceiver that notifies of important Wi-Fi p2p events.  
 \*/  
public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {  
  
    private WifiP2pManager mManager;  
    private Channel mChannel;  
    private MyWiFiActivity mActivity;  
  
    public WiFiDirectBroadcastReceiver(WifiP2pManager manager, Channel channel,  
            MyWifiActivity activity) {  
        super();  
        this.mManager = manager;  
        this.mChannel = channel;  
        this.mActivity = activity;  
    }

// The onReceive method handles the specific intents  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        String action = intent.getAction();  
  
        if (WifiP2pManager.WIFI\_P2P\_STATE\_CHANGED\_ACTION.equals(action)) {  
            // Check to see if Wi-Fi is enabled and notify appropriate activity  
        } else if (WifiP2pManager.WIFI\_P2P\_PEERS\_CHANGED\_ACTION.equals(action)) {  
            // Call WifiP2pManager.requestPeers() to get a list of current peers  
        } else if (WifiP2pManager.WIFI\_P2P\_CONNECTION\_CHANGED\_ACTION.equals(action)) {  
            // Respond to new connection or disconnections  
        } else if (WifiP2pManager.WIFI\_P2P\_THIS\_DEVICE\_CHANGED\_ACTION.equals(action)) {  
            // Respond to this device's wifi state changing  
        }  
    }  
}

The table below shows all available WI-FI P2P intents in android and a short description when the specific intents get notified.

|  |  |
| --- | --- |
| Intents | Description |
| WIFI\_P2P\_CONNECTION\_CHANGED\_ACTION | Get notified when the device’s WI-FI P2P state changes. |
| WIFI\_P2P\_PEERS\_CHANGED\_ACTION | Get notified when “discoverPeers()” is called. In this Intent you usually call “requestPeers()” and pass in a “PeerListListener” which get then a list of available peers. |
| WIFI\_P2P\_STATE\_CHANGED\_ACTION | Get notified when the WI-FI P2P is enabled or disabled on the device. |
| WIFI\_P2P\_THIS\_DEVICE\_CHANGED\_ACTION | Get notified when some specific device details have changed, like the device name. |

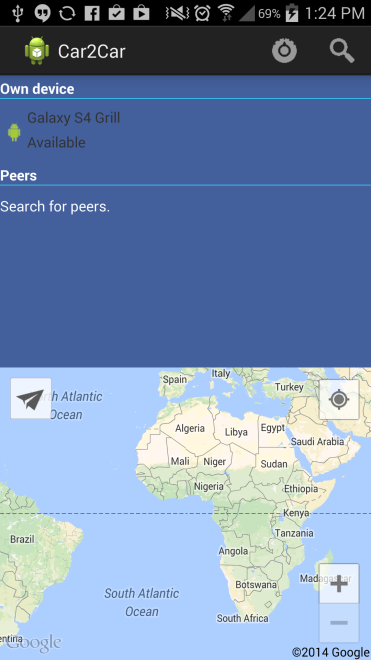
If you want to use any feature of Androids WI-FI P2P APIs you have to check if your device can access the hardware and supports the WI-FI P2P protocol. If your device fulfils all of these requirements you are able to obtain an instance of the “WifiP2pManager”, create a broadcast receiver and use Androids WI-FI P2P APIs. Furthermore it is necessary that your application have the privileges to interact with the WI-FI hardware and have the minim supported SDK available. For this you have to register the minimum supported API and the permission in the applications Android manifest. The below code snipped shows the minimum requirements, which have to register in the manifest file, for using androids WI-FI P2P APIs.

<uses-sdk android:minSdkVersion="14" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

## Android sample application

In regard to the Car2Car project an Android application which tests the reliability and the functions of the WI-FI P2P APIs was developed. In light of the idea behind the Car2Car project and the ability of modern Android phones, to track the location of a user, this subchapter will show the results of the simple WI-FI P2P and GPS prototype.

The simple prototype should discover available peers, after a successful connection it should send the GPS location of the user to all connected peers. All peers should mark the position of the other devices on the included google maps map with a marker. The picture below shows the design of the prototype application and describes the different sections.



Buttons for searching and enabling WI-FI direct on device.

Own device information

and connection status.

List for available. peers

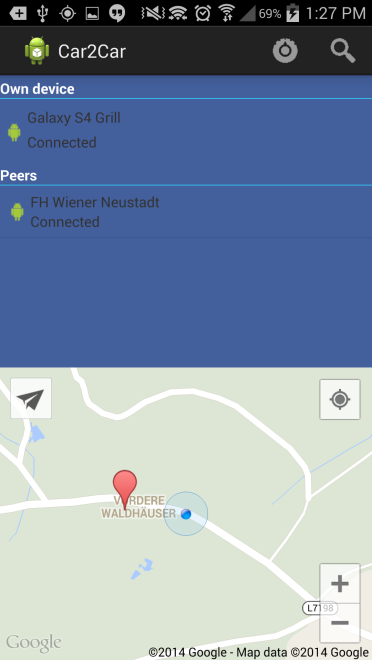
Buttons for searching and enabling WI-FI direct on device.

Custom button for

manually sending user location to other devices.

Figure 2: Design and description of the android prototype application

When the application is started is automatically begins to search for available peers additionally, the user can search manually by pressing the search button in top of the application. The application filters available peers per instance name, which means only other Car2Car peers are shown. If peers are available they appear immediately in the list view. A user can connect to other devices by clicking on the specific item in the list view. If the connection attempt was successful the invited device will get connection invitation which the user has to accept. If the other user has accepted the invitation the two application shares their GPS location information. This means, every time the location of the device will change the current GPS information (Longitude and Latitude) will send to all connected peers. In some regions with weak GPS signals it could took several minutes until the application will shares their GPS information with the other connected peers. The result of a successful connection is shown in the image below. The position of the other device is marked with a red google marker icon.



Own position and position of the other device.

Available device withe the current connection status.

Figure 3: Application with connected device