Bump Android API v0.4 Documentation

Bump Technologies Inc.

November 2010

1 Introduction

The Bump™ Android API lets your program use Bump™ to connect two Android devices or an Android and an iPhone/iPod Touch device. After the two devices are connected, a mailbox facility is provided that allows you to send data chunks between your applications. Multiple chunks may be sent and the connection can persist for some time (e.g. up to one hour).

2 Installing the Bump™ API

Before using the Bump™ API, you must add the Bump API library and the resources used by the API to your project. You will also need to add the API's activity and permissions to your `AndroidManifest.xml`.

2.1 Installing the library and resources

Before installing the Bump API, make sure to backup your project files. If you are using Mac OS, please see the note below before beginning.

To install the library and resources, simply unzip `BumpAndroidAPIv0.4.zip` into your project directory and merge into existing directories where necessary. In Windows, answer "Yes to All" to confirm folder replacement, which will merge folders. In Mac OS, please see the note below. If you are using Eclipse, make sure to refresh your project so that the Bump API files are included. You can do this by right-clicking the project name in Package Explorer and selecting "Refresh".

Finally, you will need to import your project's `R` class into the `BumpResources` class. You can do this by opening src/com/bumptech/bumpapi/BumpResources.java and adding `import <your project package>.R;` in the imports secction near the beginning of the file.

Note
Due to the way Mac OS merges folders, using the Finder to unzip the files will not merge the directories in the desired manner. You can either run `unzip` from the terminal,

```
unzip BumpAndroidAPIv0.4.zip -d <your project directory>/
```

or, if you have already unzipped the files somewhere else, `rsync` can provide the proper merging behavior,

```
rsync -avx BumpAndroidAPIv0.4/ <your project directory>/
```

2.2 Adding the API to your project build path

If you are using Eclipse for development, you will need to add bump-api.jar to you build path. To do this, right click on your project, choose "Build Path", then "Add External Archives..." and finally open the API jar that you previously unzipped into `<your project directory>/libs/bump-api.jar`. You should now see `bump-api.jar` in your project's "Referenced Libraries".

2.3 Updating AndroidManifest.xml

You need to add two sections to your `AndroidManifest.xml` file to enable the API to function properly. The first is adding the API activites to your application section.

```
<activity android:name="com.bumptech.bumpapi.BumpAPI"
         android:configChanges="keyboardHidden|orientation"
         android:theme="@style/BumpDialog" />
<activity android:name="com.bumptech.bumpapi.EditTextActivity"
         android:configChanges="keyboardHidden|orientation"
         android:theme="@style/BumpDialog" />
```

You also need to add the necessary permissions to the manifest. The Bump API requires the Android 1.5 SDK or above, so `minSdkVersion` may be set to a minimum of 3.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-sdk android:minSdkVersion="3" />
```

2.3.1 Example AndroidManifest.xml

Here is the full `AndroidManifest.xml` from our test chat application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="com.bumptech.bumpchat"
     android:versionCode="1"
     android:versionName="1.0">
    <application android:label="@string/app_name" android:icon="@drawable/icon">
      <activity android:name="BumpChat"
               android:label="@string/app_name">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      </activity>
      <activity android:name="com.bumptech.bumpapi.BumpAPI"
               android:configChanges="keyboardHidden|orientation"
               android:theme="@style/BumpDialog" />
      <activity android:name="com.bumptech.bumpapi.EditTextActivity"
               android:configChanges="keyboardHidden|orientation"
               android:theme="@style/BumpDialog" />
    </application>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

2.4 Cleaning your project

Finally, clean and rebuild your project by selecting “Project” then “Clean” from the Eclipse menu. Select “Clean all projects” and click “OK” to remove old project binaries and generate new ones. This helps ensure that your application is using the new Bump API files and that there are no errors.

3 Using the Bump™ API

To use the Bump™ API in your application you must do the three things detailed in the following sections.

3.1 Start the BumpAPI activity

This can be done using the following three lines of code:

```
Intent bump = new Intent(this, BumpAPI.class);
bump.putExtra(BumpAPI.EXTRA_API_KEY, <API Key>);
startActivityForResult(bump, BUMP_API_REQUEST_CODE);
```

where `BUMP_API_REQUEST_CODE` should be a `static final int` defined as a member variable of you class and `<API Key>` should be replaced by your API key. Optionally, you may also set the default user name and action message:

```
bump.putExtra(BumpAPI.EXTRA_USER_NAME, "Bump API User");
bump.putExtra(BumpAPI.EXTRA_ACTION_MSG, "Bump with another phone to share information.");
```

3.2 Implement BumpAPIListener in your activity

The Bump API will generate events on data received and disconnection. You will need to handle these events by implementing the BumpAPIListener interface and passing the implementing class in conn.setListener(...). The BumpAPIListener interface requires the following methods:

`void bumpDisconnect(BumpDisconnectReason reason)`	Called when the API connection terminates
`void bumpDataReceived(byte[] chunk)`	Called when a chunk of data is received from the remote client

3.3 Handle the result of the activity

As with any Android program, the activity is handled by overriding onActivityResult in your class. This can be done as follows:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (requestCode == BUMP_API_REQUEST_CODE) {
    if (resultCode == RESULT_OK) {
      // Bump connected successfully, set this activity as its listener
      conn = (BumpConnection) data.getParcelableExtra(BumpAPI.EXTRA_CONNECTION);
      conn.setListener(this, handler);
    } else if (data != null) {
      // Failed to connect, obtain the reason
      BumpConnectFailedReason reason =
        (BumpConnectFailedReason) data.getSerializableExtra(BumpAPI.EXTRA_REASON);
    }
  }
}
```

If the connection was successful, the API will return a `BumpConnection` object. It is **very** important that when you receive the `BumpConnection` object you call `setListener` on it with something that implements `BumpAPIListener` (see section 4.3). If you do not do this, you will not receive any callbacks when the API application receives data.

On failure, the API will provide a reason for disconnect, in the form of a `BumpConnectFailedReason` (see section 4.4).

3.4 Simulating a physical bump

To simulate a physical bump while running your application in the Android Emulator, tap the Bump logo at the top of the connection dialog.

4 API Reference

Below are the public methods, interfaces and enums that are necessary for interacting with the Bump™ API.

4.1 BumpConnectFailedReason

A `BumpConnectFailedReason` object will be returned by the API when the user exits without connecting.

`FAIL_USER_CANCELED`	The local user exited the API
`FAIL_NETWORK_UNAVAILABLE`	The local user exited before the network became available
`FAIL_INVALID_AUTHORIZATION`	The API key is invalid

4.2 BumpConnection

The following public methods are available on the `BumpConnection` object returned by the API.

`public void setListener(BumpAPIListener l)`	Sets the `BumpAPIListener` for this connection
`public void setListener(BumpAPIListener l, Handler handler)`	Sets the `BumpAPIListener` for this connection defining a `Handler` on which the calls will be made
`public String getOtherUserName()`	Get the user name of the other connected user
`public void send(byte[] chunk)`	Send a chunk to the other user
`public void disconnect()`	Disconnect from the API mailbox service

4.3 BumpAPIListener

The interface for `BumpAPIListener` is as follows.

`void bumpDisconnect(BumpDisconnectReason reason)`	Called when the API connection terminates
`void bumpDataReceived(byte[] chunk)`	Called when a chunk of data is received from the remote client

4.4 BumpDisconnectReason

`END_USER_QUIT`	The local user quit cleanly
`END_LOST_NET`	The connection to the server was lost
`END_OTHER_USER_QUIT`	The remote user quit cleanly
`END_OTHER_USER_LOST`	The remote user was lost