AlarmManager

public class AlarmManager
extends Object (https://developer.android.com/reference/java/lang/Object.html)

java.lang.Object (https://developer.android.com/reference/java/lang/Object.html)
↳ android.app.AlarmManager

added in API level 1
(https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)
Summary: Nested Classes (#nestedclasses) | Constants (#constants) | Methods (#pubmethods) | Inherited Methods (#inhmethods) | [Expand All] (#)

This class provides access to the system alarm services. These allow you to schedule your application to be run at some point in the future. When an alarm goes off, the `Intent` (https://developer.android.com/reference/android/content/Intent.html) that had been registered for it is broadcast by the system, automatically starting the target application if it is not already running. Registered alarms are retained while the device is asleep (and can optionally wake the device up if they go off during that time), but will be cleared if it is turned off and rebooted.

The Alarm Manager holds a CPU wake lock as long as the alarm receiver's onReceive() method is executing. This guarantees that the phone will not sleep until you have finished handling the broadcast. Once onReceive() returns, the Alarm Manager releases this wake lock. This means that the phone will in some cases sleep as soon as your onReceive() method completes. If your alarm receiver called `Context.startService()` (https://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)), it is possible that the phone will sleep before the requested service is launched. To prevent this, your BroadcastReceiver and Service will need to implement a separate wake lock policy to ensure that the phone continues running until the service becomes available.

Note: The Alarm Manager is intended for cases where you want to have your application code run at a specific time, even if your application is not currently running. For normal timing operations (ticks, timeouts, etc) it is easier and much more efficient to use `Handler` (https://developer.android.com/reference/android/os/Handler.html).

> **Note:** Beginning with API 19 (`KITKAT` (https://developer.android.com/reference/android/os/Build.VERSION_CODES.html#KITKAT)) alarm delivery is inexact: the OS will shift alarms in order to minimize wakeups and battery use. There are new APIs to support applications which need strict delivery guarantees; see `setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)) and `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)). Applications whose `targetSdkVersion` is earlier than API 19 will continue to see the previous behavior in which all alarms are delivered exactly when requested.

Instances of this class must be obtained using `Context.getSystemService(Class)` (https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.Class<T>)) with the argument `AlarmManager.class` or `Context.getSystemService(String)` (https://developer.android.com/reference/android/content/Context.html#getSystemService(java.lang.String)) with the argument `Context.ALARM_SERVICE` (https://developer.android.com/reference/android/content/Context.html#ALARM_SERVICE).

Summary

Nested classes	
class	`AlarmManager.AlarmClockInfo` (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html) An immutable description of a scheduled "alarm clock" event.
interface	`AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) Direct-notification alarms: the requester must be running continuously from the time the alarm is set to the time it is delivered, or delivery will fail.

Constants	
`String` (https://developer.android.com/reference/java/lang/String.html)	`ACTION_NEXT_ALARM_CLOCK_CHANGED` (https://developer.android.com/reference/android/app/AlarmManager.html#ACTION_NEXT_ALARM_CLOCK_CHANGED) Broadcast Action: Sent after the value returned by `getNextAlarmClock()` (https://developer.android.com/reference/android/app/AlarmManager.html#getNextAlarmClock()) has changed.
`int`	`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

	Alarm time in `SystemClock.elapsedRealtime()` (https://developer.android.com/reference/android/os/SystemClock.html#elapsedRealtime()) (time since boot, including sleep).
int	`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) Alarm time in `SystemClock.elapsedRealtime()` (https://developer.android.com/reference/android/os/SystemClock.html#elapsedRealtime()) (time since boot, including sleep), which will wake up the device when it goes off.
long	`INTERVAL_DAY` (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_DAY) Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.
long	`INTERVAL_FIFTEEN_MINUTES` (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_FIFTEEN_MINUTES) Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.
long	`INTERVAL_HALF_DAY` (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HALF_DAY) Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.
long	`INTERVAL_HALF_HOUR` (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HALF_HOUR) Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.
long	`INTERVAL_HOUR` (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HOUR) Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.
int	`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC) Alarm time in `System.currentTimeMillis()` (https://developer.android.com/reference/java/lang/System.html#currentTimeMillis()) (wall clock time in UTC).
int	`RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP) Alarm time in `System.currentTimeMillis()` (https://developer.android.com/reference/java/lang/System.html#currentTimeMillis()) (wall clock time in UTC), which will wake up the device when it goes off.

Public methods	
void	`cancel` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel((https://developer.android.com/reference/android/app/PendingIntent.html) `operation)` Remove any alarms with a matching `Intent` (https://developer.android.com/reference/
void	`cancel` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel((`AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/ Remove any alarm scheduled to be delivered to the given `AlarmManager.OnAlarmLis` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html
`AlarmManager.AlarmClockInfo` (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html)	`getNextAlarmClock` (https://developer.android.com/reference/android/app/AlarmManage Gets information about the next alarm clock currently scheduled.
void	`set` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, l

	triggerAtMillis, PendingIntent (https://developer.android.com/reference/androi Schedule an alarm.
void	set (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, l android.os.Handler))(int type, long triggerAtMillis, String (https://develop AlarmManager.OnAlarmListener (https://developer.android.com/reference/android/a (https://developer.android.com/reference/android/os/Handler.html) targetHandler) Direct callback version of set(int, long, PendingIntent) (https://developer.andr android.app.PendingIntent)).
void	setAlarmClock (https://developer.android.com/reference/android/app/AlarmManager.htm android.app.PendingIntent))(AlarmManager.AlarmClockInfo (https://developer.andr info, PendingIntent (https://developer.android.com/reference/android/app/PendingI Schedule an alarm that represents an alarm clock.
void	setAndAllowWhileIdle (https://developer.android.com/reference/android/app/AlarmMa (int type, long triggerAtMillis, PendingIntent (https://developer.android. Like set(int, long, PendingIntent) (https://developer.android.com/reference/and but this alarm will be allowed to execute even when the system is in low-power idle mo
void	setExact (https://developer.android.com/reference/android/app/AlarmManager.html#setE triggerAtMillis, PendingIntent (https://developer.android.com/reference/androi Schedule an alarm to be delivered precisely at the stated time.
void	setExact (https://developer.android.com/reference/android/app/AlarmManager.html#setE android.app.AlarmManager.OnAlarmListener, android.os.Handler))(int type, long tri (https://developer.android.com/reference/java/lang/String.html) tag, AlarmManager (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html (https://developer.android.com/reference/android/os/Handler.html) targetHandler) Direct callback version of setExact(int, long, PendingIntent) (https://develope long, android.app.PendingIntent)).
void	setExactAndAllowWhileIdle (https://developer.android.com/reference/android/app/A android.app.PendingIntent))(int type, long triggerAtMillis, PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation) Like setExact(int, long, PendingIntent) (https://developer.android.com/referen android.app.PendingIntent)), but this alarm will be allowed to execute even when the sys
void	setInexactRepeating (https://developer.android.com/reference/android/app/AlarmMan android.app.PendingIntent))(int type, long triggerAtMillis, long intervalM (https://developer.android.com/reference/android/app/PendingIntent.html) operation) Schedule a repeating alarm that has inexact trigger time requirements; for example, an hour.
void	setRepeating (https://developer.android.com/reference/android/app/AlarmManager.html type, long triggerAtMillis, long intervalMillis, PendingIntent (https:/ operation) Schedule a repeating alarm.
void	setTime (https://developer.android.com/reference/android/app/AlarmManager.html#setTi Set the system wall clock time.
void	setTimeZone (https://developer.android.com/reference/android/app/AlarmManager.html# (https://developer.android.com/reference/java/lang/String.html) timeZone) Sets the system's persistent default time zone.
void	setWindow (https://developer.android.com/reference/android/app/AlarmManager.html#set windowStartMillis, long windowLengthMillis, PendingIntent (https://devel operation) Schedule an alarm to be delivered within a given window of time.
void	setWindow (https://developer.android.com/reference/android/app/AlarmManager.html#set android.app.AlarmManager.OnAlarmListener, android.os.Handler))(int type, long win (https://developer.android.com/reference/java/lang/String.html) tag, AlarmManager (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html (https://developer.android.com/reference/android/os/Handler.html) targetHandler) Direct callback version of setWindow(int, long, long, PendingIntent) (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int,

Inherited methods

(#)From class `java.lang.Object` (https://developer.android.com/reference/java/lang/Object.html)

Constants

ACTION_NEXT_ALARM_CLOCK_CHANGED

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`String` (https://developer.android.com/reference/java/lang/String.html) `ACTION_NEXT_ALARM_CLOCK_CHANGED`

Broadcast Action: Sent after the value returned by `getNextAlarmClock()` (https://developer.android.com/reference/android/app/AlarmManager.html#getNextAlarmClock()) has changed.

> This is a protected intent that can only be sent by the system. It is only sent to registered receivers.

Constant Value: "android.app.action.NEXT_ALARM_CLOCK_CHANGED"

ELAPSED_REALTIME

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int ELAPSED_REALTIME`

Alarm time in `SystemClock.elapsedRealtime()` (https://developer.android.com/reference/android/os/SystemClock.html#elapsedRealtime()) (time since boot, including sleep). This alarm does not wake the device up; if it goes off while the device is asleep, it will not be delivered until the next time the device wakes up.

Constant Value: 3 (0x00000003)

ELAPSED_REALTIME_WAKEUP

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`int ELAPSED_REALTIME_WAKEUP`

Alarm time in `SystemClock.elapsedRealtime()` (https://developer.android.com/reference/android/os/SystemClock.html#elapsedRealtime()) (time since boot, including sleep), which will wake up the device when it goes off.

Constant Value: 2 (0x00000002)

INTERVAL_DAY

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`long INTERVAL_DAY`

Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.

Constant Value: 86400000 (0x0000000005265c00)

INTERVAL_FIFTEEN_MINUTES

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`long INTERVAL_FIFTEEN_MINUTES`

Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.

Constant Value: 900000 (0x00000000000dbba0)

INTERVAL_HALF_DAY

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
long INTERVAL_HALF_DAY
```

Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.

Constant Value: 43200000 (0x0000000002932e00)

INTERVAL_HALF_HOUR

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
long INTERVAL_HALF_HOUR
```

Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.

Constant Value: 1800000 (0x00000000001b7740)

INTERVAL_HOUR

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
long INTERVAL_HOUR
```

Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setInexactRepeating(int, long, long, android.app.PendingIntent)) when running on Android prior to API 19.

Constant Value: 3600000 (0x000000000036ee80)

RTC

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
int RTC
```

Alarm time in `System.currentTimeMillis()` (https://developer.android.com/reference/java/lang/System.html#currentTimeMillis()) (wall clock time in UTC). This alarm does not wake the device up; if it goes off while the device is asleep, it will not be delivered until the next time the device wakes up.

Constant Value: 1 (0x00000001)

RTC_WAKEUP

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
int RTC_WAKEUP
```

Alarm time in `System.currentTimeMillis()` (https://developer.android.com/reference/java/lang/System.html#currentTimeMillis()) (wall clock time in UTC), which will wake up the device when it goes off.

Constant Value: 0 (0x00000000)

Public methods

cancel

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void cancel (PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Remove any alarms with a matching `Intent` (https://developer.android.com/reference/android/content/Intent.html). Any alarm, of any type, whose Intent matches this one (as defined by `filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))), will be canceled.

Parameters	
`operation`	`PendingIntent`: IntentSender which matches a previously added IntentSender. This parameter must not be `null`.

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

cancel

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void cancel (AlarmManager.OnAlarmListener (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) listener
```

Remove any alarm scheduled to be delivered to the given `AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html).

Parameters	
`listener`	`AlarmManager.OnAlarmListener`: OnAlarmListener instance that is the target of a currently-set alarm.

getNextAlarmClock

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
AlarmManager.AlarmClockInfo (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html) getNextAlarmClock ()
```

Gets information about the next alarm clock currently scheduled. The alarm clocks considered are those scheduled by any application using the `setAlarmClock(AlarmManager.AlarmClockInfo, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setAlarmClock(android.app.AlarmManager.AlarmClockInfo, android.app.PendingIntent)) method.

Returns	
`AlarmManager.AlarmClockInfo` (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html)	An `AlarmManager.AlarmClockInfo` (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html) object describing the next upcoming alarm clock event that will occur. If there are no alarm clock events currently scheduled, this method will return `null`.

See also:

`setAlarmClock(AlarmManager.AlarmClockInfo, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setAlarmClock(android.app.AlarmManager.AlarmClockInfo, android.app.PendingIntent))

`AlarmManager.AlarmClockInfo` (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html)

`ACTION_NEXT_ALARM_CLOCK_CHANGED` (https://developer.android.com/reference/android/app/AlarmManager.html#ACTION_NEXT_ALARM_CLOCK_CHANGED)

set

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void set (int type,
              long triggerAtMillis,
              PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule an alarm. **Note: for timing operations (ticks, timeouts, etc) it is easier and much more efficient to use** `Handler` (https://developer.android.com/reference/android/os/Handler.html)**.** If there is already an alarm scheduled for the same IntentSender, that previous alarm will first be canceled.

If the stated trigger time is in the past, the alarm will be triggered immediately. If there is already an alarm for this Intent scheduled (with the equality of two intents being defined by `filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent)), then it will be removed and replaced by this one.

The alarm is an Intent broadcast that goes to a broadcast receiver that you registered with `registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)) or through the <receiver> tag in an AndroidManifest.xml file.

Alarm intents are delivered with a data extra of type int called `Intent.EXTRA_ALARM_COUNT` (https://developer.android.com/reference/android/content/Intent.html#EXTRA_ALARM_COUNT) that indicates how many past alarm events have been accumulated into this intent broadcast. Recurring alarms that have gone undelivered because the phone was asleep may have a count greater than one when delivered.

> **Note:** Beginning in API 19, the trigger time passed to this method is treated as inexact: the alarm will not be delivered before this time, but may be deferred and delivered some time later. The OS will use this policy in order to "batch" alarms together across the entire system, minimizing the number of times the device needs to "wake up" and minimizing battery use. In general, alarms scheduled in the near future will not be deferred as long as alarms scheduled far in the future.
>
> With the new batching policy, delivery ordering guarantees are not as strong as they were previously. If the application sets multiple alarms, it is possible that these alarms' *actual* delivery ordering may not match the order of their *requested* delivery times. If your application has strong ordering requirements there are other APIs that you can use to get the necessary behavior; see `setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)) and `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)).
>
> Applications whose `targetSdkVersion` is before API 19 will continue to get the previous alarm behavior: all of their scheduled alarms will be treated as exact.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`Handler` (https://developer.android.com/reference/android/os/Handler.html)

`setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent))

`setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent))

`setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

set

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void set (int type,
                long triggerAtMillis,
                String (https://developer.android.com/reference/java/lang/String.html) tag,
                AlarmManager.OnAlarmListener (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) liste
                Handler (https://developer.android.com/reference/android/os/Handler.html) targetHandler)
```

Direct callback version of `set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)). Rather than supplying a PendingIntent to be sent when the alarm time is reached, this variant supplies an `AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) instance that will be invoked at that time.

The OnAlarmListener's `onAlarm()` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html#onAlarm()) method will be invoked via the specified target Handler, or on the application's main looper if `null` is passed as the `targetHandler` parameter.

Parameters	
`type`	`int`: type of alarm. Value is RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), RTC (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), ELAPSED_REALTIME_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or ELAPSED_REALTIME (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
`tag`	`String`: string describing the alarm, used for logging and battery-use attribution
`listener`	`AlarmManager.OnAlarmListener`: `AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) instance whose `onAlarm()` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html#onAlarm()) method will be called when the alarm time is reached. A given OnAlarmListener instance can only be the target of a single pending alarm, just as a given PendingIntent can only be used with one alarm at a time.
`targetHandler`	`Handler`: `Handler` (https://developer.android.com/reference/android/os/Handler.html) on which to execute the listener's onAlarm() callback, or `null` to run that callback on the main looper.

setAlarmClock

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setAlarmClock (AlarmManager.AlarmClockInfo (https://developer.android.com/reference/android/app/AlarmManager.AlarmClockInfo.html) inf
                PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule an alarm that represents an alarm clock. The system may choose to display information about this alarm to the user.

This method is like `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)), but implies RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP).

Parameters	
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent))

`setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))

`setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`getNextAlarmClock()` (https://developer.android.com/reference/android/app/AlarmManager.html#getNextAlarmClock())

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

setAndAllowWhileIdle

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setAndAllowWhileIdle (int type,
                long triggerAtMillis,
                PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Like `set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)), but this alarm will be allowed to execute even when the system is in low-power idle modes. This type of alarm must **only** be used for situations where it is actually required that the alarm go off while in idle -- a reasonable example would be for a calendar notification that should make a sound so the user is aware of it. When the alarm is dispatched, the app will also be added to the system's temporary whitelist for approximately 10 seconds to allow that application to acquire further wake locks in which to complete its work.

These alarms can significantly impact the power use of the device when idle (and thus cause significant battery blame to the app scheduling them), so they should be used with care. To reduce abuse, there are restrictions on how frequently these alarms will go off for a particular application. Under normal system operation, it will not dispatch these alarms more than about every minute (at which point every such pending alarm is dispatched); when in low-power idle modes this duration may be significantly longer, such as 15 minutes.

Unlike other alarms, the system is free to reschedule this type of alarm to happen out of order with any other alarms, even those from the same app. This will clearly happen when the device is idle (since this alarm can go off while idle, when any other alarms from the app will be held until later), but may also happen even when not idle.

Regardless of the app's target SDK version, this call always allows batching of the alarm.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setExactAndAllowWhileIdle(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExactAndAllowWhileIdle(int, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

`RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

setExact

added in API level 19 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setExact (int type,
               long triggerAtMillis,
               PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule an alarm to be delivered precisely at the stated time.

This method is like `set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)), but does not permit the OS to adjust the delivery time. The alarm will be delivered as nearly as possible to the requested trigger time.

Note: only alarms for which there is a strong demand for exact-time delivery (such as an alarm clock ringing at the requested time) should be scheduled as exact. Applications are strongly discouraged from using exact alarms unnecessarily as they reduce the OS's ability to minimize battery use.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent))

`setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

`RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

setExact

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setExact (int type,
               long triggerAtMillis,
               String (https://developer.android.com/reference/java/lang/String.html) tag,
               AlarmManager.OnAlarmListener (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) liste
               Handler (https://developer.android.com/reference/android/os/Handler.html) targetHandler)
```

Direct callback version of `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)). Rather than supplying a PendingIntent to be sent when the alarm time is reached, this variant supplies an `AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) instance that will be invoked at that time.

The OnAlarmListener's `onAlarm()` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html#onAlarm()) method will be invoked via the specified target Handler, or on the application's main looper if `null` is passed as the `targetHandler` parameter.

Parameters	
`type`	`int` Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`
`tag`	`String`
`listener`	`AlarmManager.OnAlarmListener`
`targetHandler`	`Handler`

setExactAndAllowWhileIdle

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setExactAndAllowWhileIdle (int type,
               long triggerAtMillis,
               PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Like `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)), but this alarm will be allowed to execute even when the system is in low-power idle modes. If you don't need exact scheduling of the alarm but still need to execute while idle, consider using `setAndAllowWhileIdle(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setAndAllowWhileIdle(int, long, android.app.PendingIntent)). This type of alarm must **only** be used for situations where it is actually required that the alarm go off while in idle -- a reasonable example would be for a calendar notification that should make a sound so the user is aware of it. When the alarm is dispatched, the app will also be added to the system's temporary whitelist for approximately 10 seconds to allow that application to acquire further wake locks in which to complete its work.

These alarms can significantly impact the power use of the device when idle (and thus cause significant battery blame to the app scheduling them), so they should be used with care. To reduce abuse, there are restrictions on how frequently these alarms will go off for a particular application. Under normal system operation, it will not dispatch these alarms more than about every minute (at which point every such pending alarm is dispatched); when in low-power idle modes this duration may be significantly longer, such as 15 minutes.

Unlike other alarms, the system is free to reschedule this type of alarm to happen out of order with any other alarms, even those from the same app. This will clearly happen when the device is idle (since this alarm can go off while idle, when any other alarms from the app will be held until later), but may also happen even when not idle. Note that the OS will allow itself more flexibility for scheduling these alarms than regular exact alarms, since the application has opted into this behavior. When the device is idle it may take even more liberties with scheduling in order to optimize for battery life.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int,

	android.content.Intent, int)).

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent))

`setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

`RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

setInexactRepeating

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setInexactRepeating (int type,
                long triggerAtMillis,
                long intervalMillis,
                PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule a repeating alarm that has inexact trigger time requirements; for example, an alarm that repeats every hour, but not necessarily at the top of every hour. These alarms are more power-efficient than the strict recurrences traditionally supplied by `setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent)), since the system can adjust alarms' delivery times to cause them to fire simultaneously, avoiding waking the device from sleep more than necessary.

Your alarm's first trigger will not be before the requested time, but it might not occur for almost a full interval after that time. In addition, while the overall period of the repeating alarm will be as requested, the time between any two successive firings of the alarm may vary. If your application demands very low jitter, use one-shot alarms with an appropriate window instead; see `setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)) and `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)).

> As of API 19, all repeating alarms are inexact. Because this method has been available since API 3, your application can safely call it and be assured that it will get similar behavior on both current and older versions of Android.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should first go off, using the appropriate clock (depending on the alarm type). This is inexact: the alarm will not fire before this time, but there may be a delay of almost an entire alarm interval before the first invocation of the alarm.
`intervalMillis`	`long`: interval in milliseconds between subsequent repeats of the alarm. Prior to API 19, if this is one of INTERVAL_FIFTEEN_MINUTES, INTERVAL_HALF_HOUR, INTERVAL_HOUR, INTERVAL_HALF_DAY, or INTERVAL_DAY then the alarm will be phase-aligned with other alarms to reduce the number of wakeups. Otherwise, the alarm will be set as though the application had called `setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long,

	android.app.PendingIntent)). As of API 19, all repeating alarms will be inexact and subject to batching with other alarms regardless of their stated repeat interval.
operation	PendingIntent: Action to perform when the alarm goes off; typically comes from IntentSender.getBroadcast() (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

Handler (https://developer.android.com/reference/android/os/Handler.html)

set(int, long, PendingIntent) (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

cancel(AlarmManager.OnAlarmListener) (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

sendBroadcast(Intent) (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

registerReceiver(BroadcastReceiver, IntentFilter) (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

filterEquals(Intent) (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

ELAPSED_REALTIME (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

ELAPSED_REALTIME_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

RTC (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

INTERVAL_FIFTEEN_MINUTES (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_FIFTEEN_MINUTES)

INTERVAL_HALF_HOUR (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HALF_HOUR)

INTERVAL_HOUR (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HOUR)

INTERVAL_HALF_DAY (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_HALF_DAY)

INTERVAL_DAY (https://developer.android.com/reference/android/app/AlarmManager.html#INTERVAL_DAY)

setRepeating

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setRepeating (int type,
                long triggerAtMillis,
                long intervalMillis,
                PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule a repeating alarm. **Note: for timing operations (ticks, timeouts, etc) it is easier and much more efficient to use** Handler (https://developer.android.com/reference/android/os/Handler.html)**.** If there is already an alarm scheduled for the same IntentSender, it will first be canceled.

Like set(int, long, PendingIntent) (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)), except you can also supply a period at which the alarm will automatically repeat. This alarm continues repeating until explicitly removed with cancel(AlarmManager.OnAlarmListener) (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener)). If the stated trigger time is in the past, the alarm will be triggered immediately, with an alarm count depending on how far in the past the trigger time is relative to the repeat interval.

If an alarm is delayed (by system sleep, for example, for non _WAKEUP alarm types), a skipped repeat will be delivered as soon as possible. After that, future alarms will be delivered according to the original schedule; they do not drift over time. For example, if you have set a recurring alarm for the top of every hour but the phone was asleep from 7:45 until 8:45, an alarm will be sent as soon as the phone awakens, then the next alarm will be sent at 9:00.

If your application wants to allow the delivery times to drift in order to guarantee that at least a certain time interval always elapses between alarms, then the approach to take is to use one-time alarms, scheduling the next one yourself when handling each alarm delivery.

> **Note:** as of API 19, all repeating alarms are inexact. If your application needs precise delivery times then it must use one-time exact alarms, rescheduling each time as described above. Legacy applications whose targetSdkVersion is earlier than API 19 will continue to have all of their alarms, including repeating alarms, treated as exact.

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`triggerAtMillis`	`long`: time in milliseconds that the alarm should first go off, using the appropriate clock (depending on the alarm type).
`intervalMillis`	`long`: interval in milliseconds between subsequent repeats of the alarm.
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`Handler` (https://developer.android.com/reference/android/os/Handler.html)

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent))

`setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

`RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

`RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

setTime

added in API level 8 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setTime (long millis)
```

Set the system wall clock time. Requires the permission android.permission.SET_TIME.

Parameters	
`millis`	`long`: time in milliseconds since the Epoch

setTimeZone

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setTimeZone (String (https://developer.android.com/reference/java/lang/String.html) timeZone)
```

Sets the system's persistent default time zone. This is the time zone for all apps, even after a reboot. Use `setDefault(TimeZone)` (https://developer.android.com/reference/java/util/TimeZone.html#setDefault(java.util.TimeZone)) if you just want to change the time zone within your app, and even then prefer to pass an explicit `TimeZone` (https://developer.android.com/reference/java/util/TimeZone.html) to APIs that require it rather than changing the time zone for all threads.

On android M and above, it is an error to pass in a non-Olson timezone to this function. Note that this is a bad idea on all Android releases because POSIX and the `TimeZone` class have opposite interpretations of `'+'` and `'-'` in the same non-Olson ID.

Parameters	
`timeZone`	`String`: one of the Olson ids from the list returned by `getAvailableIDs()` (https://developer.android.com/reference/java/util/TimeZone.html#getAvailableIDs())

setWindow

added in API level 19 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setWindow (int type,
                long windowStartMillis,
                long windowLengthMillis,
                PendingIntent (https://developer.android.com/reference/android/app/PendingIntent.html) operation)
```

Schedule an alarm to be delivered within a given window of time. This method is similar to `set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)), but allows the application to precisely control the degree to which its delivery might be adjusted by the OS. This method allows an application to take advantage of the battery optimizations that arise from delivery batching even when it has modest timeliness requirements for its alarms.

This method can also be used to achieve strict ordering guarantees among multiple alarms by ensuring that the windows requested for each alarm do not intersect.

When precise delivery is not required, applications should use the standard `set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent)) method. This will give the OS the most flexibility to minimize wakeups and battery use. For alarms that must be delivered at precisely-specified times with no acceptable variation, applications can use `setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)).

Parameters	
`type`	`int`: type of alarm. Value is `RTC_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), `RTC` (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), `ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or `ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`windowStartMillis`	`long`: The earliest time, in milliseconds, that the alarm should be delivered, expressed in the appropriate clock's units (depending on the alarm type).
`windowLengthMillis`	`long`: The length of the requested delivery window, in milliseconds. The alarm will be delivered no later than this many milliseconds after `windowStartMillis`. Note that this parameter is a *duration,* not the timestamp of the end of the window.
`operation`	`PendingIntent`: Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()` (https://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast(android.content.Context, int, android.content.Intent, int)).

See also:

`set(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#set(int, long, android.app.PendingIntent))

`setExact(int, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent))

`setRepeating(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent))

`cancel(AlarmManager.OnAlarmListener)` (https://developer.android.com/reference/android/app/AlarmManager.html#cancel(android.app.AlarmManager.OnAlarmListener))

`sendBroadcast(Intent)` (https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent))

`registerReceiver(BroadcastReceiver, IntentFilter)` (https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))

`filterEquals(Intent)` (https://developer.android.com/reference/android/content/Intent.html#filterEquals(android.content.Intent))

`ELAPSED_REALTIME` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME)

`ELAPSED_REALTIME_WAKEUP` (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP)

RTC (https://developer.android.com/reference/android/app/AlarmManager.html#RTC)

RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP)

setWindow

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
void setWindow (int type,
                long windowStartMillis,
                long windowLengthMillis,
                String (https://developer.android.com/reference/java/lang/String.html) tag,
                AlarmManager.OnAlarmListener (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) liste
                Handler (https://developer.android.com/reference/android/os/Handler.html) targetHandler)
```

Direct callback version of `setWindow(int, long, long, PendingIntent)` (https://developer.android.com/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)). Rather than supplying a PendingIntent to be sent when the alarm time is reached, this variant supplies an `AlarmManager.OnAlarmListener` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html) instance that will be invoked at that time.

The OnAlarmListener `onAlarm()` (https://developer.android.com/reference/android/app/AlarmManager.OnAlarmListener.html#onAlarm()) method will be invoked via the specified target Handler, or on the application's main looper if `null` is passed as the `targetHandler` parameter.

Parameters	
`type`	`int` Value is RTC_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#RTC_WAKEUP), RTC (https://developer.android.com/reference/android/app/AlarmManager.html#RTC), ELAPSED_REALTIME_WAKEUP (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME_WAKEUP) or ELAPSED_REALTIME (https://developer.android.com/reference/android/app/AlarmManager.html#ELAPSED_REALTIME).
`windowStartMillis`	`long`
`windowLengthMillis`	`long`
`tag`	`String`
`listener`	`AlarmManager.OnAlarmListener`
`targetHandler`	`Handler`