Input event codes

The input protocol uses a map of types and codes to express input device values to userspace. This document describes the types and codes and how and when they may be used.

A single hardware event generates multiple input events. Each input event contains the new value of a single data item. A special event type, EV_SYN, is used to separate input events into packets of input data changes occurring at the same moment in time. In the following, the term "event" refers to a single input event encompassing a type, code, and value.

The input protocol is a stateful protocol. Events are emitted only when values of event codes have changed. However, the state is maintained within the Linux input subsystem; drivers do not need to maintain the state and may attempt to emit unchanged values without harm. Userspace may obtain the current state of event code values using the EVIOCG* ioctls defined in linux/input.h. The event reports supported by a device are also provided by sysfs in class/input/event*/device/capabilities/, and the properties of a device are provided in class/input/event*/device/properties.

Event types

Event types are groupings of codes under a logical input construct. Each type has a set of applicable codes to be used in generating events. See the Codes section for details on valid codes for each type.

- EV_SYN:
  - Used as markers to separate events. Events may be separated in time or in space, such as with the multitouch protocol.
- EV_KEY:
  - Used to describe state changes of keyboards, buttons, or other key-like devices.
- EV_REL:
  - Used to describe relative axis value changes, e.g. moving the mouse 5 units to the left.
- EV_ABS:
  - Used to describe absolute axis value changes, e.g. describing the coordinates of a touch on a touchscreen.
- EV_MSC:
  - Used to describe miscellaneous input data that do not fit into other types.
- EV_SW:
  - Used to describe binary state input switches.
- EV_LED:
  - Used to turn LEDs on devices on and off.
- EV_SND:
  - Used to output sound to devices.
- EV_REP:
  - Used for autorepeating devices.
- EV_FF:
  - Used to send force feedback commands to an input device.
- EV_PWR:
  - A special type for power button and switch input.
- EV_FF_STATUS:
  - Used to receive force feedback device status.

Event codes

Event codes define the precise type of event.

EV_SYN

EV_SYN event values are undefined. Their usage is defined only by when they are sent in the evdev event stream.

- SYN_REPORT:
  - Used to synchronize and separate events into packets of input data changes occurring at the same moment in time. For example, motion of a mouse may set the REL_X and REL_Y values for one motion, then emit a SYN_REPORT. The next motion will emit more REL_X and REL_Y values and send another SYN_REPORT.
- SYN_CONFIG:
  - TBD
- SYN_MT_REPORT:
  - Used to synchronize and separate touch events. See the multi-touch-protocol.txt document for more information.
- SYN_DROPPED:
  - Used to indicate buffer overrun in the evdev client's event queue. Client should ignore all events up to and including next SYN_REPORT event and query the device (using EVIOCG* ioctls) to obtain its current state.

EV_KEY

EV_KEY events take the form KEY_<name> or BTN_<name>. For example, KEY_A is used to represent the 'A' key on a keyboard. When a key is depressed, an event with the key's code is emitted with value 1. When the key is released, an event is emitted with value 0. Some hardware send events when a key is repeated. These events have a value of 2. In general, KEY_<name> is used for keyboard keys, and BTN_<name> is used for other types of momentary switch events.

A few EV_KEY codes have special meanings:

- BTN_TOOL_<name>:
    - These codes are used in conjunction with input trackpads, tablets, and touchscreens. These devices may be used with fingers, pens, or other tools. When an event occurs and a tool is used, the corresponding BTN_TOOL_<name> code should be set to a value of 1. When the tool is no longer interacting with the input device, the BTN_TOOL_<name> code should be reset to 0. All trackpads, tablets, and touchscreens should use at least one BTN_TOOL_<name> code when events are generated. Likewise all trackpads, tablets, and touchscreens should export only one BTN_TOOL_<name> at a time. To not break existing userspace, it is recommended to not switch tool in one EV_SYN frame but first emitting the old BTN_TOOL_<name> at 0, then emit one SYN_REPORT and then set the new BTN_TOOL_<name> at 1.
- BTN_TOUCH:

    BTN_TOUCH is used for touch contact. While an input tool is determined to be within meaningful physical contact, the value of this property must be set to 1. Meaningful physical contact may mean any contact, or it may mean contact conditioned by an implementation defined property. For example, a touchpad may set the value to 1 only when the touch pressure rises above a certain value. BTN_TOUCH may be combined with BTN_TOOL_<name> codes. For example, a pen tablet may set BTN_TOOL_PEN to 1 and BTN_TOUCH to 0 while the pen is hovering over but not touching the tablet surface.

Note: For appropriate function of the legacy mousedev emulation driver, BTN_TOUCH must be the first evdev code emitted in a synchronization frame.

Note: Historically a touch device with BTN_TOOL_FINGER and BTN_TOUCH was interpreted as a touchpad by userspace, while a similar device without BTN_TOOL_FINGER was interpreted as a touchscreen. For backwards compatibility with current userspace it is recommended to follow this distinction. In the future, this distinction will be deprecated and the device properties ioctl EVIOCGPROP, defined in linux/input.h, will be used to convey the device type.

- BTN_TOOL_FINGER, BTN_TOOL_DOUBLETAP, BTN_TOOL_TRIPLETAP, BTN_TOOL_QUADTAP:
    - These codes denote one, two, three, and four finger interaction on a trackpad or touchscreen. For example, if the user uses two fingers and moves them on the touchpad in an effort to scroll content on screen, BTN_TOOL_DOUBLETAP should be set to value 1 for the duration of the motion. Note that all BTN_TOOL_<name> codes and the BTN_TOUCH code are orthogonal in purpose. A trackpad event generated by finger touches should generate events for one code from each group. At most only one of these BTN_TOOL_<name> codes should have a value of 1 during any synchronization frame.

Note: Historically some drivers emitted multiple of the finger count codes with a value of 1 in the same synchronization frame. This usage is deprecated.

Note: In multitouch drivers, the input_mt_report_finger_count() function should be used to emit these codes. Please see multi-touch-protocol.txt for details.

EV_REL

EV_REL events describe relative changes in a property. For example, a mouse may move to the left by a certain number of units, but its absolute position in space is unknown. If the absolute position is known, EV_ABS codes should be used instead of EV_REL codes.

A few EV_REL codes have special meanings:

- REL_WHEEL, REL_HWHEEL:
    - These codes are used for vertical and horizontal scroll wheels, respectively. The value is the number of detents moved on the wheel, the physical size of which varies by device. For high-resolution wheels this may be an approximation based on the high-resolution scroll events, see REL_WHEEL_HI_RES. These event codes are legacy codes and REL_WHEEL_HI_RES and REL_HWHEEL_HI_RES should be preferred where available.
- REL_WHEEL_HI_RES, REL_HWHEEL_HI_RES:
    - High-resolution scroll wheel data. The accumulated value 120 represents movement by one detent. For devices that do not provide high-resolution scrolling, the value is always a multiple of 120. For devices with high-resolution scrolling, the value may be a fraction of 120.

      If a vertical scroll wheel supports high-resolution scrolling, this code will be emitted in addition to REL_WHEEL or REL_HWHEEL. The REL_WHEEL and REL_HWHEEL may be an approximation based on the high-resolution scroll events. There is no guarantee that the high-resolution data is a multiple of 120 at the time of an emulated REL_WHEEL or REL_HWHEEL event.

EV_ABS

EV_ABS events describe absolute changes in a property. For example, a touchpad may emit coordinates for a touch location.

A few EV_ABS codes have special meanings:

- ABS_DISTANCE:
  - Used to describe the distance of a tool from an interaction surface. This event should only be emitted while the tool is hovering, meaning in close proximity of the device and while the value of the BTN_TOUCH code is 0. If the input device may be used freely in three dimensions, consider ABS_Z instead.
  - BTN_TOOL_<name> should be set to 1 when the tool comes into detectable proximity and set to 0 when the tool leaves detectable proximity. BTN_TOOL_<name> signals the type of tool that is currently detected by the hardware and is otherwise independent of ABS_DISTANCE and/or BTN_TOUCH.
- ABS_MT_<name>:
  - Used to describe multitouch input events. Please see multi-touch-protocol.txt for details.
- ABS_PRESSURE/ABS_MT_PRESSURE:
  - For touch devices, many devices converted contact size into pressure. A finger flattens with pressure, causing a larger contact area and thus pressure and contact size are directly related. This is not the case for other devices, for example digitizers and touchpads with a true pressure sensor ("pressure pads").

    A device should set the resolution of the axis to indicate whether the pressure is in measurable units. If the resolution is zero, the pressure data is in arbitrary units. If the resolution is non-zero, the pressure data is in units/gram. For example, a value of 10 with a resolution of 1 represents 10 gram, a value of 10 with a resolution of 1000 represents 10 microgram.

EV_SW

EV_SW events describe stateful binary switches. For example, the SW_LID code is used to denote when a laptop lid is closed.

Upon binding to a device or resuming from suspend, a driver must report the current switch state. This ensures that the device, kernel, and userspace state is in sync.

Upon resume, if the switch state is the same as before suspend, then the input subsystem will filter out the duplicate switch state reports. The driver does not need to keep the state of the switch at any time.

EV_MSC

EV_MSC events are used for input and output events that do not fall under other categories.

A few EV_MSC codes have special meaning:

- MSC_TIMESTAMP:
  - Used to report the number of microseconds since the last reset. This event should be coded as an uint32 value, which is allowed to wrap around with no special consequence. It is assumed that the time difference between two consecutive events is reliable on a reasonable time scale (hours). A reset to zero can happen, in which case the time since the last event is unknown. If the device does not provide this information, the driver must not provide it to user space.

EV_LED

EV_LED events are used for input and output to set and query the state of various LEDs on devices.

EV_REP

EV_REP events are used for specifying autorepeating events.

EV_SND

EV_SND events are used for sending sound commands to simple sound output devices.

EV_FF

EV_FF events are used to initialize a force feedback capable device and to cause such device to feedback.

EV_PWR

EV_PWR events are a special type of event used specifically for power management. Its usage is not well defined. To be addressed later.

Device properties

Normally, userspace sets up an input device based on the data it emits, i.e., the event types. In the case of two devices emitting the same event types, additional information can be provided in the form of device properties.

INPUT_PROP_DIRECT + INPUT_PROP_POINTER

The INPUT_PROP_DIRECT property indicates that device coordinates should be directly mapped to screen coordinates (not taking into account trivial transformations, such as scaling, flipping and rotating). Non-direct input devices require non-trivial transformation, such as absolute to relative transformation for touchpads. Typical direct input devices: touchscreens, drawing tablets; non-direct devices: touchpads, mice.

The INPUT_PROP_POINTER property indicates that the device is not transposed on the screen and thus requires use of an on-screen pointer to trace user's movements. Typical pointer devices: touchpads, tablets, mice; non-pointer device: touchscreen.

If neither INPUT_PROP_DIRECT or INPUT_PROP_POINTER are set, the property is considered undefined and the device type should be deduced in the traditional way, using emitted event types.

INPUT_PROP_BUTTONPAD

For touchpads where the button is placed beneath the surface, such that pressing down on the pad causes a button click, this property should be set. Common in Clickpad notebooks and Macbooks from 2009 and onwards.

Originally, the buttonpad property was coded into the bcm5974 driver version field under the name integrated button. For backwards compatibility, both methods need to be checked in userspace.

INPUT_PROP_SEMI_MT

Some touchpads, most common between 2008 and 2011, can detect the presence of multiple contacts without resolving the individual positions; only the number of contacts and a rectangular shape is known. For such touchpads, the SEMI_MT property should be set.

Depending on the device, the rectangle may enclose all touches, like a bounding box, or just some of them, for instance the two most recent touches. The diversity makes the rectangle of limited use, but some gestures can normally be extracted from it.

If INPUT_PROP_SEMI_MT is not set, the device is assumed to be a true MT device.

INPUT_PROP_TOPBUTTONPAD

Some laptops, most notably the Lenovo 40 series provide a trackstick device but do not have physical buttons associated with the trackstick device. Instead, the top area of the touchpad is marked to show visual/haptic areas for left, middle, right buttons intended to be used with the trackstick.

If INPUT_PROP_TOPBUTTONPAD is set, userspace should emulate buttons accordingly. This property does not affect kernel behavior. The kernel does not provide button emulation for such devices but treats them as any other INPUT_PROP_BUTTONPAD device.

INPUT_PROP_ACCELEROMETER

Directional axes on this device (absolute and/or relative x, y, z) represent accelerometer data. Some devices also report gyroscope data, which devices can report through the rotational axes (absolute and/or relative rx, ry, rz).

All other axes retain their meaning. A device must not mix regular directional axes and accelerometer axes on the same event node.

Guidelines

The guidelines below ensure proper single-touch and multi-finger functionality. For multi-touch functionality, see the multi-touch-protocol.rst document for more information.

Mice

REL_{X,Y} must be reported when the mouse moves. BTN_LEFT must be used to report the primary button press. BTN_{MIDDLE,RIGHT,4,5,etc.} should be used to report further buttons of the device. REL_WHEEL and REL_HWHEEL should be used to report scroll wheel events where available.

Touchscreens

ABS_{X,Y} must be reported with the location of the touch. BTN_TOUCH must be used to report when a touch is active on the screen. BTN_{MOUSE,LEFT,MIDDLE,RIGHT} must not be reported as the result of touch contact. BTN_TOOL_<name> events should be reported where possible.

For new hardware, INPUT_PROP_DIRECT should be set.

Trackpads

Legacy trackpads that only provide relative position information must report events like mice described above.

Trackpads that provide absolute touch position must report ABS_{X,Y} for the location of the touch. BTN_TOUCH should be used to report when a touch is active on the trackpad. Where multi-finger support is available, BTN_TOOL_<name> should be used to report the number of touches active on the trackpad.

For new hardware, INPUT_PROP_POINTER should be set.

Tablets

BTN_TOOL_<name> events must be reported when a stylus or other tool is active on the tablet. ABS_{X,Y} must be reported with the location of the tool. BTN_TOUCH should be used to report when the tool is in contact with the tablet. BTN_{STYLUS,STYLUS2} should be used to report buttons on the tool itself. Any button may be used for buttons on the tablet except BTN_{MOUSE,LEFT}. BTN_{0,1,2,etc} are good generic codes for unlabeled buttons. Do not use meaningful buttons, like BTN_FORWARD, unless the button is labeled for that purpose on the device.

For new hardware, both INPUT_PROP_DIRECT and INPUT_PROP_POINTER should be set.