MediaProjectionManager | API reference | Android Developers

Archived: 2026-04-05 23:33:19 UTC

```
public final class MediaProjectionManager
extends Object
```

Manages the retrieval of certain types of `MediaProjection` tokens.

An example flow of starting a media projection will be:

1. Declare a foreground service with the type `mediaProjection` in the `AndroidManifest.xml`.
2. Create an intent by calling `MediaProjectionManager.createScreenCaptureIntent()` and pass this intent to `Activity.startActivityForResult(Intent,int)`.
3. On getting `Activity.onActivityResult(int,int,Intent)`, start the foreground service with the type `ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION`.
4. Retrieve the media projection token by calling `MediaProjectionManager.getMediaProjection(int,Intent)` with the result code and intent from the `Activity.onActivityResult(int,int,Intent)` above.
5. Register a `MediaProjection.Callback` by calling `MediaProjection.registerCallback(MediaProjection.Callback,Handler)`. This is required to receive notifications about when the `MediaProjection` or captured content changes state. When receiving an `onStop()` callback the `MediaProjection` session has been finished and the client must clean up any resources it is holding, e.g. the `VirtualDisplay` and `Surface`. The MediaProjection may further no longer create any new `VirtualDisplay`s via `MediaProjection.createVirtualDisplay(String,int,int,int,int,Surface,VirtualDisplay.Callback,Handler)`. Note that the `onStop()` callback can be a result of the system stopping MediaProjection due to various reasons. This includes the user stopping the MediaProjection via UI affordances in system-level UI, the screen being locked, or another `MediaProjection` session starting.
6. Start the screen capture session for media projection by calling `MediaProjection.createVirtualDisplay(String,int,int,int,int,Surface,android.hardware.display.VirtualDisplay.Callback,`

Summary

Public methods	
`Intent`	`createScreenCaptureIntent()` Returns an `Intent` that **must** be passed to `Activity.startActivityForResult(Intent,int)` (or similar) in order to start screen capture.
`Intent`	`createScreenCaptureIntent(MediaProjectionConfig config)` Returns an `Intent` that **must** be passed to `Activity.startActivityForResult(Intent,int)` (or similar) in order to start screen capture.
`MediaProjection`	`getMediaProjection(int resultCode, Intent resultData)` Retrieves the `MediaProjection` obtained from a successful screen capture request.
Inherited methods	

From class `java.lang.Object`

`Object`	`clone()` Creates and returns a copy of this object.
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.
`void`	`finalize()` Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`final Class<?>`	`getClass()` Returns the runtime class of this `Object`.
`int`	`hashCode()` Returns a hash code value for the object.
`final void`	`notify()` Wakes up a single thread that is waiting on this object's monitor.
`final void`	`notifyAll()` Wakes up all threads that are waiting on this object's monitor.
`String`	`toString()` Returns a string representation of the object.
`final void`	`wait(long timeoutMillis, int nanos)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait(long timeoutMillis)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait()` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*.

Public methods

createScreenCaptureIntent

```
public Intent createScreenCaptureIntent ()
```

Returns an `Intent` that **must** be passed to `Activity.startActivityForResult(Intent,int)` (or similar) in order to start screen capture. The activity will prompt the user whether to allow screen capture. The result of this activity (received by overriding `onActivityResult(int, int, Intent)`) should be passed to `getMediaProjection(int,Intent)`.

Identical to calling `createScreenCaptureIntent(MediaProjectionConfig)` with a `MediaProjectionConfig.createConfigForUserChoice()`.

Should be used instead of `createScreenCaptureIntent(MediaProjectionConfig)` when the calling app does not want to customize the activity shown to the user.

Returns	
`Intent`	This value cannot be `null`.

createScreenCaptureIntent

```
public Intent createScreenCaptureIntent (MediaProjectionConfig config)
```

Returns an `Intent` that **must** be passed to `Activity.startActivityForResult(Intent,int)` (or similar) in order to start screen capture. Customizes the activity and resulting `MediaProjection` session based up the provided `config`. The activity will prompt the user whether to allow screen capture. The result of this activity (received by overriding `onActivityResult(int, int, Intent)`) should be passed to `getMediaProjection(int,Intent)`.

If `MediaProjectionConfig` was created from:

- `MediaProjectionConfig.createConfigForDefaultDisplay()`, then creates an `Intent` for capturing the default display. The activity limits the user's choice to just the display specified.
- `MediaProjectionConfig.createConfigForUserChoice()`, then creates an `Intent` for deferring which region to capture to the user. This gives the user the same behaviour as calling `createScreenCaptureIntent()`. The activity gives the user the choice between `Display.DEFAULT_DISPLAY`, or a different region.

Should be used instead of `createScreenCaptureIntent()` when the calling app wants to customize the activity shown to the user.

Parameters	
`config`	`MediaProjectionConfig`: Customization for the `MediaProjection` that this `Intent` requests the user's consent for. This value cannot be `null`.
Returns	
`Intent`	An `Intent` requesting the user's consent, specialized based upon the given configuration. This value cannot be `null`.
Throws	
`IllegalArgumentException`	if `MediaProjectionConfig.isOwnAppContentProvided()` is true but no `AppContentProjectionService` is declared.

getMediaProjection

```
public MediaProjection getMediaProjection (int resultCode,
                Intent resultData)
```

Retrieves the `MediaProjection` obtained from a successful screen capture request. The result code and data from the request are provided by overriding `onActivityResult(int, int, Intent)`, which is called after starting an activity using

`createScreenCaptureIntent()` .

Starting from Android `R` , if your application requests the `SYSTEM_ALERT_WINDOW` permission, and the user has not explicitly denied it, the permission will be automatically granted until the projection is stopped. The permission allows your app to display user controls on top of the screen being captured.

An app targeting SDK version `Q` or later must invoke `getMediaProjection` and maintain the capture session (`MediaProjection#createVirtualDisplay`) while running a foreground service. The app must set the `foregroundServiceType` attribute to `FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION` in the `<service>` element of the app's manifest file.

For an app targeting SDK version `U` or later, the user must have granted the app with the permission to start a projection, before the app starts a foreground service with the type `ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION` . Additionally, the app must have started the foreground service with that type before calling this API here, or else it'll receive a `SecurityException` from this API call, unless it's a privileged app. Apps can request the permission via the `createScreenCaptureIntent()` and `Activity.startActivityForResult(Intent,int)` (or similar APIs).

Parameters	
`resultCode`	`int` : The result code from `onActivityResult(int, int, Intent)` .
`resultData`	`Intent` : The result data from `onActivityResult(int, int, Intent)` . This value cannot be `null` .
Returns	
`MediaProjection`	The media projection obtained from a successful screen capture request, or null if the result of the screen capture request is not `RESULT_OK` .
Throws	
`IllegalStateException`	On pre- `Q` devices if a previously obtained `MediaProjection` from the same `resultData` has not yet been stopped.
`SecurityException`	On `Q` + devices if not invoked from a foreground service with type `FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION` , unless caller is a privileged app.

Source: https://developer.android.com/reference/android/media/projection/MediaProjectionManager