Batching

What is batching?

Batching refers to buffering sensor events in a sensor hub and/or hardware FIFO before reporting the events through the Sensors HAL (/devices/sensors/sensors-hal2). The location where sensor events are buffered (sensor hub and/or hardware FIFO) are referred to as "FIFO" on this page. When sensor event batching is not active, sensor events are immediately reported to the Sensors HAL when available.

Note: Batching and the Sensors HAL's **`batch()`** function are separate concepts, although related. Batching refers to buffering sensor events to save power. The **`batch()`** function is used to configure a sensor's reporting frequency and the maximum delay between an event being generated and it being reported to the Sensors HAL.

Batching can enable significant power savings by only waking up the main applications processor (AP), which runs Android, when many sensor events are ready to be processed, instead of waking it for each individual event. The potential power savings is directly correlated to the number of events that the sensor hub and/or FIFO can buffer: there's a greater potential for power savings if more events can be batched. Batching leverages the use of low-power memory in order to reduce the number of high-power AP wake-ups.

Batching can occur only when a sensor possesses a hardware FIFO and/or can buffer events within a sensor hub. In either case, the sensor must report the maximum number of events that can be batched at once through **`SensorInfo.fifoMaxEventCount`**.

If a sensor has space reserved within a FIFO, the sensor must report the number of reserved events through **`SensorInfo.fifoReservedEventCount`**. If the FIFO is dedicated to the sensor, then **`SensorInfo.fifoReservedEventCount`** is the size of of the FIFO. If the FIFO is shared amongst several sensors, this value may be zero. A common use case is to allow a sensor to use the entire FIFO if it's the only active sensor. If multiple sensors are active, then each sensor is guaranteed space for at least

`SensorInfo.fifoReservedEventCount` events in the FIFO. If a sensor hub is used, the guarantee may be enforced through software.

Sensor events are batched in the following situations:

- The sensor's current maximum report latency is greater than zero, which means that sensor events can be delayed up to the maximum report latency before being

reported through the HAL.

- The AP is in suspend mode and the sensor is a non-wake-up sensor. In this case, events must not wake the AP and must be stored until the AP wakes up.

If a sensor doesn't support batching and the AP is asleep, only wake-up sensor events are reported to the AP and non-wake-up events must not be reported to the AP.

Batching parameters

The two parameters that govern the behavior of batching are sampling_period_ns (#sampling_period_ns) and max_report_latency_ns (#max_report_latency_ns). **`sampling_period_ns`** determines how often a new sensor event is generated, and **`max_report_latency_ns`** determines how long until the event must be reported to the Sensors HAL.

sampling_period_ns

What the **`sampling_period_ns`** parameter means depends on the specified sensor's reporting mode:

- Continuous: **`sampling_period_ns`** is the sampling rate, meaning the rate at which events are generated.
- On-change: **`sampling_period_ns`** limits the sampling rate of events, meaning events are generated no faster than every **`sampling_period_ns`** nanoseconds. Periods can be longer than **`sampling_period_ns`** if no event is generated and the measured values don't change for long periods. For more details, see on-change (/devices/sensors/report-modes#on-change) reporting mode.
- One-shot: **`sampling_period_ns`** is ignored. It has no effect.
- Special: For details on how **`sampling_period_ns`** is used for special sensors, see Sensor Types (/devices/sensors/sensor-types).

For more information about the impact of **`sampling_period_ns`** in the different modes, see Reporting modes (/devices/sensors/report-modes).

For continuous and on-change sensors:

- if **`sampling_period_ns`** is less than **`SensorInfo.minDelay`**, then the HAL implementation must silently clamp it to **`max(SensorInfo.minDelay, 1ms)`**. Android

doesn't support the generation of events at more than 1000 Hz.

- if `sampling_period_ns` is greater than `SensorInfo.maxDelay`, then the HAL implementation must silently truncate it to `SensorInfo.maxDelay`.

Physical sensors sometimes have limitations on the rates at which they can run and the accuracy of their clocks. To account for this, the actual sampling frequency can differ from the requested frequency as long as it satisfies the requirements in the table below.

If the requested frequency is	Then the actual frequency must be
below min frequency (<1/maxDelay)	between 90% and 110% of the min frequency
between min and max frequency	between 90% and 220% of the requested frequency
above max frequency (>1/minDelay)	between 90% and 110% of the max frequency, and below 1100 Hz

Note: This contract is valid only at the HAL level, which always has a single client. At the SDK level, applications might get different rates due to multiplexing in the framework. For more details, see Framework (/devices/sensors/sensor-stack#framework).

max_report_latency_ns

`max_report_latency_ns` sets the maximum time in nanoseconds, by which events can be delayed and stored in the hardware FIFO before being reported through the HAL while the AP is awake.

A value of zero signifies that the events must be reported as soon as they are measured, either skipping the FIFO altogether, or emptying the FIFO as soon as one event from the sensor is present.

For example, an accelerometer activated at 50 Hz with `max_report_latency_ns=0` will trigger interrupts 50 times per second when the AP is awake.

When `max_report_latency_ns>0`, sensor events do not need to be reported as soon as they are detected. They can be temporarily stored in the FIFO and reported in batches, as long as no event is delayed by more than `max_report_latency_ns` nanoseconds. This means that all events since the previous batch are recorded and returned at once. This reduces the amount of interrupts sent to the AP and allows the AP to switch to a lower power mode (idle) while the sensor is capturing and batching data.

Each event has a timestamp associated with it. Delaying the time at which an event is reported does not impact the event timestamp. The timestamp must be accurate and correspond to the time at which the event physically happened, not the time it was reported.

Allowing sensor events to be stored temporarily in the FIFO doesn't modify the behavior of submitting events to the HAL; events from different sensors can be interleaved and all events from the same sensor are time-ordered.

Wake-up and non-wake-up events

Sensor events from wake-up sensors (/devices/sensors/suspend-mode#wake-up_sensors) must be stored in one or more wake-up FIFOs. A common design is to have a single, large, shared wake-up FIFO where events from all wake-up sensors are interleaved. Alternatively, you can have one wake-up FIFO per sensor or have dedicated FIFOs for particular wake-up sensors and a shared FIFO for the rest of the wake-up sensors.

Similarly, sensor events from non-wake-up sensors (/devices/sensors/suspend-mode#non-wake-up_sensors) must be stored in one or more non-wake-up FIFOs.

In all cases, wake-up sensor events and non-wake-up sensor events can't be interleaved in the same FIFO. Wake-up events must be stored in a wake-up FIFO, and non-wake-up events must be stored in a non-wake-up FIFO.

For the wake-up FIFO, the single, large, shared FIFO design provides the best power benefits. For the non-wake-up FIFO, the single, large shared FIFO and several small reserved FIFOs designs have similar power characteristics. For more suggestions on how to configure each FIFO, see FIFO allocation priority (#fifo_allocation_priority).

Behavior outside of suspend mode

When the AP is awake (not in suspend mode), events are stored temporarily in FIFOs as long as they are not delayed by more than **`max_report_latency`**.

As long as the AP doesn’t enter suspend mode, no event shall be dropped or lost. If internal FIFOs become full before **`max_report_latency`** elapses, events are reported at that point to ensure that no event is lost.

If several sensors share the same FIFO and the **`max_report_latency`** of one of them elapses, all events from the FIFO are reported, even if the **`max_report_latency`** of the other sensors haven't yet elapsed. This reduces the number of times batches of events are

sensors haven't yet elapsed. This reduces the number of times batches of events are reported. When one event must be reported, all events from all sensors are reported.

For example, if the following sensors are activated:

- accelerometer batched with `max_report_latency` = 20s
- gyroscope batched with `max_report_latency` = 5s

The accelerometer batches are reported at the same time the gyroscope batches are reported (every 5 seconds), even if the accelerometer and the gyroscope do not share the same FIFO.

Behavior in suspend mode

Batching is particularly beneficial for collecting sensor data in the background without keeping the AP awake. Because the sensor drivers and HAL implementation aren't allowed to hold a wake-lock*, the AP can enter the suspend mode even while sensor data is being collected.

The behavior of sensors while the AP is suspended depends on whether the sensor is a wake-up sensor. For more details, see Wake-up sensors (/devices/sensors/suspend-mode#wake-up_sensors).

When a non-wake-up FIFO fills up, it must wrap around and behave like a circular buffer, overwriting older events with the new events replacing the old ones. `max_report_latency` has no impact on non-wake-up FIFOs while in suspend mode.

When a wake-up FIFO fills up, or when the `max_report_latency` of one of the wake-up sensors elapses, the hardware must wake up the AP and report the data.

In both cases (wake-up and non-wake-up), as soon as the AP comes out of suspend mode, a batch is produced with the content of all FIFOs, even if `max_report_latency` of some sensors haven't yet elapsed. This minimizes the risk of the AP having to wake up again soon after it returns to suspend mode and, therefore, minimizes power consumption.

*One notable exception of drivers not being allowed to hold a wake lock is when a wake-up sensor with the continuous reporting mode (/devices/sensors/report-modes#continuous) is activated with `max_report_latency` < 1 second. In this case, the driver can hold a wake lock because the AP doesn't have time to enter suspend mode, as it would be awoken by a wake-up event before reaching the suspend mode.

Precautions to take when batching wake-up sensors

Depending on the device, it might take a few milliseconds for the AP to come out of suspend mode completely and start flushing the FIFO. Enough head room must be allocated in the FIFO to allow the device to come out of suspend mode without the wake-up FIFO overflowing. No events shall be lost, and the **`max_report_latency`** must be respected.

Precautions to take when batching non-wake-up on-change sensors

On-change sensors only generate events when the value they are measuring is changing. If the measured value changes while the AP is in suspend mode, applications expect to receive an event as soon as the AP wakes up. Because of this, batching of non-wake-up (/devices/sensors/suspend-mode#non-wake-up_sensors) on-change sensor events must be performed carefully if the sensor shares its FIFO with other sensors. The last event generated by each on-change sensor must always be saved outside of the shared FIFO so it can never be overwritten by other events. When the AP wakes up, after all events from the FIFO have been reported, the last on-change sensor event must be reported.

Here is a situation to avoid:

1. An application registers to the non-wake-up step counter (on-change) and the non-wake-up accelerometer (continuous), both sharing the same FIFO.
2. The application receives a step counter event **`step_count=1000 steps`**code>.
3. The AP goes to suspend.
4. The user walks 20 steps, causing step counter and accelerometer events to be interleaved, the last step counter event being **`step_count = 1020 steps`**.
5. The user doesn't move for a long time, causing accelerometer events to continue accumulating in the FIFO, eventually overwriting every **`step_count`** event in the shared FIFO.
6. AP wakes up and all events from the FIFO are sent to the application.
7. The application receives only accelerometer events and thinks that the user didn't walk.

By saving the last step counter event outside of the FIFO, the HAL can report this event when the AP wakes up, even if all other step counter events were overwritten by

accelerometer events. This way, the application receives `step_count = 1020 steps` when the AP wakes up.

Implementing batching

To save power, batching must be implemented without the aid of the AP, and the AP must be allowed to suspend during batching.

If batching is performed in a sensor hub, the sensor hub's power usage should be minimized.

The maximum report latency can be modified at any time, in particular while the specified sensor is already enabled; and this shouldn't result in the loss of events.

FIFO allocation priority

On platforms in which hardware FIFO and/or sensor hub buffer size is limited, system designers may have to choose how much FIFO to reserve for each sensor. To help with this choice, here is a list of possible applications when batching is implemented on the different sensors.

High value: Low power pedestrian dead reckoning

Target batching time: 1 to 10 minutes

Sensors to batch:

- Wake-up step detector
- Wake-up game rotation vector at 5 Hz
- Wake-up barometer at 5 Hz
- Wake-up uncalibrated magnetometer at 5 Hz

Batching this data allows performing pedestrian dead reckoning while letting the AP go to suspend.

High value: Medium power intermittent activity/gesture recognition

Target batching time: 3 seconds

Sensors to batch: Non-wake-up accelerometer at 50 Hz

Batching this data allows periodically recognizing arbitrary activities and gestures without having to keep the AP awake while the data is collected.

Medium value: Medium power continuous activity/gesture recognition

Target batching time: 1 to 3 minutes

Sensors to batch: Wake-up accelerometer at 50 Hz

Batching this data allows continuously recognizing arbitrary activities and gestures without having to keep the AP awake while the data is collected.

Medium-high value: Interrupt load reduction

Target batching time: < 1 second

Sensors to batch: any high-frequency sensor, usually non-wake-up.

If the gyroscope is set at 240 Hz, even batching just 10 gyro events can reduce the number of interrupts from 240/second to 24/second.

Medium value: Continuous low-frequency data collection

Target batching time: 1 to 10 minutes

Sensors to batch:

- Wake-up barometer at 1 Hz
- Wake-up humidity sensor at 1 Hz
- Other low-frequency wake-up sensors at similar rates

Allows creating monitoring applications at low power.

Medium-low value: Continuous full-sensors collection

Target batching time: 1 to 10 minutes

Sensors to batch: All wake-up sensors, at high frequencies

Allows full collection of sensor data while leaving the AP in suspend mode. Only consider if

FIFO space isn't an issue.