public class

Summary: Constants | Methods | Inherited Methods | [Expand All]

Added in API level 1

AlarmManager

extends Object

java.lang.Object
  ↳android.app.AlarmManager

Class Overview

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 (/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()` `(/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 (/reference/android/os/Handler.html)`.

> **Note:** Beginning with API 19 (`KITKAT` `(/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)` `(/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent))` and `setExact(int, long, PendingIntent)` `(/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.

You do not instantiate this class directly; instead, retrieve it through `Context.getSystemService(Context.ALARM_SERVICE)` `(/reference/android/content/Context.html#getSystemService(java.lang.String))`.

Summary

Constants	
int ELAPSED_REALTIME	Alarm time in `SystemClock.elapsedRealtime()` (time since boot, including sleep).
int ELAPSED_REALTIME_WAKEUP	Alarm time in `SystemClock.elapsedRealtime()` (time since boot, including sleep), which will wake up the device when it goes off.
long INTERVAL_DAY	Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` when running on Android prior to API 19.
long INTERVAL_FIFTEEN_MINUTES	Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` when running on Android prior to API 19.
long INTERVAL_HALF_DAY	Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` when running on Android prior to API 19.
long INTERVAL_HALF_HOUR	Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` when running on Android prior to API 19.
long INTERVAL_HOUR	Available inexact recurrence interval recognized by `setInexactRepeating(int, long, long, PendingIntent)` when running on Android prior to API 19.
int RTC	Alarm time in `System.currentTimeMillis()` (wall clock time in UTC).
int RTC_WAKEUP	Alarm time in `System.currentTimeMillis()` (wall clock time in UTC), which will wake up the device when it goes off.

Public Methods	
void	cancel (PendingIntent operation) Remove any alarms with a matching `Intent`.
void	set (int type, long triggerAtMillis, PendingIntent operation) Schedule an alarm.
void	setExact (int type, long triggerAtMillis, PendingIntent operation) Schedule an alarm to be delivered precisely at the stated time.
	setInexactRepeating (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation)

void	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.
void	setRepeating (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation) Schedule a repeating alarm.
void	setTime (long millis) Set the system wall clock time.
void	setTimeZone (String timeZone) Set the system default time zone.
void	setWindow (int type, long windowStartMillis, long windowLengthMillis, PendingIntent operation) Schedule an alarm to be delivered within a given window of time.

Inherited Methods [Expand]

▸ From class java.lang.Object

Constants

public static final int **ELAPSED_REALTIME**

Added in API level 1

Alarm time in `SystemClock.elapsedRealtime()` (/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)

public static final int **ELAPSED_REALTIME_WAKEUP**

Added in API level 1

Alarm time in `SystemClock.elapsedRealtime()` (/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)

public static final long **INTERVAL_DAY**

Added in API level 3

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

Constant Value: 86400000 (0x0000000005265c00)

public static final long **INTERVAL_FIFTEEN_MINUTES**

Added in API level 3

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

Constant Value: 900000 (0x00000000000dbba0)

public static final long **INTERVAL_HALF_DAY**

Added in API level 3

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

Constant Value: 43200000 (0x0000000002932e00)

public static final long **INTERVAL_HALF_HOUR**

Added in API level 3

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

Constant Value: 1800000 (0x00000000001b7740)

public static final long **INTERVAL_HOUR**

Added in API level 3

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

Constant Value: 3600000 (0x000000000036ee80)

public static final int **RTC**

Added in API level 1

Alarm time in `System.currentTimeMillis()` (/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)

public static final int **RTC_WAKEUP**

Added in API level 1

Alarm time in `System.currentTimeMillis()` (/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

public void **cancel** (PendingIntent operation)

Added in API level 1

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

Parameters

operation	IntentSender which matches a previously added IntentSender.

See Also

`set(int, long, PendingIntent)`

public void **set** (int type, long triggerAtMillis, PendingIntent operation)

Added in API level 1

Schedule an alarm. **Note: for timing operations (ticks, timeouts, etc) it is easier and much more efficient to use `Handler` (/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)` (/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)` (/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` (/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)` (/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)) and `setExact(int, long, PendingIntent)` (/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	One of ELAPSED_REALTIME, ELAPSED_REALTIME_WAKEUP, RTC, or RTC_WAKEUP.
triggerAtMillis	time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
operation	Action to perform when the alarm goes off; typically comes from IntentSender.getBroadcast().

See Also

Handler
setExact(int, long, PendingIntent)
setRepeating(int, long, long, PendingIntent)
setWindow(int, long, long, PendingIntent)
cancel(PendingIntent)
sendBroadcast(Intent)
registerReceiver(BroadcastReceiver, IntentFilter)
filterEquals(Intent)
ELAPSED_REALTIME
ELAPSED_REALTIME_WAKEUP
RTC
RTC_WAKEUP

public void **setExact** (int type, long triggerAtMillis, PendingIntent operation)

Added in API level 19

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

This method is like set(int, long, PendingIntent) (/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	One of ELAPSED_REALTIME, ELAPSED_REALTIME_WAKEUP, RTC, or RTC_WAKEUP.
triggerAtMillis	time in milliseconds that the alarm should go off, using the appropriate clock (depending on the alarm type).
operation	Action to perform when the alarm goes off; typically comes from IntentSender.getBroadcast().

See Also

set(int, long, PendingIntent)
setRepeating(int, long, long, PendingIntent)
setWindow(int, long, long, PendingIntent)
cancel(PendingIntent)
sendBroadcast(Intent)
registerReceiver(BroadcastReceiver, IntentFilter)
filterEquals(Intent)
ELAPSED_REALTIME
ELAPSED_REALTIME_WAKEUP
RTC
RTC_WAKEUP

public void **setInexactRepeating** (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation)

Added in API level 3

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) (/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) (/reference/android/app/AlarmManager.html#setWindow(int, long, long, android.app.PendingIntent)) and setExact(int, long, PendingIntent) (/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	One of ELAPSED_REALTIME, ELAPSED_REALTIME_WAKEUP, RTC, or RTC_WAKEUP.
triggerAtMillis	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	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)`. As of API 19, all repeating alarms will be inexact and subject to batching with other alarms regardless of their stated repeat interval.
operation	Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()`.

See Also

`Handler`
`set(int, long, PendingIntent)`
`cancel(PendingIntent)`
`sendBroadcast(Intent)`
`registerReceiver(BroadcastReceiver, IntentFilter)`
`filterEquals(Intent)`
`ELAPSED_REALTIME`
`ELAPSED_REALTIME_WAKEUP`
`RTC`
`RTC_WAKEUP`
`INTERVAL_FIFTEEN_MINUTES`
`INTERVAL_HALF_HOUR`
`INTERVAL_HOUR`
`INTERVAL_HALF_DAY`
`INTERVAL_DAY`

public void **setRepeating** (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation)

Added in API level 1

Schedule a repeating alarm. **Note: for timing operations (ticks, timeouts, etc) it is easier and much more efficient to use `Handler` (/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)` (/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(PendingIntent)` (/reference/android/app/AlarmManager.html#cancel(android.app.PendingIntent)). 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	One of `ELAPSED_REALTIME`, `ELAPSED_REALTIME_WAKEUP`, `RTC`, or `RTC_WAKEUP`.
triggerAtMillis	time in milliseconds that the alarm should first go off, using the appropriate clock (depending on the alarm type).
intervalMillis	interval in milliseconds between subsequent repeats of the alarm.
operation	Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()`.

See Also

`Handler`
`set(int, long, PendingIntent)`
`setExact(int, long, PendingIntent)`
`setWindow(int, long, long, PendingIntent)`
`cancel(PendingIntent)`
`sendBroadcast(Intent)`
`registerReceiver(BroadcastReceiver, IntentFilter)`
`filterEquals(Intent)`
`ELAPSED_REALTIME`
`ELAPSED_REALTIME_WAKEUP`
`RTC`
`RTC_WAKEUP`

public void **setTime** (long millis)

Added in API level 8

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

Parameters

millis	time in milliseconds since the Epoch

public void **setTimeZone** (String timeZone)

Added in API level 1

Set the system default time zone. Requires the permission android.permission.SET_TIME_ZONE.

Parameters

timeZone	in the format understood by `TimeZone`

public void **setWindow** (int type, long windowStartMillis, long windowLengthMillis, PendingIntent operation) Added in API level 19

Schedule an alarm to be delivered within a given window of time. This method is similar to `set(int, long, PendingIntent)` (/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)` (/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)` (/reference/android/app/AlarmManager.html#setExact(int, long, android.app.PendingIntent)).

Parameters

type	One of ELAPSED_REALTIME, ELAPSED_REALTIME_WAKEUP, RTC, or RTC_WAKEUP.
windowStartMillis	The earliest time, in milliseconds, that the alarm should be delivered, expressed in the appropriate clock's units (depending on the alarm type).
windowLengthMillis	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	Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()`.

See Also

`set(int, long, PendingIntent)`
`setExact(int, long, PendingIntent)`
`setRepeating(int, long, long, PendingIntent)`
`cancel(PendingIntent)`
`sendBroadcast(Intent)`
`registerReceiver(BroadcastReceiver, IntentFilter)`
`filterEquals(Intent)`

ELAPSED_REALTIME
ELAPSED_REALTIME_WAKEUP
RTC
RTC_WAKEUP