In this document, we present the guidelines for the manual analysis of Apps

1. We only consider the following entry points of a program. That means, we only cover methods that are reached directly or indirectly from one the these start points.
Start points may be methods within classes derived from
    - `android.app.Activity`
    - `android.content.BroadcastReceiver`
    - `android.app.Service`
    - `android.content.ContentProvider`

Additionally, callback methods will be considered. We use assume every methods overriding a method in an Android base class is a callback method.

2. We use the following list of APIs and their mapping the a corresponding configuration name

| Field | Configuration Option | Value Tracking |
|---|---|---|
| `android.os.Build: java.lang.String MANUFACTURER` | `MANUFACTURER` | false |
| `android.os.Build: java.lang.String DEVICE` | `DEVICE` | false |
| `android.content.res.Configuration: int keyboard` | `KEYBOARD` | false |
| `android.content.res.Configuration: java.util.Locale locale` | `LOCALE` | false |
| `android.os.Build$VERSION: int SDK_INT` | `SDK` | false |
| `android.os.Build$VERSION: java.lang.String SDK` | `SDK` | false |
| `android.content.res.Configuration: int screenLayout` | `SCREENLAYOUT` | false |
| `android.content.res.Configuration: int uiMode` | `UIMODE` | false |
| `android.content.res.Configuration: int touchscreen` | `TOUCHSCREEN` | false |
| `android.content.res.Configuration: int navigation` | `NAVIGATION` | false |
| `android.content.res.Configuration: int densityDpi` | `DPI` | false |
| `android.os.Build: java.lang.String CPU_ABI` | `CPU` | false |
| `android.os.Build: java.lang.String CPU_ABI2` | `CPU` | false |

| | | |
|---|---|---|
| `android.os.Build: java.lang.String BOARD` | `BOARD` | false |
| `android.os.Build: java.lang.String BOOTLOADER` | `BOOTLOADER` | false |
| `android.os.Build: java.lang.String BRAND` | `BRAND` | false |
| `android.os.Build: java.lang.String HARDWARE` | `HARDWARE` | false |
| `android.os.Build: java.lang.String MODEL` | `MODEL` | false |
| `android.os.Build: java.lang.String HOST` | `HOST` | false |
| `android.os.Build: java.lang.String ID` | `ID` | false |
| `android.os.Build: java.lang.String PRODUCT` | `PRODUCT` | false |
| `android.os.Build: java.lang.String RADIO` | `RADIO` | false |
| `android.os.Build: java.lang.String SERIAL` | `SERIAL` | false |
| `android.os.Build: java.lang.String TAGS` | `TAGS` | false |
| `android.os.Build: java.lang.String TYPE` | `TYPE` | false |
| `android.os.Build: java.lang.String USER` | `USER` | false |

| Method Call | Configuration Option | Value Tracking |
|---|---|---|
| `Environment.getExternalStorageState()` | `STORAGE` | false |
| `Context.getSystemService(java.lang.String)("location")` | `LOCATION` | false |
| `Context.getSystemService(java.lang.String)("audio")` | `AUDIO` | false |
| `Context.getSystemService(java.lang.String)("wifi")` | `WIFI` | false |
| `Context.getSystemService(java.lang.String)("bluetooth")` | `BLUETOOTH` | false |
| `Context.getSystemService(java.lang.String)("nfc")` | `NFC` | false |
| `Context.getSystemService(java.lang.String)("usb")` | `USB` | false |
| `Context.getSystemService(java.lang.String)("vibrator")` | `VIBRATOR` | false |
| `Context.getSystemService(java.lang.String)("connectivity")` | `NETWORK` | false |
| `Context.getSystemService(java.lang.String)("consumer_ir")` | `INFRARED` | false |

| | | |
|---|---|---|
| Context.getSystemService(java.lang.String)("sensor") | SENSORS | false |
| Context.getSystemService(java.lang.String)("phone") | PHONE | false |
| Context.getSystemService(java.lang.String)("textservices") | TEXT | false |
| android.bluetooth.BluetoothAdapter getDefaultAdapter() | BLUETOOTH | false |
| java.lang.String getRadioVersion() | RADIO | false |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "accessibility_enabled"[, 0]) | ACCESSIBILITY | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "adb_enabled"[, 0]) | ADB | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "background_data"[, 0]) | BACKGROUND_DATA | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "bluetooth_on"[, 0]) | BLUETOOTH_ON | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "data_roaming"[, 0]) | DATA_ROAMING | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "development_settings_enabled"[, 0]) | DEVELOPMENT_SETTINGS_ENABLED | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "device_provisioned"[, 0]) | DEVICE_PROVISIONED | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "usb_mass_storage_enabled"[, 0]) | USB_MASS_STORAGE_ENABLED | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "use_google_mail"[, 0]) | USE_GOOGLE_MAIL | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "wifi_networks_available_notification_on"[, 0]) | WIFI_NETWORKS_AVAILABLE_NOTIFICATION_ON | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "wifi_on"[, 0]) | WIFI_ON | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "airplane_mode_on"[, 0]) | AIRPLANE_MODE_ON | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "debug_app"[, 0]) | DEBUG_APP | true |
| Settings.Secure.getInt(android.content.ContentResolver,java.lang.String[,int])(*, "wait_for_debugger"[, 0]) | WAIT_FOR_DEBUGGER | true |

| | | |
|---|---|---|
| `Settings.Secure.isLocationProviderEnabled(android.content.ContentResolver,java.lang.String)` | LOCATION | false |

## 3. Notes

1. A statement should be annotated if is only reachable depending certain values of a configuration option as defined in 2)
2. A statement's reachability can be restricted by an if statement or if the statement is within a procedure, whose calls are annotated
   Examples:
   ```
   if(A)
      b;
   ```
   b should be annotated, if A encodes a configuration option
   ```
   if(A)
      b();

        ...
      b() {
         c;
      }
   ```
   The statement c within the method b and the call b() should be annotated if A is a configuration option
3. We will not annotate statement in which a value is simply affected by a configuration value:
   ```
   foo(A);
   ```
   The call is not annotated, even if A is a configuration option.
4. A return statement may influence if a statement is reached
   ```
   if(A) return
   foo();
   ```
   Here, foo has to be annotated with !A and return with A.
5. Constraint can be resolved:

```
if(A)
    foo();
bar();
```

Foo has the constraint A, while B, as it is reachable whether A is set or not, must not be annotated.

6. Library calls are handled conservatively, if they take a configuration options as a parameter or the parameter is the base object, we associate the return value with the configuration option.

7. We do consider complex expression if they may influence what part is executed and what not:

   if(A && foo()) …

   In this case, foo() is dependent on A.

8. We consider inter-procedural data-flow

```
foo() {
  return A;
}

bar()
{
  if(foo())
     blub()
}
```

If A is a configuration option, we annotate the call to blub() with A.