Android Alarms
&
Notifications

Android Alarms

- AlarmManager (System Service)
- Alarms are delivered to apps as a **broadcast message** (from the system)
- Inexact alarm (most common use): delivered at *some point* in the future
    - set(), setInexactRepeating(), setAndAllowWhileIdle()
- Exact alarm: delivered at a precise moment in the future
    - Alarm Clock
    - Calendar reminder

Alarm Types

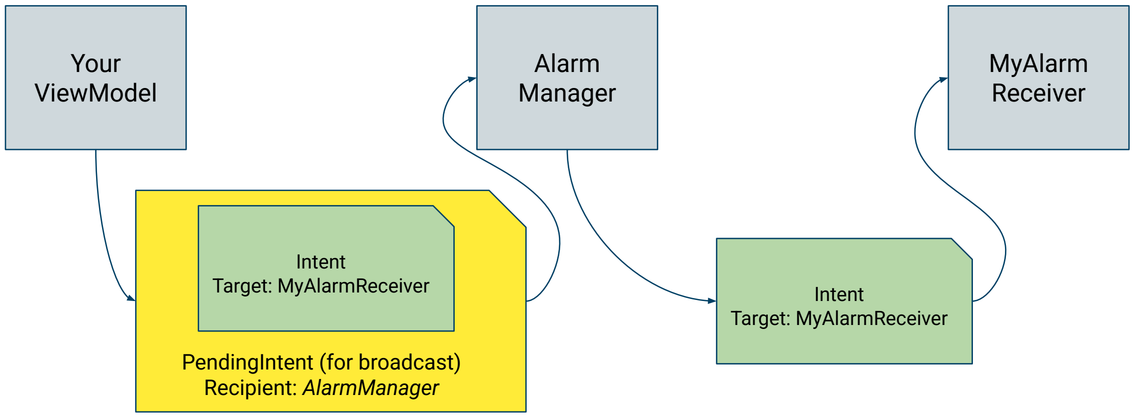
	Inexact Alarm	Exact Alarm
When delivered	Some point in the future	Precise moment in the future
Permission Required	No	Yes `USE_EXACT_ALARM` (Before SDK 33) `SCHEDULE_EXACT_ALARM` (Since SDK 33)

Function calls	`set()`
	`setInexactRepeating()`
	`setAndAllowWhileIdle()`
		`setExact()`
		`setAlarmClock()`
		`setExactAndAllowWhileIdle()`

Step 1: <uses-permission> & <receiver>

```
<manifest...>
    <!-- needed only for EXACT alarm -->
    <uses-permission android:name="android.permission.USE_EXACT_ALARM"/>
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"/>
    <application ...>
        <activity ...>
        ...
        </activity>
        <!-- needed for both INEXACT and EXACT alarms -->
        <receiver android:name=".MyAlarmReceiver">
        </receiver>
    </application>
</manifest>
```

Order of Operations



```
val recvIntent = Intent(context, MyAlarmReceiver::class.java)
val alarmIntent = PendingIntent.getBroadcast(context, 0, recvIntent, PendingIntent.FLAG_IMMUTABLE)
```

Step 2: Alarm Setup (in ViewModel)

```
class AppViewModel(val app: Application): AndroidViewModel(app) {
   lateinit var alarmManager: AlarmManager
   val context = app.applicationContext

   init {
       alarmManager = context.getSystemService<AlarmManager>()!!
   }

   fun wakeUpSecondsFromNow(seconds: Int) {
       val  recvIntent = Intent(context, MyAlarmReceiver::class.java)
          .putExtra("EXTRA_MESSAGE", "Message from App")
       val alarmIntent = PendingIntent.getBroadcast(context, 0, recvIntent, PendingIntent.FLAG_IMMUTABLE)
       // Schedule inexact alarm
       alarmManager?.set(AlarmManager.ELAPSED_REALTIME,
           SystemClock.elapsedRealtime() + seconds * 1000, alarmIntent)
   }
}
```

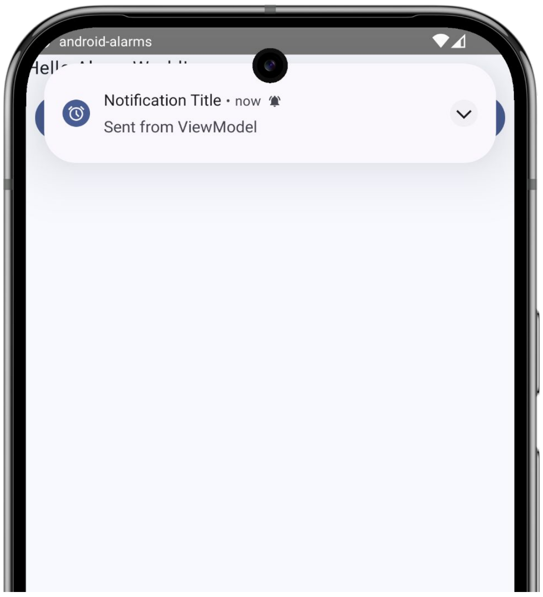
Step 3: Broadcast Receiver

```
class MyAlarmReceiver: BroadcastReceiver() {

   override fun onReceive(ctx: Context?, payload: Intent?) {
       val msg = payload?.getStringExtra("EXTRA_MESSAGE") ?: "None"
       println("Message from ViewModel ${msg}")
   }
}
```

Also declared in AndroidManifest.xml

```
<manifest ...>
    <application ...>
        <receiver android:name=".MyAlarmReceiver" />
    </application>
</manifest>
```

Notifications



Notification Title • now

Sent from ViewModel

Step A: Create Notification Channel

```
class MyApp: Application() {
   override fun onCreate() {
       super.onCreate()
       val notificationManager = getSystemService<NotificationManager>()
       // Use NC-gvsu for notification channel id
       val channel = NotificationChannel("NC-gvsu", "Demo", NotificationManager.IMPORTANCE_HIGH)
       notificationManager?.createNotificationChannel(channel)
   }
}
```

- If your app uses Notifications from several places (*Activity*, *Foreground Service*, *Broadcast Receiver*), it makes sense to create the channel in the Application
- Otherwise, create the channel in your ViewModel

Why Use Channels?

- Notification channels (“categories”) are available in Android 8.0 Oreo (API Level 26)
- Each channel can be associated with priority / importance level
    - In general, higher importance notifications are shown with a sound
- As a developer you can
    - Create multiple channels, each with a specific priority level
- The users of your app can disable/enable these channels to their preferences

Notification Channels

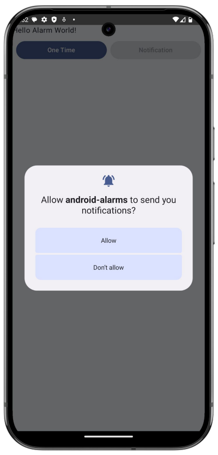
Step B: Runtime Permission & Custom App

```
<manifest...>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>

    <!-- Notification channel(s) are created in MyApp -->
    <application android:name=".MyApp" ...>
        ...
    </application>
</manifest>
```

Step C: Launch Permission Dialog

```
// Add this dependency in build.gradle.kts
implementation("com.google.accompanist:accompanist-permissions:0.37.3")
```

```
// Add permission state variable
val notifPermission = rememberPermissionState(android.Manifest.permission.POST_NOTIFICATIONS)

// Launch permission dialog
if (!notifPermission.status.isGranted) {
   notifPermission.launchPermissionRequest()
}
```



Allow **android-alarms** to send you notifications?

Step D1: Send Notification (Option #1 from ViewModel)

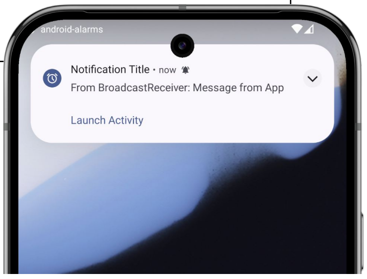
```
class AppViewModel(val app: Application): AndroidViewModel(app) {
    lateinit var notificationManager: NotificationManager
    val context = app.applicationContext
    init {
        notificationManager = context.getSystemService<NotificationManager>()!!
    }

    fun showNotification(msg: String) {
        val notification = NotificationCompat.Builder(context!!, "NC-gvsu")
           .setContentTitle("Notification Title")
           .setContentText(msg)
           .setSmallIcon(R.drawable.outline_alarm_24)
           .build()
        notificationManager.notify(Random.nextInt(), notification)
    }
}
```

Step D2: Send Notification (Option #2 from Broadcast Receiver)

```
class XYZ: BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val notificationManager = context?.getSystemService<NotificationManager>()

        val notification = NotificationCompat.Builder(context!!, "NC-gvsu")
           .setContentTitle("Notification Title")
           .setContentText(msg)
           .setSmallIcon(R.drawable.outline_alarm_24)
           .build()
        notificationManager.notify(Random.nextInt(), notification)
    }
}
```

Variation #1: Launch One Activity

```
val nBuilder= NotificationCompat.Builder(context!!, "NC-gvsu")
   .setContentTitle(_____)
   .setContentText (_____)
   .setSmallIcon   (_____)

val launchIntent = Intent(context!!, TargetActivity:class.java)
val pendingIntent = PendingIntent.getActivity(context!!, 0, launchIntent, FLAG_IMMUTABLE)
nBUilder.addAction("Launch Activity", pendingIntent)
val notification = nBuilder.build()
notificationManager.notify(Random.nextInt(), notification)
```



From BroadcastReceiver: Message from App

Variation #2: Launch Activity + BackStack

```
<application ...>
  <activity android:name=”.MainActivity” .../>
  <activity android:name=”.SecondActivity” android:parentActivityName=”.MainActivity”
</application>
```

```
val nBuilder= NotificationCompat.Builder(context!!, "NC-gvsu")
   .setContentTitle(_____).setContentText (_____).setSmallIcon   (_____)

val launchIntent = Intent(ctx, SecondActivity::class.java)
val notifPendingIntent = TaskStackBuilder.create(context!!).run {
   addNextIntentWithParentStack(launchIntent)
   getPendingIntent(0, FLAG_IMMUTABLE or FLAG_UPDATE_CURRENT)
}
nBUilder.addAction(“Launch Activity”, notifPendingIntent)
val notification = nBuilder.build()
notificationManager.notify(Random.nextInt(), notification)
```