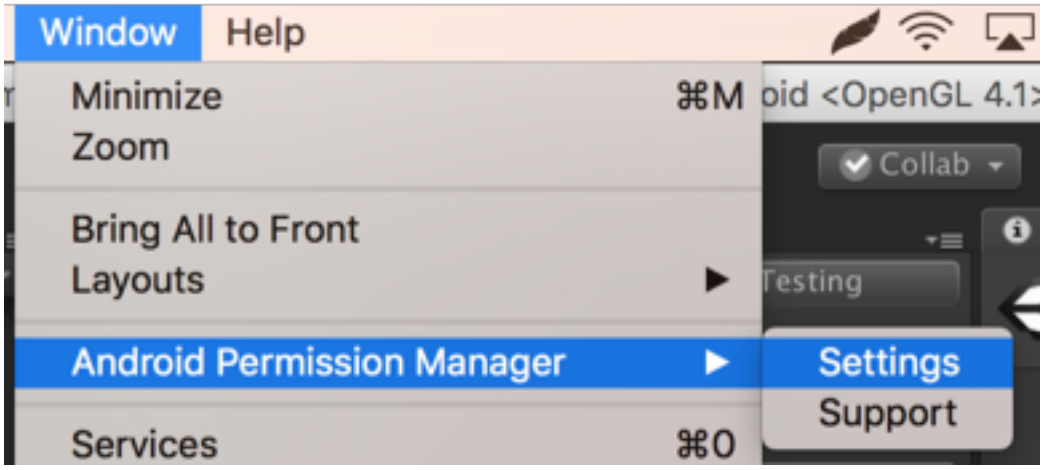
Android Permission Manager - Documentation

Asset Store Page | Support

Quick Start

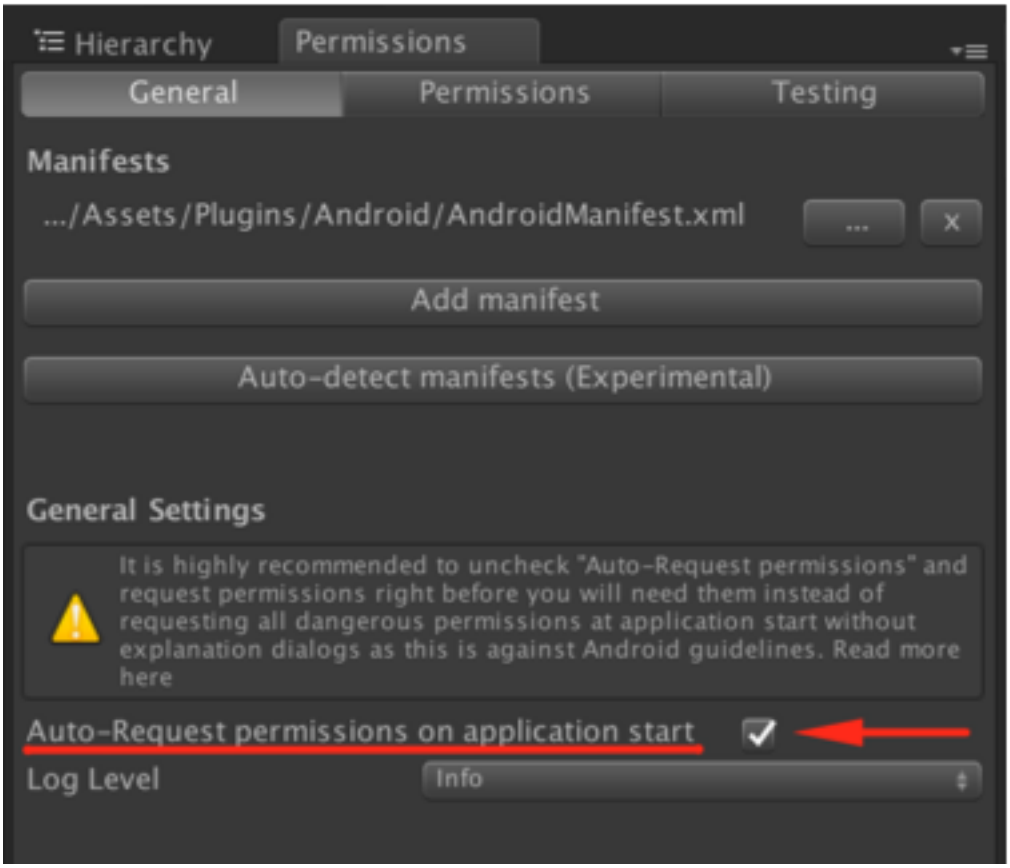
To start using this plugin, you will need to open the window at “Window->Android Permission Manager-> Settings”.



Press “Find All” on the “General” tab to add all manifests the project has (including the ones in .jar/.aar files)

By default unity requests all dangerous permissions at the start which is mostly against Android guidelines.

If you see the warning message – click the checkbox to disable auto-requesting



By default, this plugin will test android SDK 23 which will ask the user to grant the permission for all dangerous permission requested.

But before requesting you need to add a permission to manifest which you can do on the “Permissions” tab with the help of autocomplete. For “example scenes" to work – please add `android.permission.READ_EXTERNAL_STORAGE` and `android.permission.WRITE_EXTERNAL_STORAGE`

You can also remove any permission from all manifests by clicking on the ‘x’ next to it.

To request a permission use:

```
PermissionManager.RequestPermission(permissionName, OnPermissionGranted, OnPermissionDenied);
```

With `permissionName` you want to request and methods to be called when permission was granted or denied (see Example “TestReadPermission” and "API" section)

This will request real permissions on Android device, and also do a testing implementation in the editor if Testing is enabled.

All permission constants can be found in PermissionConstants.cs and used like so - Permission.READ_EXTERNAL_STORAGE

While testing in the editor - you can change the state of permissions without exiting Play Mode, see “Permission States”. For example, change to "Not Requested" to request freshly again.

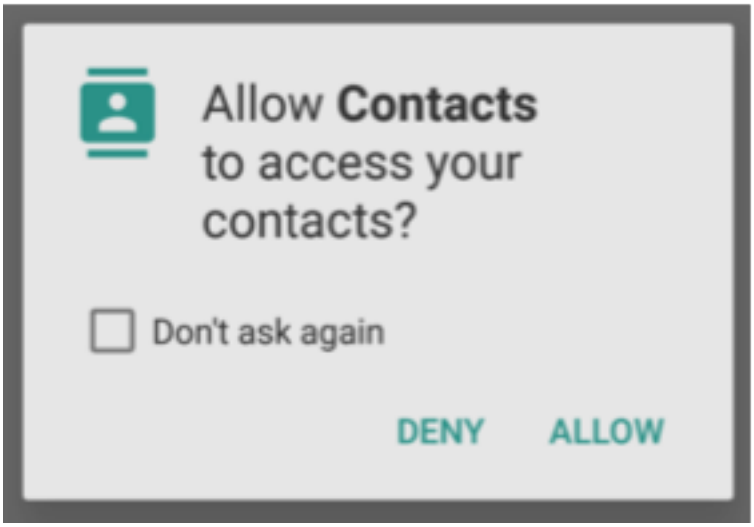
You can disable In-Editor testing on the “Testing” tab.

You can change the Android SDK version used for testing on the “Testing” tab (if the version is 23 or higher, then dangerous permissions will show a dialog asking the user to grant or deny this permission, where user can select the “Don’t show again” options and all permissions will be rejected without asking the user again).

To check if rationale dialogs need to be shown, you can use the PermissionManager.ShouldShowRequestPermissionRationale and pass the permission name. (More in "Rationale Dialog” section)

Dangerous and normal permissions

There are different protection levels for permissions, but most notable are normal and dangerous permissions. Normal permissions are granted upon app installation and Dangerous need to be granted by the user explicitly, the request can be rejected, and even if the user granted it, it can be later revoked from setting. (See more here)

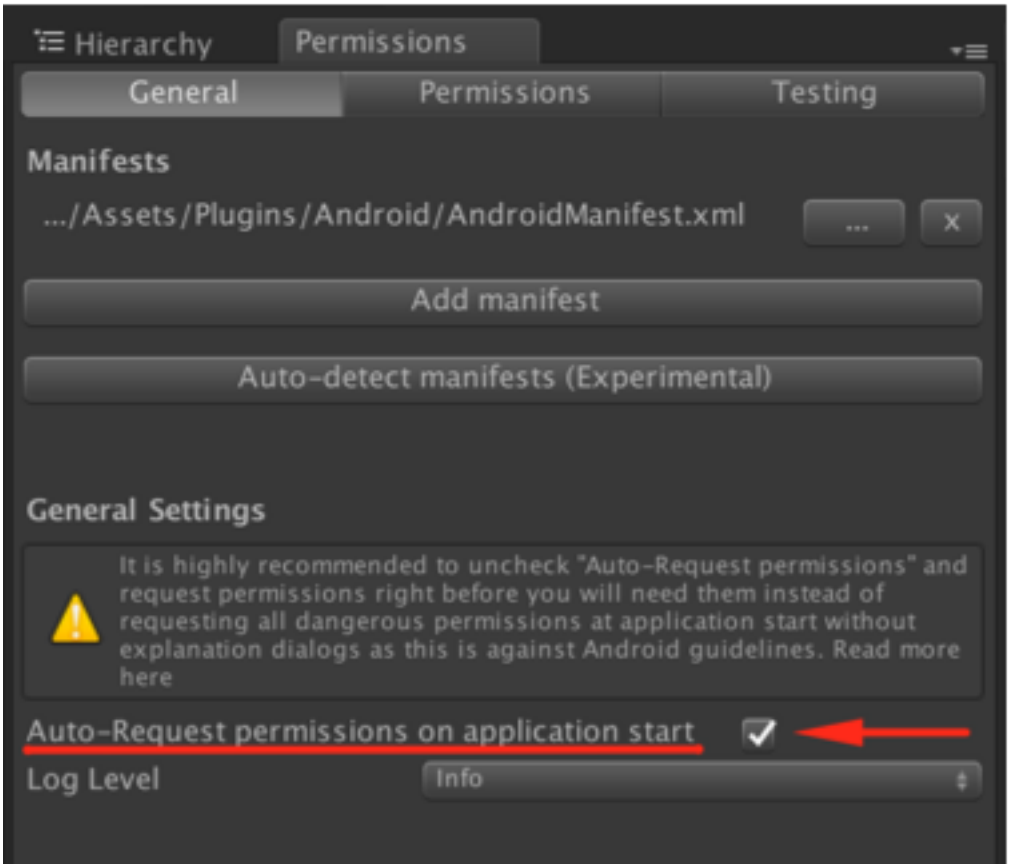


For devices with OS version lower than Android 6 (SDK 22 and lower) all permissions, including dangerous, are granted before the app was launched for the first time.

Unity Auto-Request Permissions

By default - Unity requests all dangerous permission at the start of the app, but this will be one of the first things to be pointed out by Android Review team if you want to get featured. Actually, you need to request permissions right before you need them and also add an explanation why are they required, which Unity doesn't allow you to do.

If you have auto-requesting enabled - this plugin will warn you about it like so

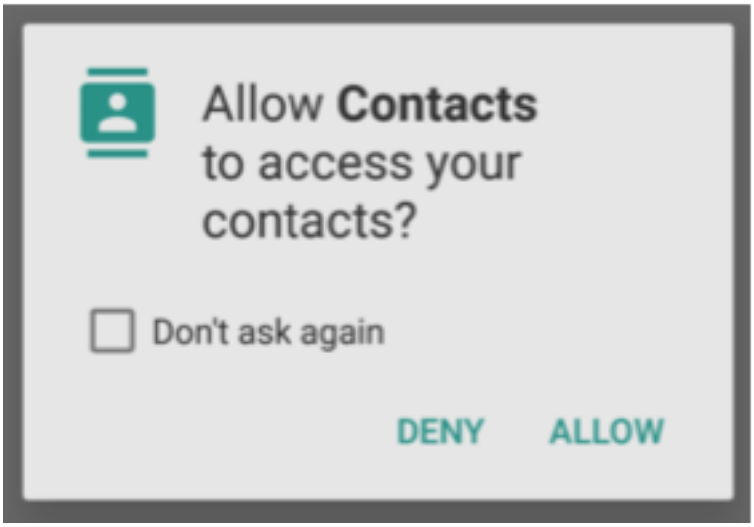


When you click the checkbox to disable it

`<meta-data android:name="unityplayer.SkipPermissionsDialog" android:value="true" \>` will be added to the manifest. And as it says here this will disable automatic permission requesting on app start and you will need to request dangerous permissions yourself (See “API” section)

"Don’t ask again"

After the user denied the permission, if you will request this permission again - a slightly different permission dialog will be shown



If the user clicks the "Don’t show again" checkbox, only Deny button remains active and if the user presses it - all further permission requests for this permission or any other from the same group will instantly be denied. And after that - the only way to get this permission is to ask the user to enable it in the settings.

Permission Groups

Dangerous permissions are grouped into “Permission Groups” like shown below:

Permission Group	Permissions
android.permission-group.CALENDAR	• android.permission.READ_CALENDAR • android.permission.WRITE_CALENDAR
android.permission-group.CAMERA	• android.permission.CAMERA
android.permission-group.CONTACTS	• android.permission.READ_CONTACTS • android.permission.WRITE_CONTACTS • android.permission.GET_ACCOUNTS
android.permission-group.LOCATION	• android.permission.ACCESS_FINE_LOCATION • android.permission.ACCESS_COARSE_LOCATION
android.permission-group.MICROPHONE	• android.permission.RECORD_AUDIO
android.permission-group.PHONE	• android.permission.READ_PHONE_STATE • android.permission.CALL_PHONE • android.permission.READ_CALL_LOG • android.permission.WRITE_CALL_LOG • com.android.voicemail.permission.ADD_VOICEMAIL • android.permission.USE_SIP • android.permission.PROCESS_OUTGOING_CALLS
android.permission-group.SENSORS	• android.permission.BODY_SENSORS
android.permission-group.SMS	• android.permission.SEND_SMS • android.permission.RECEIVE_SMS • android.permission.READ_SMS • android.permission.RECEIVE_WAP_PUSH • android.permission.RECEIVE_MMS • android.permission.READ_CELL_BROADCASTS
android.permission-group.STORAGE	• android.permission.READ_EXTERNAL_STORAGE • android.permission.WRITE_EXTERNAL_STORAGE

Any requests to one permission from the group will also affect other permissions in the same group.

"For example, if you request a READ_EXTERNAL_STORAGE permission from STORAGE category and Deny it, and then request the WRITE_EXTERNAL_STORAGE that is from the same group - the permission dialog will already have a checkbox to not ask again, and if you would check it and Deny, then when you request either READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE they both will be rejected without dialog showing, even though you pressed "Don't ask again" for only one of them.

Likewise, if you requested READ_EXTERNAL_STORAGE initially, and it was "Allowed", then when WRITE_EXTERNAL_STORAGE is requested - it will be automatically guaranteed without any permission dialog.

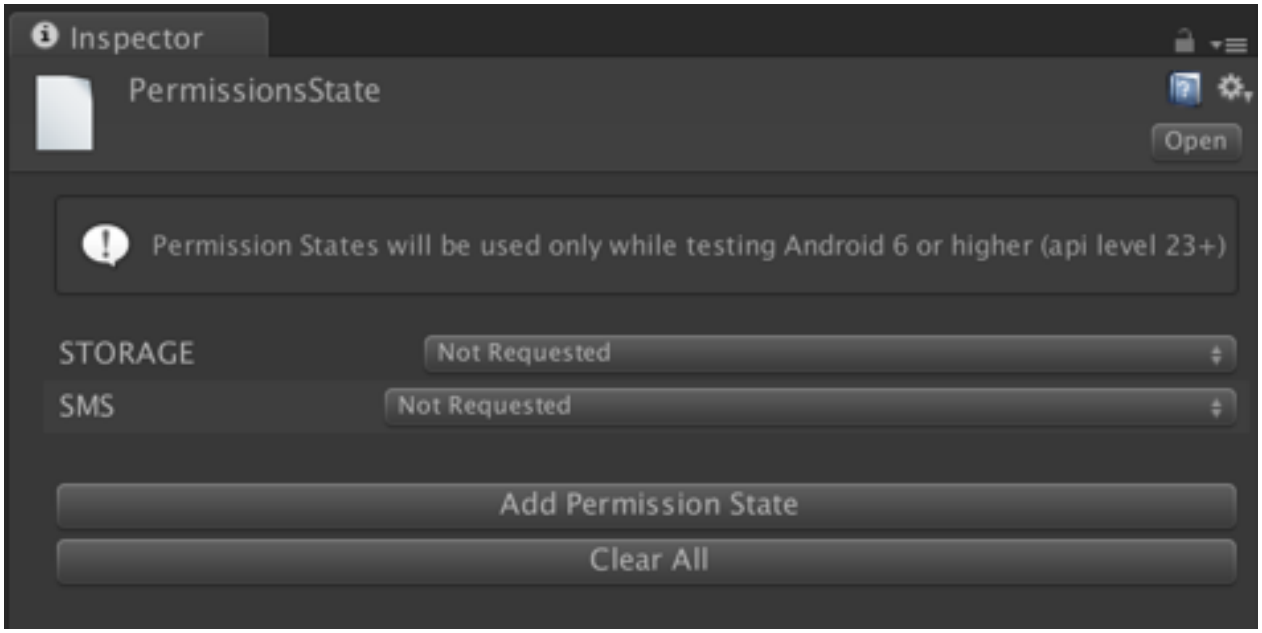
Although you shouldn't base your game logic on the structure of permission groups as it may be changed in future versions of Android SDK. (You can read more about permission groups here)

Permission States

This plugin allows you to test permission changes between launches.

This feature changes the file: Plugins/HG/Settings/Resources/ PermissionsState.asset and if you use a VCS, you'll probably want to add it to a .gitignore file or an alternative. (Or disable "persistent storage" on the "Testing" tab)

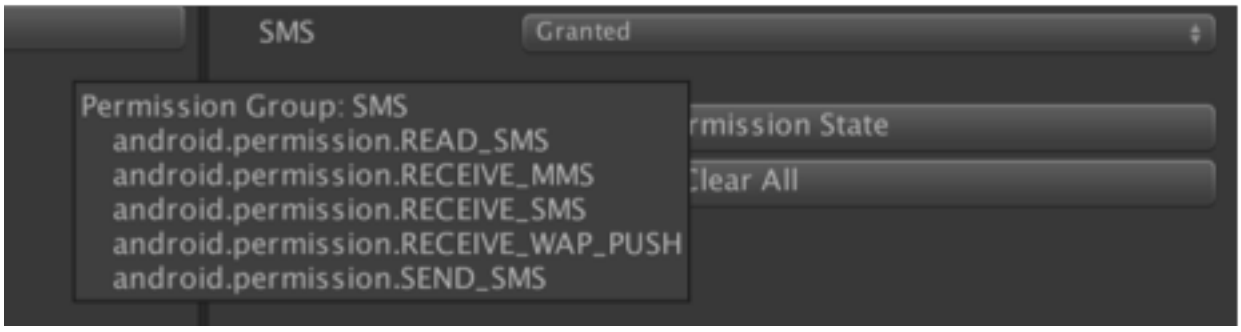
You can edit permission state even in play mode, to do that go to "Testing" tab and press "Edit test permissions state" or just select "PermissionsState.asset" in the project window. You will need to have Inspector visible.



For permissions that have a permission group, this list will display permission group names instead of actual permission names because the result of the permission request will actually affect all permissions in its group.

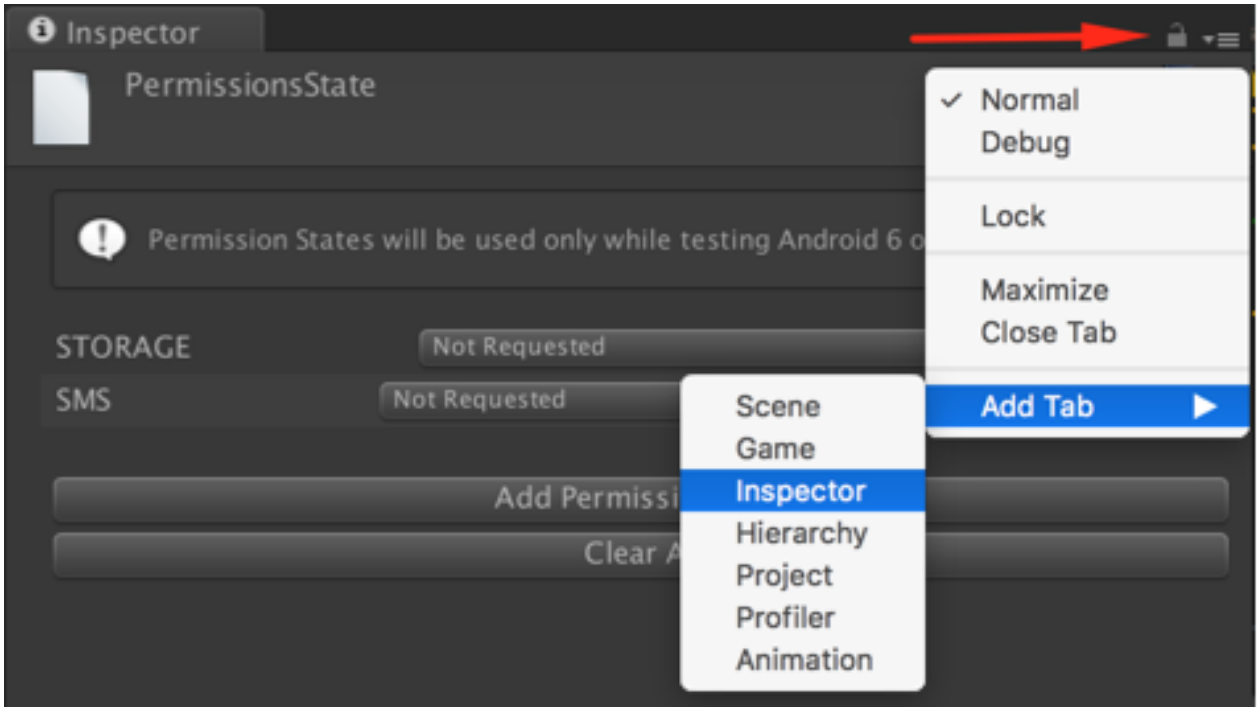
For example, when you add android.permission.WRITE_EXTERNAL_STORAGE it will be automatically converted to the permission group it is contained in (STORAGE)

You can hover over a permission group to get a tooltip with all permissions that go into it.



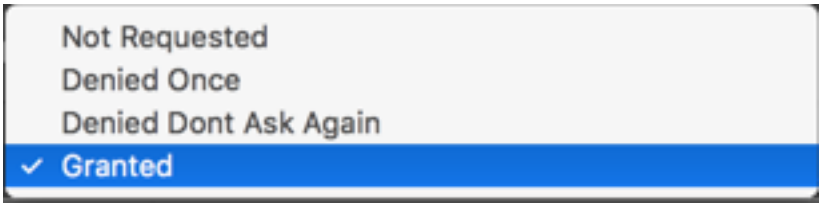
Note: tooltips do not work while in Play Mode which is "By Design" in Unity

Tip: You can add a new inspector window and lock it to only show the last selected file.



Possible States

These states take effect only for API level 23+ so I will go over the behavior only for that case



Not Requested

This is the same as if permission (or permission group) wasn't on this list at all. Permission dialog will appear but "Don’t ask again" checkbox will not be visible

Denied Once

The permission was denied at least once when requesting a permission with this state - permission dialog will appear and "Don’t ask again" checkbox will be visible.

Denied Don’t Ask Again

This is the state of permission that was denied with "Don’t ask again" checked.

Any later requests to grant this permission will result in it being denied without any dialog, although the user may change it in Settings (see “Turn permissions on or of”)

Granted

Any later requests to grant this permission will result in it being granted without any dialog, although the user can revoke this access.

Rationale Dialogs

Google recommends to show the rationale dialog when the user may not know why will it be required, or why the app wouldn’t work without it.

But you can’t show this rationale dialog every time before requesting a permission because:

- If permission was already granted - you don’t need to show it (for example the user can allow the permission from the settings)
- If “Don't ask again” was checked, then the permissions dialog will not be shown, and so the rationale also shouldn’t be.

- If the permission is in the same “Permission Group” as the other permission that was already Granted or Denied with “Don’t ask again” checked.
- If app runs on Android 5 or lower, rationale dialogs also don't need to be shown as all permissions are granted on app install (the ones present in the manifest)
- You may want to show the rationale dialog only after permission was denied once (this is how `shouldShowRequestPermissionRationale` on Android is implemented)

And so you need to check if you need to show the rationale or not - see "ShouldShowRequestPermissionRationale" in API section.

API

There is a lot of code handling different behaviors under the hood, but the API itself is pretty simple.

Note: you can call any of these API methods on any android version

PermissionManager class

IsPermissionGranted

```
bool IsPermissionGranted(string permissionName)
```

Checks whether your app has a given permission

`permissionName` - is the name of the permission to be checked. You can pass in any string here, although all constants from here can be found in the HG.Permission class

For API 23+ it checks if permission was already granted after request and for API 22 and lower - returns true if permission can be found in the manifest.

RequestPermission

```
void RequestPermission(string permissionName, Action<string> OnPermissionGranted, Action<string> OnPermissionDenied)
```

Requests permissions to be granted to this application. These permissions must be added in your manifest. Normal permissions are granted at install time if they were added in the manifest. On API 23+ dangerous permissions will show a native permission request dialog. On API 22 and lower - permission is instantly granted if it can be found in the manifest.

`permissionName` - Which permission to request

`OnPermissionGranted` - Method to be called if permission was granted

`OnPermissionDenied` - Method to be called if permission was denied

You can pass a method that receives the name of the granted/denied permission like so:

```
public void TestRequestPermission()
{
   PermissionManager.RequestPermission(STORAGE_PERMISSION,
       OnPermissionGranted,
       OnPermissionDenied);
}
private void OnPermissionGranted(string grantedPermission)
{
```

```
    Debug.Log("Permission was granted after request");
}
private void OnPermissionDenied(string deniedPermission)
{
    Debug.Log("Permission was denied");
}
```

Or you can also use lambdas:

```
 PermissionManager.RequestPermission(STORAGE_PERMISSION,
    (permission) => { Debug.Log("Permission was granted
after request"); },
    (permission) => { Debug.Log("Permission was
denied"); });
```

Note: You need to pass in methods to PermissionManager.RequestPermission instead of just using a return value, because on android the request is actually asynchronous and can't return the result instantly.

IsPlatformSupported

`bool IsPlatformSupported()`

IsPlatformSupported returns true on Android, and also if Testing is enabled - in the unity editor.

Requesting permissions on any platform will result in OnPermissionGranted or OnPermissionDenied being called, although any platform aside from Android and also Editor while Testing is enabled - will always call OnPermissionDenied and IsPermissionGranted method will always return false.

But instead of wrapping the code in `#if UNITY_ANDROID` you can use IsPlatformSupported which will allow you to run tests in unity editor even if current build platform isn't Android.

ShouldShowRequestPermissionRationale

```
bool ShouldShowRequestPermissionRationale(string permissionName)
```

Returns true if rationale should be shown before requesting `permissionName`. (see "Rationale Dialogs" section)

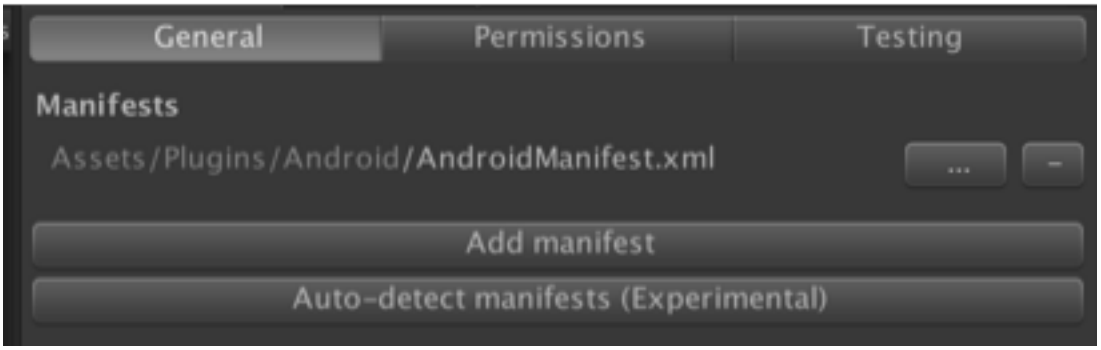
HG.Permission class

This class contains nearly 150 constants to help and simplify permission requesting. HG.Permission class contains all the constants that can be found here

For example, you can pass in `Permission.CALL_PHONE` with the help of your IDE's autocomplete instead of writing `"android.permission.CALL_PHONE"` in multiple places.

Settings window

General tab



“Add manifest" this will allow you to add an XML manifest or a jar/aar library which may contain a manifest. This will allow the plugin to notice it and read the list of permissions, allow to modify the permissions in these files

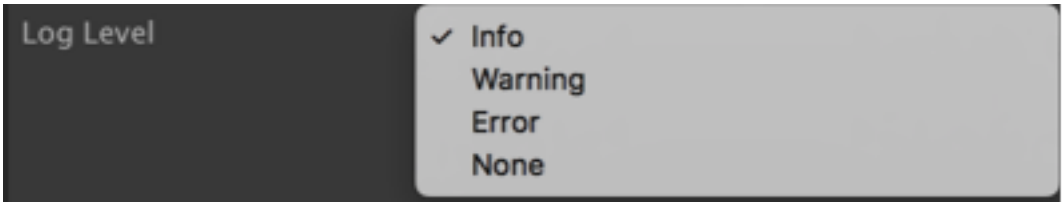
“Find all manifests”, this will find all the manifests and libraries that contain AndroidManifest.xml file.

Here, you will also see a full list of manifests this plugin knows about.

You can open each of them in Explorer by pressing on the “…” next to its location. “-” button removes the manifest from the list of manifests this plugin knows about but doesn’t actually delete the manifest from the disk.

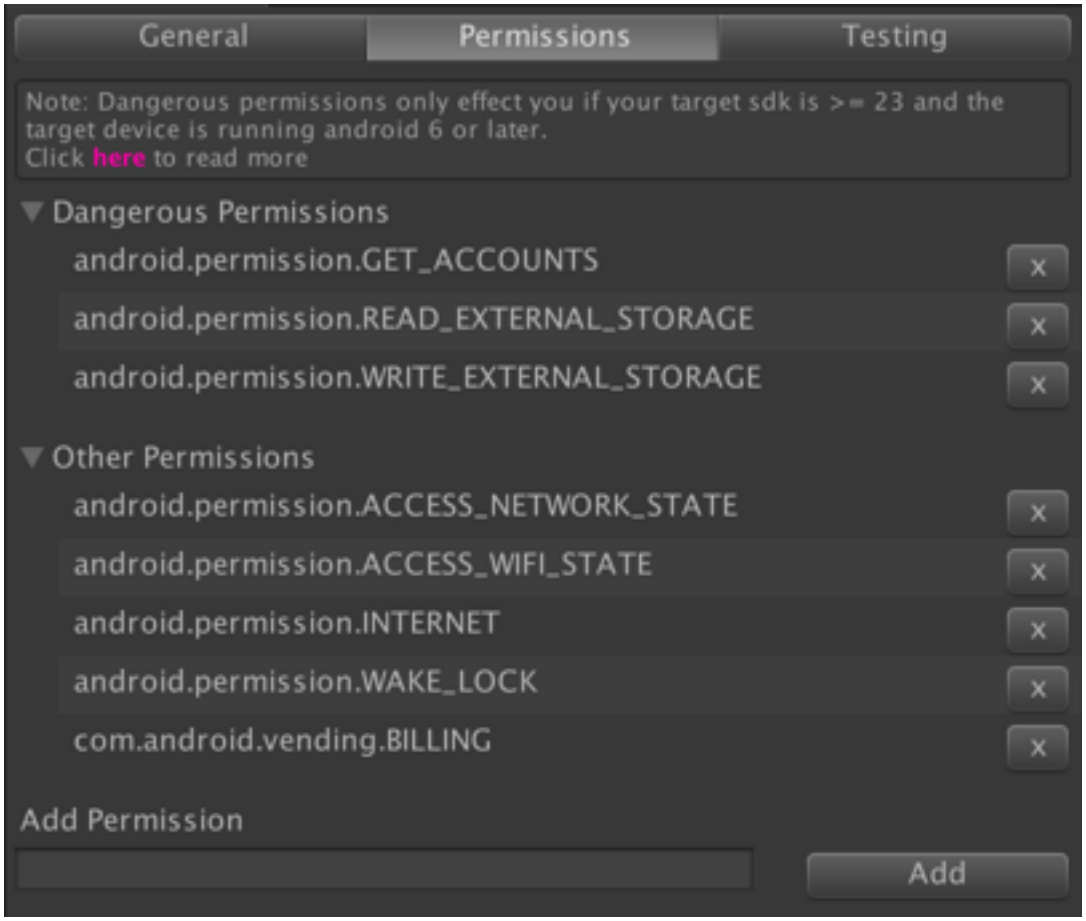
Auto-request permissions checkbox allows to disable the Unity’s default behavior of requesting all dangerous permissions right at the apps launch. (You can read more in the "Unity Auto-Request Permissions" section)

“Log level”



Changing the log level to "Warning" will disable some explanation logs (i.e., why permission was granted without a dialog). Selecting “Error” level - will leave only error messages. For example, if you request permission that wasn’t added to a manifest file. Changing log level to “None” will hide all logs from this plugin, including Errors.

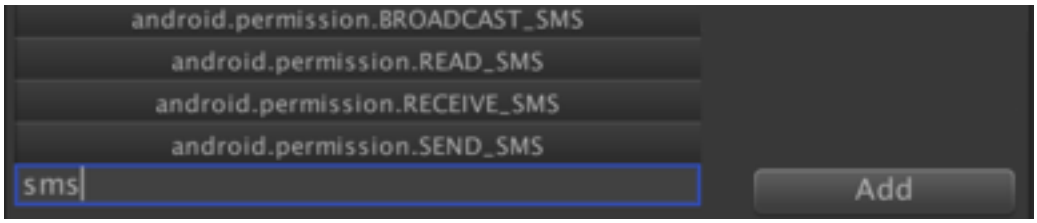
Permissions tab



Found permissions are spit into dangerous and “other” categories. If your target SDK is lower than 23, they will be granted when the user installs the app, you don't need additional requesting. But if you target SDK 23 or higher - you will need to request the ones listed as dangerous permissions using the PermissionManager class (see the API section).

Here you may find some permission that you haven't added yourself directly, for example, from the libraries and you can hover over them and see their description in the tooltip.

By clicking the ‘x’ - permission will be removed from all Manifests so that you don’t need to hunt them down. (Even from .jar/.aar files, although there will be a confirmation dialog if you want to remove permission from a particular archive or not as it may influence library behavior). You may want to backup your manifests in case something goes wrong.



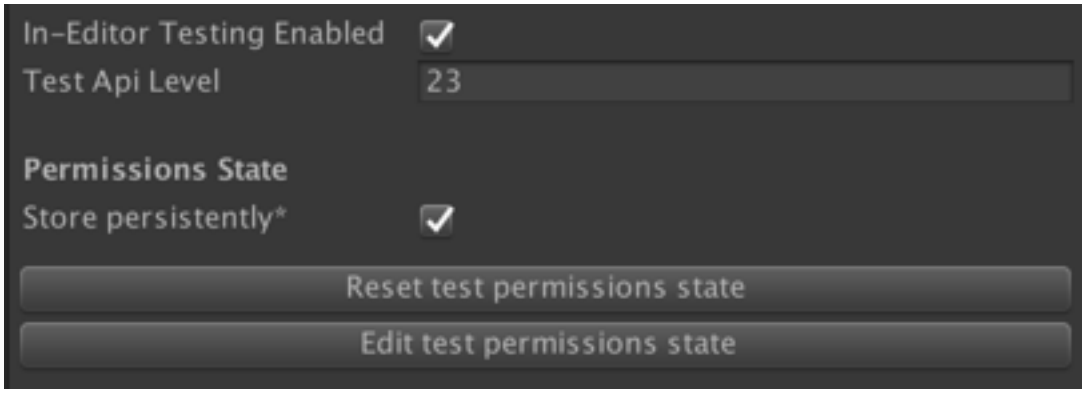
At the bottom of this tab, you can also add new permissions with the help of autocomplete.

This will add this permission into the primary manifest if it was already added, or create a new one and add it there.

Primary manifest location is "Assets/Plugins/Android/AndroidManifest.xml"

Note: if you have a primary manifest but it wasn’t added on the General tab, this manifest will be added instead of overwriting it.

Testing tab

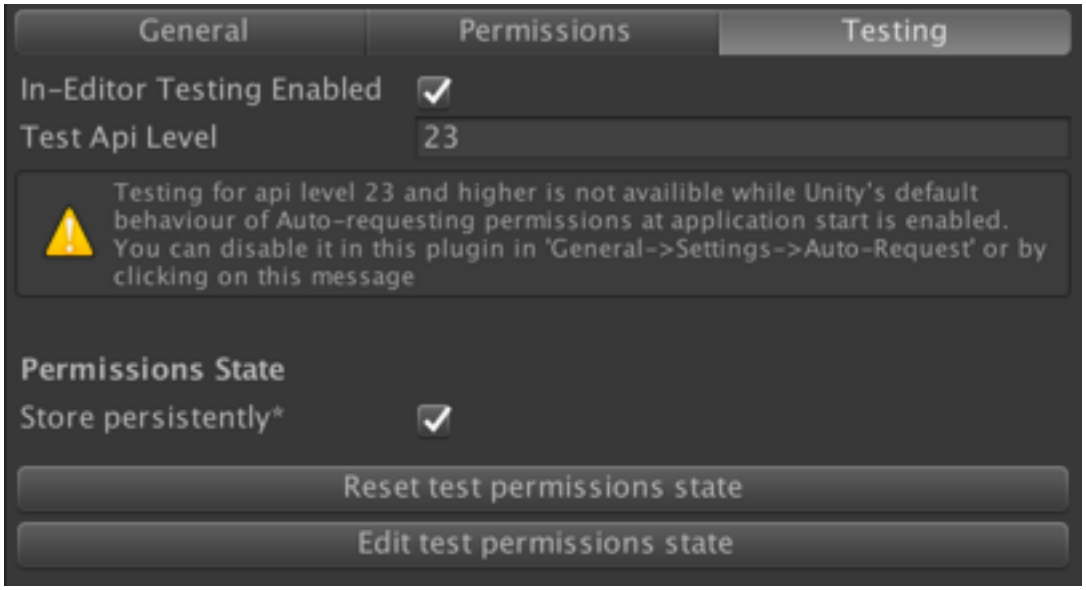


"In-Editor testing enabled"

This is the heart of in editor testing for this plugin. If disabled - all permission requests will be denied, but this influences only in-editor testing. No matter the state of this checkbox - permissions will work on actual android device or in emulator using the same method calls (see API section).

If In-editor testing is enabled - all calls to PermissionManager will emulate the real device behaviour with the ability to modify states to test some particular case and without the need to build to device.

Testing for API level 23+ is not available while “Auto-Request Permissions” is checked.



"Test api level"

Emulation behaviour depends on the selected android api level , the same way the SDK level of the device you are running your game on - influences the behaviour of real permission requesting.

If you set the api level to 23 or higher - then the new android 6's behaviour will be tested with dialogs that allow the user to allow or deny them, including the ability to never ask again (see "Don't ask again" section).

Also see "Examples" section for more info on how behaviour changes when api level is changed.

"Permissions state"

"Store persistently"

Disabling this checkbox will disable saving permission states between game launches inside the editor. So it will be reset each time you press play.

"Reset test permissions state" button

This will reset all the current permissions states, which you can use as a way to reset states without exiting play mode or reset to have a fresh new launch the next time you press play.

"Edit test permissions state" button

This will open a window in inspector that will allow you to fine tune the current state of each permission or permission group even at runtime without exiting playmode.

See "Permission states" section for more info

Examples

Examples can be found in the “/Plugins/HG/Examples” folder.

Examples request permissions and will have errors if you do not have these permissions in the manifest files.

Examples use only two permissions (`android.permission.READ_EXTERNAL_STORAGE` and `android.permission.WRITE_EXTERNAL_STORAGE`) to show the functionallity so

please add them if you want to test them in editor or on the device. You can add them in the “Permissions” tab of the editor window.

You can change the “sdk level” in “Testing” tab of the Settings window, to see how each Example changes when for sdk 23+ or 22 and lower.

For sdk 23+ testing to work - you need to disable permission auto-requesting, see “Unity Auto-Request Permissions” section for more info.

While any of the example scene is opened - you can open “Permission States” window and see how the states change when you request & deny permissions, or change them to specific states to see what happens.

You can reset permissions state between tests by going to “Testing” tab and pressing the “Reset test permissions state“.

To build a specific example to the device or emulator - open File->Build Settings and there add the scene you want to build as the top one.

Request Permission Example

This example shows a way to request and to check if permission was already granted

“Request Permission” button will call the “TestRequestPermission” method when pressed.

“Check Permission” button will call the “CheckPermission” method. This will output if permission is granted to you and if you can use some specifc feature related to it.

api level 23+

If “Allow” was pressed in the dialog - OnPermissionGranted will be called, but if “Deny” was pressed - then OnPermissionDenied will be called instead.

If permission was previously denied with "Don’t ask again" checked - OnPermissionDenied will be called without dialog appearig. See "Don’t ask again" section.

`PermissionManager.IsPermissionGranted(STORAGE_PERMISSION)` will return true only if permission was previously requested and “Allow” was pressed on the dialog

api level 22 and lower

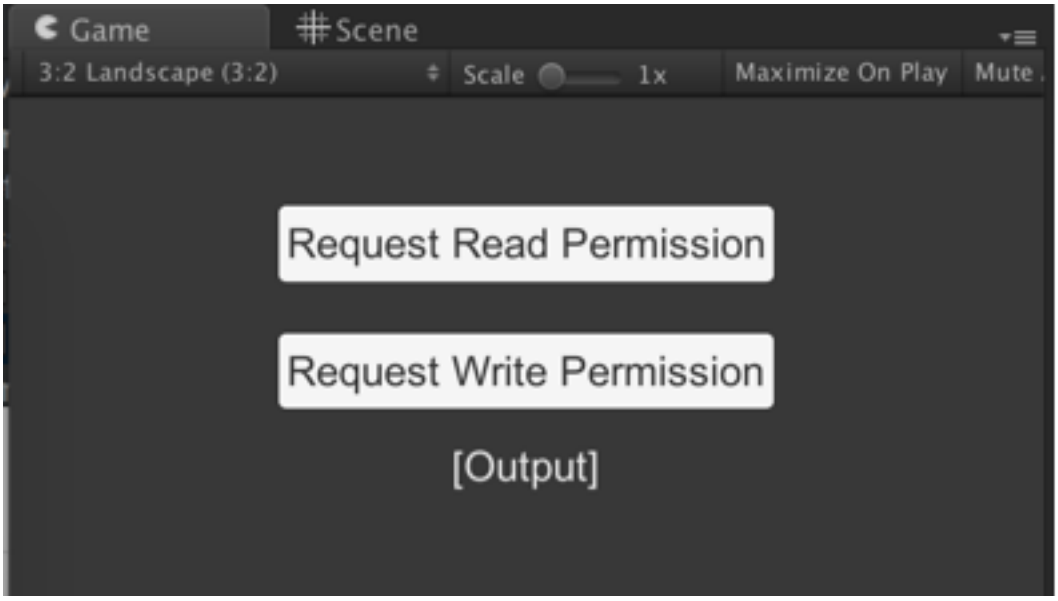
Pressing the “Request Permission” button will grant the permission without any dialog if this permission can be found in the AndroidManifest.xml file.

"Check Permission" will output that permission is granted also if it can be found in the manifest.

Same “Permission Group” Example

This example allows you to test the behaviour when requesting multiple permissions from the same Permission Group.

Both buttons request permissions from the “STORAGE” permission group.



Please see “Permission Groups” section for more info.

api level 23+

If deny with “Don’t ask again“ was pressed for any of these - the other permission will be then denied. And also when one permission is granted - the other is granted automaticly without dialog appearing, although you shoudn’t rely on what permissions are contained in which permission groups as it may be changed.

api level 22 and lower

Both permissions are automaticly granted if they were added to “AndroidManifest.xml”, for example by using the “Permissions” tab

Rationale Dialog Example

“Request Permission” button will call the “TestRequestPermission” method

See “Rationale Dialogs” section for more info.

api level 23+

An example rationale dialog will be shown when requesting this dangerous permission if permission was previously denied atleast once with “Don’t ask again” not checked.

api level 22 and lower

ShouldShowRequestPermissionRationale will always return false

Extra

You may find WRITE_EXTERNAL_PERMISSION & READ_EXTERNAL_PERMISSION added to your final manifest when this plugin doesn't show it - this may be caused by "Development Build" beeing enabled in "Build Settings"

READ_PHONE_STATE being in the final manifest may be caused by missing targetSdk (including jar/aar files that don't have a targetSdk in their manifests)

Note that unity may add some permissions on their own, just if you use some api call, you can read more here under "Permissions".

If this happens you can add them to main manifest for consistency

AndroidManifest.xml in subfolders of Assets/Plugins/Android will only work if they have a properties file with androidlibrary=true https://docs.unity3d.com/Manual/AndroidAARPlugins.html but this is not currently checked by the plugin

You can look at the resulting manifest using https://developer.android.com/studio/build/apk-analyzer.html or AndroidStudio

More info on manifests in Unity

Support

If you still have some questions, please feel free to contact me here: http://letsmakeagame.net/support

Asset Store Page | Support | Request Feature