

# Android - Runtime permissions

## Permissions prior to Android Marshmallow

Prior to android Marshmallow(6.0) permissions were asked for at install time. When the user wanted to install the app he got a list of all the permissions it needed. Once installed, the application will be able to access all of things granted without any user's acknowledgement what exactly application does with the permission. This means that we declared all the permissions we needed in the android manifest file, when the user installed the app he automatically accepted and granted the app all the permissions and had no control over it. , This can be a security hazard.

In Android 6.0 Marshmallow, application will not be granted any permission at installation time. Instead, application has to ask user for a permission one-by-one at runtime.

## Runtime permissions

When we declared both the compile version and the target Sdk to 23(in the build gradle file) we must now comply with the new android runtime permissions. If we declared the target sdk as 22 and below our app will continue to work as before and even android 6.0 (Marshmallow) users will still grant us all the permissions at installation time. This is because the target sdk means which sdk the app was tested against, and since we haven't tested it on api 23 our app will continue to work as before even when installed on marshmallow device (note: that the user can still revoke the permission we got at install time and this can cause crashes in unprepared apps, but the user will be warned about this when revoking the permission).

In the run time permission , when the app attempts to perform an action that requires the user permission a dialog requesting that permission should appear and the user can allow or deny us that permission. This dialog appear only in the first time we need the permission but the user can still revoke the permission he gave us before in the settings menu, that's we we always have to check whether we have that permission before performing the action.

The permission request dialog shown will *not* launch automatically. Developer has to call for it manually. **In the case that developer try to call some function that requires a permission which user has not granted yet(or granted and revoke it), the function will suddenly throw an Exception which will lead to the application crashing.**

This means that we cannot just call a function to do the job like previous but you have to check for the permission for every single feature or your application will just simply crash !

This new Runtimepermission will work like described only when we set the application's `targetSdkVersion` to 23 which mean it is declared that application has already been tested on API Level 23. And this feature will work only on Android 6.0 Marshmallow. The same app will run with same old behavior on pre-Marshmallow device.

## Automatically granted permissions

The following permissions are not needed be asked at runtime but given at install time even on marshmallow device, and we only need to declare them in the `androidManifest` file:

```
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_NOTIFICATION_POLICY
android.permission.ACCESS_WIFI_STATE
android.permission.ACCESS_WIMAX_STATE
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.BROADCAST_STICKY
android.permission.CHANGE_NETWORK_STATE
android.permission.CHANGE_WIFI_MULTICAST_STATE
android.permission.CHANGE_WIFI_STATE
android.permission.CHANGE_WIMAX_STATE
android.permission.DISABLE_KEYGUARD
android.permission.EXPAND_STATUS_BAR
android.permission.FLASHLIGHT
android.permission.GET_ACCOUNTS
android.permission.GET_PACKAGE_SIZE
android.permission.INTERNET
android.permission.KILL_BACKGROUND_PROCESSES
android.permission.MODIFY_AUDIO_SETTINGS
```

```
android.permission.NFC
android.permission.READ_SYNC_SETTINGS
android.permission.READ_SYNC_STATS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.REORDER_TASKS
android.permission.REQUEST_INSTALL_PACKAGES
android.permission.SET_TIME_ZONE
android.permission.SET_WALLPAPER
android.permission.SET_WALLPAPER_HINTS
android.permission.SUBSCRIBED_FEEDS_READ
android.permission.TRANSMIT_IR
android.permission.USE_FINGERPRINT
android.permission.VIBRATE
android.permission.WAKE_LOCK
android.permission.WRITE_SYNC_SETTINGS
com.android.alarm.permission.SET_ALARM
com.android.launcher.permission.INSTALL_SHORTCUT
com.android.launcher.permission.UNINSTALL_SHORTCUT
```

## Support the new run time permissions

There are few steps we must do in order to support the new android's runtime permissions:

1. Declare both the compile version and the target sdk version to 23 in the build gradle file.
2. Add the requested permission to the android manifest file as before.
3. In your java code, and before attempting to perform the action the requires the permission, we need to check if we have it. in the following example we check if we have the permission to make a phone call:  

```
int hasCallPermission = checkSelfPermission(Manifest.permission.CALL_PHONE);
```
4. If hasCallPermission is not equals to `PackageManager.PERMISSION_GRANTED` we must show the dialog and ask for it before making the call.
5. To show the dialog we simple call to: `requestPermissions(new String[] {Manifest.permission.CALL_PHONE},REQUEST_CODE_ASK_PERMISSIONS);`  
Where the second parameter is an int representing the request code for that permission request.
6. In the activity override the `onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)`. The request code will identify the request after

checking it we can then check whether we got the permission or not. the `String[]` array passed representing the permissions we asked for and the `int[]` array representing the results, let's say we asked only for the permission mentioned above so we should check if `grantResults[0] == PackageManager.PERMISSION_GRANTED` and if so perform the action we want. If not present a dialog to the user telling him that you need that permission.

## Permissions groups

To make things easier android os has divided the permissions into groups. once a single permission in a group is given to us - all the other permissions in that group are automatically granted as well. the permissions groups are:

Permission Group	Permissions
<code>android.permission-group.CALENDAR</code>	<ul style="list-style-type: none"><li><code>android.permission.READ_CALENDAR</code></li><li><code>android.permission.WRITE_CALENDAR</code></li></ul>
<code>android.permission-group.CAMERA</code>	<ul style="list-style-type: none"><li><code>android.permission.CAMERA</code></li></ul>
<code>android.permission-group.CONTACTS</code>	<ul style="list-style-type: none"><li><code>android.permission.READ_CONTACTS</code></li><li><code>android.permission.WRITE_CONTACTS</code></li><li><code>android.permission.GET_ACCOUNTS</code></li></ul>
<code>android.permission-group.LOCATION</code>	<ul style="list-style-type: none"><li><code>android.permission.ACCESS_FINE_LOCATION</code></li><li><code>android.permission.ACCESS_COARSE_LOCATION</code></li></ul>
<code>android.permission-group.MICROPHONE</code>	<ul style="list-style-type: none"><li><code>android.permission.RECORD_AUDIO</code></li></ul>
<code>android.permission-group.PHONE</code>	<ul style="list-style-type: none"><li><code>android.permission.READ_PHONE_STATE</code></li><li><code>android.permission.CALL_PHONE</code></li><li><code>android.permission.READ_CALL_LOG</code></li><li><code>android.permission.WRITE_CALL_LOG</code></li><li><code>com.android.voicemail.permission.ADD_VOICEMAIL</code></li><li><code>android.permission.USE_SIP</code></li><li><code>android.permission.PROCESS_OUTGOING_CALLS</code></li></ul>
<code>android.permission-group.SENSORS</code>	<ul style="list-style-type: none"><li><code>android.permission.BODY_SENSORS</code></li></ul>
<code>android.permission-group.SMS</code>	<ul style="list-style-type: none"><li><code>android.permission.SEND_SMS</code></li><li><code>android.permission.RECEIVE_SMS</code></li><li><code>android.permission.READ_SMS</code></li><li><code>android.permission.RECEIVE_WAP_PUSH</code></li><li><code>android.permission.RECEIVE_MMS</code></li><li><code>android.permission.READ_CELL_BROADCASTS</code></li></ul>
<code>android.permission-group.STORAGE</code>	<ul style="list-style-type: none"><li><code>android.permission.READ_EXTERNAL_STORAGE</code></li><li><code>android.permission.WRITE_EXTERNAL_STORAGE</code></li></ul>

## Support older android versions

The above code will work perfectly on android 6.0 but will throw an exception on android prior to 6.0. To solve that issue we have two solutions:

1. Check the Build version sdk and put all the new code only if its equal or greater the 23:

```
if (Build.VERSION.SDK_INT >= 23) {  
    // Marshmallow+  
} else {  
    // Pre-Marshmallow  
}
```

2. Use the v4 support library functions that are already prepared for that option:

**ContextCompat.checkSelfPermission()** - for checking the permission state  
**ActivityCompat.requestPermissions()** - for showing the dialog

**Reference:** [Everything every Android Developer must know about new Android's Runtime Permission](#), The cheese factory Blog.