Permission handling

The `PermissionsManager` is a set of utilities that help you to check for, request, and respond to any number of Android system permissions such as device location or camera.

`PermissionsManager`

If you build your Android project targeting API level 23 or higher, then your application will need to request permissions during runtime. Handling this directly in your activity produces boilerplate code and can often be hard to manage. That's where the `PermissionsManager` class comes into play. With the `PermissionsManager` class, you can check whether the user has granted location permission and request permissions if the user hasn't granted them yet. You can use `val permissionsManager = PermissionsManager(this)`, you'll need to implement `PermissionsListener`.

Once you have set up your permissions manager, you will still need to override `onRequestPermissionsResult()` in your `Activity` and call the `permissionsManager`'s same method.

Note: *The `PermissionsManager` can also be used for requesting other permissions besides location.*

```
lateinit var permissionsManager: PermissionsManager

override fun onCreate(savedInstanceState: Bundle?) {
  if (PermissionsManager.areLocationPermissionsGranted(context)) {
    // Permission sensitive logic called here, such as activating the Maps SDK's LocationComponent
  } else {
    permissionsManager = PermissionsManager(this)
    permissionsManager.requestLocationPermissions(this)
  }
}

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String>, grantResults
  permissionsManager.onRequestPermissionsResult(requestCode, permissions, grantResults)
}
```

EXAMPLE **Show the user's location on the map →**

See how the PermissionsManager is used to check permissions before showing the device location puck.

`PermissionsListener`

The `PermissionsListener` is an interface that returns information about the state of permissions. Set up the interface and pass it into the `PermissionsManager`'s constructor.

The permission result is invoked once the user decides whether to allow or deny the permission. You can use `granted` boolean parameter to write an `if` statement. Both cases should be handled correctly. Continue with your permission-sensitive logic if the user approves. Otherwise, if the user denies, display a message that tells the user that the permission is required for your application to work. An explanation isn't required but strongly encouraged to allow the user to understand why you are requesting this permission.

```
var permissionsListener: PermissionsListener = object : PermissionsListener {
  override fun onExplanationNeeded(permissionsToExplain: List<String>) {

  }

  override fun onPermissionResult(granted: Boolean) {
    if (granted) {

      // Permission sensitive logic called here, such as activating the Maps SDK's LocationCompone

    } else {

      // User denied the permission

    }
  }
}
```

EXAMPLE **Show the user's location on the map →**

See how the PermissionsListener is used to respond to changes in permissions.

Additional Developer Resources

- Mapbox Developer Discord
- Developer Cheatsheet
- Mapbox Support
- Ask AI

Was this page helpful? Yes No