public final class
MotionEvent
extends InputEvent
implements Parcelable

Summary: Nested Classes | Constants | Inherited Constants | Fields | Inherited Fields | Methods | Protected Methods | Inherited Methods | [Expand All]
Added in API level 1

java.lang.Object
↳ android.view.InputEvent
↳ android.view.MotionEvent

Class Overview

Object used to report movement (mouse, pen, finger, trackball) events. Motion events may hold either absolute or relative movements and other data, depending on the type of device.

Overview

Motion events describe movements in terms of an action code and a set of axis values. The action code specifies the state change that occurred such as a pointer going down or up. The axis values describe the position and other movement properties.

For example, when the user first touches the screen, the system delivers a touch event to the appropriate View (/reference/android/view/View.html) with the action code ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) and a set of axis values that include the X and Y coordinates of the touch and information about the pressure, size and orientation of the contact area.

Some devices can report multiple movement traces at the same time. Multi-touch screens emit one movement trace for each finger. The individual fingers or other objects that generate movement traces are referred to as *pointers*. Motion events contain information about all of the pointers that are currently active even if some of them have not moved since the last event was delivered.

The number of pointers only ever changes by one as individual pointers go up and down, except when the gesture is canceled.

Each pointer has a unique id that is assigned when it first goes down (indicated by ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) or ACTION_POINTER_DOWN (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN)). A pointer id remains valid until the pointer eventually goes up (indicated by ACTION_UP (/reference/android/view/MotionEvent.html#ACTION_UP) or ACTION_POINTER_UP (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP)) or when the gesture is canceled (indicated by ACTION_CANCEL (/reference/android/view/MotionEvent.html#ACTION_CANCEL)).

The MotionEvent class provides many methods to query the position and other properties of pointers, such as getX(int) (/reference/android/view/MotionEvent.html#getX(int)), getY(int) (/reference/android/view/MotionEvent.html#getY(int)), getAxisValue(int) (/reference/android/view/MotionEvent.html#getAxisValue(int)), getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)), getToolType(int) (/reference/android/view/MotionEvent.html#getToolType(int)), and many others. Most of these methods accept the pointer index as a parameter rather than the pointer id. The pointer index of each pointer in the event ranges from 0 to one less than the value returned by getPointerCount() (/reference/android/view/MotionEvent.html#getPointerCount()).

The order in which individual pointers appear within a motion event is undefined. Thus the pointer index of a pointer can change from one event to the next but the pointer id of a pointer is guaranteed to remain constant as long as the pointer remains active. Use the getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) method to obtain the pointer id of a pointer to track it across all subsequent motion events in a gesture. Then for successive motion events, use the findPointerIndex(int) (/reference/android/view/MotionEvent.html#findPointerIndex(int)) method to obtain the pointer index for a given pointer id in that motion event.

Mouse and stylus buttons can be retrieved using getButtonState() (/reference/android/view/MotionEvent.html#getButtonState()). It is a good idea to check the button state while handling ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) as part of a touch event. The application may choose to perform some different action if the touch event starts due to a secondary button click, such as presenting a context menu.

Batching

For efficiency, motion events with ACTION_MOVE (/reference/android/view/MotionEvent.html#ACTION_MOVE) may batch together multiple movement samples within a single object. The most current pointer coordinates are available using getX(int) (/reference/android/view/MotionEvent.html#getX(int)) and getY(int) (/reference/android/view/MotionEvent.html#getY(int)). Earlier coordinates within the batch are accessed using getHistoricalX(int, int) (/reference/android/view/MotionEvent.html#getHistoricalX(int, int)) and getHistoricalY(int, int) (/reference/android/view/MotionEvent.html#getHistoricalY(int, int)). The coordinates are "historical" only insofar as they are older than the current coordinates in the batch; however, they are still distinct from any other coordinates reported in prior motion events. To process all coordinates in the batch in time order, first consume the historical coordinates then consume the current coordinates.

Example: Consuming all samples for all pointers in a motion event in time order.

```
void printSamples(MotionEvent ev) {
    final int historySize = ev.getHistorySize();
    final int pointerCount = ev.getPointerCount();
    for (int h = 0; h < historySize; h++) {
        System.out.printf("At time %d:", ev.getHistoricalEventTime(h));
        for (int p = 0; p < pointerCount; p++) {
            System.out.printf("  pointer %d: (%f,%f)",
                ev.getPointerId(p), ev.getHistoricalX(p, h), ev.getHistoricalY(p, h));
        }
    }
    System.out.printf("At time %d:", ev.getEventTime());
    for (int p = 0; p < pointerCount; p++) {
        System.out.printf("  pointer %d: (%f,%f)",
            ev.getPointerId(p), ev.getX(p), ev.getY(p));
    }
}
```

Device Types

The interpretation of the contents of a MotionEvent varies significantly depending on the source class of the device.

On pointing devices with source class SOURCE_CLASS_POINTER (/reference/android/view/InputDevice.html#SOURCE_CLASS_POINTER) such as touch screens, the pointer coordinates specify absolute positions such as view X/Y coordinates. Each complete gesture is represented by a sequence of motion events with actions that describe pointer state transitions and movements. A gesture starts with a motion event with ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) that provides the location of the first pointer down. As each additional pointer that goes down or up, the framework will generate a motion event with ACTION_POINTER_DOWN (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN) or ACTION_POINTER_UP (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP) accordingly. Pointer movements are described by motion events with ACTION_MOVE (/reference/android/view/MotionEvent.html#ACTION_MOVE). Finally, a gesture end either when the final pointer goes up as represented by a motion event with ACTION_UP (/reference/android/view/MotionEvent.html#ACTION_UP) or when gesture is canceled with ACTION_CANCEL (/reference/android/view/MotionEvent.html#ACTION_CANCEL).

Some pointing devices such as mice may support vertical and/or horizontal scrolling. A scroll event is reported as a generic motion event with ACTION_SCROLL (/reference/android/view/MotionEvent.html#ACTION_SCROLL) that includes the relative scroll offset in the AXIS_VSCROLL (/reference/android/view/MotionEvent.html#AXIS_VSCROLL) and AXIS_HSCROLL (/reference/android/view/MotionEvent.html#AXIS_HSCROLL) axes. See getAxisValue(int) (/reference/android/view/MotionEvent.html#getAxisValue(int)) for information about retrieving these additional axes.

On trackball devices with source class SOURCE_CLASS_TRACKBALL (/reference/android/view/InputDevice.html#SOURCE_CLASS_TRACKBALL), the pointer coordinates specify relative movements as X/Y deltas. A trackball gesture consists of a sequence of movements described by motion events with ACTION_MOVE (/reference/android/view/MotionEvent.html#ACTION_MOVE) interspersed with occasional ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) or ACTION_UP (/reference/android/view/MotionEvent.html#ACTION_UP) motion events when the trackball button is pressed or released.

On joystick devices with source class SOURCE_CLASS_JOYSTICK (/reference/android/view/InputDevice.html#SOURCE_CLASS_JOYSTICK), the pointer coordinates specify the absolute position of the joystick axes. The joystick axis values are normalized to a range of -1.0 to 1.0 where 0.0 corresponds to the center position. More information about the set of available axes and the range of motion can be obtained using getMotionRange(int) (/reference/android/view/InputDevice.html#getMotionRange(int)). Some common joystick axes are AXIS_X (/reference/android/view/MotionEvent.html#AXIS_X), AXIS_Y (/reference/android/view/MotionEvent.html#AXIS_Y), AXIS_HAT_X (/reference/android/view/MotionEvent.html#AXIS_HAT_X), AXIS_HAT_Y (/reference/android/view/MotionEvent.html#AXIS_HAT_Y), AXIS_Z (/reference/android/view/MotionEvent.html#AXIS_Z) and AXIS_RZ (/reference/android/view/MotionEvent.html#AXIS_RZ).

Refer to InputDevice (/reference/android/view/InputDevice.html) for more information about how different kinds of input devices and sources represent pointer coordinates.

Consistency Guarantees

Motion events are always delivered to views as a consistent stream of events. What constitutes a consistent stream varies depending on the type of device. For touch events, consistency implies that pointers go down one at a time, move around as a group and then go up one at a time or are canceled.

While the framework tries to deliver consistent streams of motion events to views, it cannot guarantee it. Some events may be dropped or modified by containing views in the application before they are delivered thereby making the stream of events inconsistent. Views should always be prepared to handle ACTION_CANCEL (/reference/android/view/MotionEvent.html#ACTION_CANCEL) and should tolerate anomalous situations such as receiving a new ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) without first having received an ACTION_UP (/reference/android/view/MotionEvent.html#ACTION_UP) for the prior gesture.

Summary

Nested Classes

class MotionEvent.PointerCoords	Transfer object for pointer coordinates.
class MotionEvent.PointerProperties	Transfer object for pointer properties.

Constants

int ACTION_CANCEL	Constant for `getActionMasked()`: The current gesture has been aborted.
int ACTION_DOWN	Constant for `getActionMasked()`: A pressed gesture has started, the motion contains the initial starting location.
int ACTION_HOVER_ENTER	Constant for `getActionMasked()`: The pointer is not down but has entered the boundaries of a window or view.
int ACTION_HOVER_EXIT	Constant for `getActionMasked()`: The pointer is not down but has exited the boundaries of a window or view.
int ACTION_HOVER_MOVE	Constant for `getActionMasked()`: A change happened but the pointer is not down (unlike `ACTION_MOVE`).
int ACTION_MASK	Bit mask of the parts of the action code that are the action itself.
int ACTION_MOVE	Constant for `getActionMasked()`: A change has happened during a press gesture (between `ACTION_DOWN` and `ACTION_UP`).
int ACTION_OUTSIDE	Constant for `getActionMasked()`: A movement has happened outside of the normal bounds of the UI element.
int ACTION_POINTER_1_DOWN	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_DOWN`.*
int ACTION_POINTER_1_UP	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_UP`.*
int ACTION_POINTER_2_DOWN	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_DOWN`.*
int ACTION_POINTER_2_UP	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_UP`.*
int ACTION_POINTER_3_DOWN	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_DOWN`.*
int ACTION_POINTER_3_UP	*This constant was deprecated in API level 8. Use `ACTION_POINTER_INDEX_MASK` to retrieve the data index associated with `ACTION_POINTER_UP`.*

int ACTION_POINTER_DOWN	Constant for `getActionMasked()`: A non-primary pointer has gone down.
int ACTION_POINTER_ID_MASK	*This constant was deprecated in API level 8. Renamed to `ACTION_POINTER_INDEX_MASK` to match the actual data contained in these bits.*
int ACTION_POINTER_ID_SHIFT	*This constant was deprecated in API level 8. Renamed to `ACTION_POINTER_INDEX_SHIFT` to match the actual data contained in these bits.*
int ACTION_POINTER_INDEX_MASK	Bits in the action code that represent a pointer index, used with `ACTION_POINTER_DOWN` and `ACTION_POINTER_UP`.
int ACTION_POINTER_INDEX_SHIFT	Bit shift for the action bits holding the pointer index as defined by `ACTION_POINTER_INDEX_MASK`.
int ACTION_POINTER_UP	Constant for `getActionMasked()`: A non-primary pointer has gone up.
int ACTION_SCROLL	Constant for `getActionMasked()`: The motion event contains relative vertical and/or horizontal scroll offsets.
int ACTION_UP	Constant for `getActionMasked()`: A pressed gesture has finished, the motion contains the final release location as well as any intermediate points since the last down or move event.
int AXIS_BRAKE	Axis constant: Brake axis of a motion event.
int AXIS_DISTANCE	Axis constant: Distance axis of a motion event.
int AXIS_GAS	Axis constant: Gas axis of a motion event.
int AXIS_GENERIC_1	Axis constant: Generic 1 axis of a motion event.
int AXIS_GENERIC_10	Axis constant: Generic 10 axis of a motion event.
int AXIS_GENERIC_11	Axis constant: Generic 11 axis of a motion event.
int AXIS_GENERIC_12	Axis constant: Generic 12 axis of a motion event.
int AXIS_GENERIC_13	Axis constant: Generic 13 axis of a motion event.
int AXIS_GENERIC_14	Axis constant: Generic 14 axis of a motion event.
int AXIS_GENERIC_15	Axis constant: Generic 15 axis of a motion event.
int AXIS_GENERIC_16	Axis constant: Generic 16 axis of a motion event.
int AXIS_GENERIC_2	Axis constant: Generic 2 axis of a motion event.
int AXIS_GENERIC_3	Axis constant: Generic 3 axis of a motion event.
int AXIS_GENERIC_4	Axis constant: Generic 4 axis of a motion event.
int AXIS_GENERIC_5	Axis constant: Generic 5 axis of a motion event.
int AXIS_GENERIC_6	Axis constant: Generic 6 axis of a motion event.
int AXIS_GENERIC_7	Axis constant: Generic 7 axis of a motion event.
int AXIS_GENERIC_8	Axis constant: Generic 8 axis of a motion event.
int AXIS_GENERIC_9	Axis constant: Generic 9 axis of a motion event.
int AXIS_HAT_X	Axis constant: Hat X axis of a motion event.
int AXIS_HAT_Y	Axis constant: Hat Y axis of a motion event.
int AXIS_HSCROLL	Axis constant: Horizontal Scroll axis of a motion event.
int AXIS_LTRIGGER	Axis constant: Left Trigger axis of a motion event.
int AXIS_ORIENTATION	Axis constant: Orientation axis of a motion event.
int AXIS_PRESSURE	Axis constant: Pressure axis of a motion event.
int AXIS_RTRIGGER	Axis constant: Right Trigger axis of a motion event.
int AXIS_RUDDER	Axis constant: Rudder axis of a motion event.
int AXIS_RX	Axis constant: X Rotation axis of a motion event.
int AXIS_RY	Axis constant: Y Rotation axis of a motion event.
int AXIS_RZ	Axis constant: Z Rotation axis of a motion event.
int AXIS_SIZE	Axis constant: Size axis of a motion event.
int AXIS_THROTTLE	Axis constant: Throttle axis of a motion event.
int AXIS_TILT	Axis constant: Tilt axis of a motion event.
int AXIS_TOOL_MAJOR	Axis constant: ToolMajor axis of a motion event.
int AXIS_TOOL_MINOR	Axis constant: ToolMinor axis of a motion event.
int AXIS_TOUCH_MAJOR	Axis constant: TouchMajor axis of a motion event.
int AXIS_TOUCH_MINOR	Axis constant: TouchMinor axis of a motion event.
int AXIS_VSCROLL	Axis constant: Vertical Scroll axis of a motion event.
int AXIS_WHEEL	Axis constant: Wheel axis of a motion event.
int AXIS_X	Axis constant: X axis of a motion event.
int AXIS_Y	Axis constant: Y axis of a motion event.
int AXIS_Z	Axis constant: Z axis of a motion event.
int BUTTON_BACK	Button constant: Back button pressed (mouse back button).
int BUTTON_FORWARD	Button constant: Forward button pressed (mouse forward button).
int BUTTON_PRIMARY	Button constant: Primary button (left mouse button).
int BUTTON_SECONDARY	Button constant: Secondary button (right mouse button, stylus first button).
int BUTTON_TERTIARY	Button constant: Tertiary button (middle mouse button, stylus second button).
int EDGE_BOTTOM	Flag indicating the motion event intersected the bottom edge of the screen.
int EDGE_LEFT	Flag indicating the motion event intersected the left edge of the screen.
int EDGE_RIGHT	Flag indicating the motion event intersected the right edge of the screen.
int EDGE_TOP	Flag indicating the motion event intersected the top edge of the screen.
int FLAG_WINDOW_IS_OBSCURED	This flag indicates that the window that received this motion event is partly or wholly obscured by another visible window above it.
int INVALID_POINTER_ID	An invalid pointer id.
int TOOL_TYPE_ERASER	Tool type constant: The tool is an eraser or a stylus being used in an inverted posture.
int TOOL_TYPE_FINGER	Tool type constant: The tool is a finger.
int TOOL_TYPE_MOUSE	Tool type constant: The tool is a mouse or trackpad.
int TOOL_TYPE_STYLUS	Tool type constant: The tool is a stylus.
int TOOL_TYPE_UNKNOWN	Tool type constant: Unknown tool type.

Inherited Constants [Expand]

▸ From interface android.os.Parcelable

Fields

public static final Creator<MotionEvent> CREATOR

Inherited Fields [Expand]

▸ From class android.view.InputEvent

Public Methods

static String	actionToString (int action) Returns a string that represents the symbolic name of the specified unmasked action such as "ACTION_DOWN", "ACTION_POINTER_DOWN(3)" or an equivalent numeric constant such as "35" if unknown.
final void	addBatch (long eventTime, PointerCoords[] pointerCoords, int metaState) Add a new movement to the batch of movements in this event.
final void	addBatch (long eventTime, float x, float y, float pressure, float size, int metaState) Add a new movement to the batch of movements in this event.
static int	axisFromString (String symbolicName) Gets an axis by its symbolic name such as "AXIS_X" or an equivalent numeric constant such as "42".
static String	axisToString (int axis) Returns a string that represents the symbolic name of the specified axis such as "AXIS_X" or an equivalent numeric constant such as "42" if unknown.
final int	findPointerIndex (int pointerId) Given a pointer identifier, find the index of its data in the event.
final int	getAction () Return the kind of action being performed.
final int	getActionIndex () For `ACTION_POINTER_DOWN` or `ACTION_POINTER_UP` as returned by `getActionMasked()`, this returns the associated pointer index.
final int	getActionMasked () Return the masked action being performed, without pointer index information.
final float	getAxisValue (int axis) `getAxisValue(int)` for the first pointer index (may be an arbitrary pointer identifier).
final float	getAxisValue (int axis, int pointerIndex) Returns the value of the requested axis for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).
final int	getButtonState () Gets the state of all buttons that are pressed such as a mouse or stylus button.
final int	getDeviceId () Gets the id for the device that this event came from.
final long	getDownTime () Returns the time (in ms) when the user originally pressed down to start a stream of position events.
final int	getEdgeFlags () Returns a bitfield indicating which edges, if any, were touched by this MotionEvent.
final long	getEventTime () Retrieve the time this event occurred, in the `uptimeMillis()` time base.
final int	getFlags () Gets the motion event flags.
final float	getHistoricalAxisValue (int axis, int pointerIndex, int pos) Returns the historical value of the requested axis, as per `getAxisValue(int, int)`, occurred between this event and the previous event for the given pointer.
final float	getHistoricalAxisValue (int axis, int pos) `getHistoricalAxisValue(int, int, int)` for the first pointer index (may be an arbitrary pointer identifier).
final long	getHistoricalEventTime (int pos) Returns the time that a historical movement occurred between this event and the previous event, in the `uptimeMillis()` time base.

final float **getHistoricalOrientation** (int pointerIndex, int pos)
Returns a historical orientation coordinate, as per `getOrientation(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalOrientation** (int pos)
`getHistoricalOrientation(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final void **getHistoricalPointerCoords** (int pointerIndex, int pos, MotionEvent.PointerCoords outPointerCoords)
Populates a `MotionEvent.PointerCoords` object with historical pointer coordinate data, as per `getPointerCoords(int, MotionEvent.PointerCoords)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalPressure** (int pos)
`getHistoricalPressure(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalPressure** (int pointerIndex, int pos)
Returns a historical pressure coordinate, as per `getPressure(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalSize** (int pos)
`getHistoricalSize(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalSize** (int pointerIndex, int pos)
Returns a historical size coordinate, as per `getSize(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalToolMajor** (int pointerIndex, int pos)
Returns a historical tool major axis coordinate, as per `getToolMajor(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalToolMajor** (int pos)
`getHistoricalToolMajor(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalToolMinor** (int pointerIndex, int pos)
Returns a historical tool minor axis coordinate, as per `getToolMinor(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalToolMinor** (int pos)
`getHistoricalToolMinor(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalTouchMajor** (int pointerIndex, int pos)
Returns a historical touch major axis coordinate, as per `getTouchMajor(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalTouchMajor** (int pos)
`getHistoricalTouchMajor(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalTouchMinor** (int pointerIndex, int pos)
Returns a historical touch minor axis coordinate, as per `getTouchMinor(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalTouchMinor** (int pos)
`getHistoricalTouchMinor(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalX** (int pos)
`getHistoricalX(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalX** (int pointerIndex, int pos)
Returns a historical X coordinate, as per `getX(int)`, that occurred between this event and the previous event for the given pointer.

final float **getHistoricalY** (int pos)
`getHistoricalY(int, int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getHistoricalY** (int pointerIndex, int pos)
Returns a historical Y coordinate, as per `getY(int)`, that occurred between this event and the previous event for the given pointer.

final int **getHistorySize** ()
Returns the number of historical points in this event.

final int **getMetaState** ()
Returns the state of any meta / modifier keys that were in effect when the event was generated.

final float **getOrientation** (int pointerIndex)
Returns the orientation of the touch area and tool area in radians clockwise from vertical for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getOrientation** ()
`getOrientation(int)` for the first pointer index (may be an arbitrary pointer identifier).

final void **getPointerCoords** (int pointerIndex, MotionEvent.PointerCoords outPointerCoords)
Populates a `MotionEvent.PointerCoords` object with pointer coordinate data for the specified pointer index.

final int **getPointerCount** ()
The number of pointers of data contained in this event.

final int **getPointerId** (int pointerIndex)
Return the pointer identifier associated with a particular pointer data index is this event.

final void **getPointerProperties** (int pointerIndex, MotionEvent.PointerProperties outPointerProperties)
Populates a `MotionEvent.PointerProperties` object with pointer properties for the specified pointer index.

final float **getPressure** ()
`getPressure(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getPressure** (int pointerIndex)
Returns the current pressure of this event for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getRawX** ()
Returns the original raw X coordinate of this event.

final float **getRawY** ()
Returns the original raw Y coordinate of this event.

final float **getSize** (int pointerIndex)
Returns a scaled value of the approximate size for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getSize** ()
`getSize(int)` for the first pointer index (may be an arbitrary pointer identifier).

final int **getSource** ()
Gets the source of the event.

final float **getToolMajor** (int pointerIndex)
Returns the length of the major axis of an ellipse that describes the size of the approaching tool for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getToolMajor** ()
`getToolMajor(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getToolMinor** ()
`getToolMinor(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getToolMinor** (int pointerIndex)
Returns the length of the minor axis of an ellipse that describes the size of the approaching tool for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final int **getToolType** (int pointerIndex)
Gets the tool type of a pointer for the given pointer index.

final float **getTouchMajor** ()
`getTouchMajor(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getTouchMajor** (int pointerIndex)
Returns the length of the major axis of an ellipse that describes the touch area at the point of contact for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getTouchMinor** ()
`getTouchMinor(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getTouchMinor** (int pointerIndex)
Returns the length of the minor axis of an ellipse that describes the touch area at the point of contact for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getX** (int pointerIndex)
Returns the X coordinate of this event for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getX** ()
`getX(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getXPrecision** ()
Return the precision of the X coordinates being reported.

final float **getY** (int pointerIndex)
Returns the Y coordinate of this event for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index).

final float **getY** ()
`getY(int)` for the first pointer index (may be an arbitrary pointer identifier).

final float **getYPrecision** ()
Return the precision of the Y coordinates being reported.

static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, PointerProperties[] pointerProperties, PointerCoords[] pointerCoords, int metaState, int buttonState, float xPrecision, float yPrecision, int deviceId, int edgeFlags, int source, int flags)
Create a new MotionEvent, filling in all of the basic values that define the motion.

static MotionEvent **obtain** (long downTime, long eventTime, int action, float x, float y, float pressure, float size, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags)
Create a new MotionEvent, filling in all of the basic values that define the motion.

static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, float x, float y, float pressure, float size, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags)
This method was deprecated in API level 9. Use `obtain(long, long, int, float, float, float, float, int, float, float, int, int)` instead.

static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, int[] pointerIds, PointerCoords[] pointerCoords, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags, int source, int flags)
This method was deprecated in API level 14. Use `obtain(long, long, int, int, PointerProperties[], PointerCoords[], int, int, float, float, int, int, int, int)` instead.

static MotionEvent **obtain** (MotionEvent other)
Create a new MotionEvent, copying from an existing one.

static MotionEvent **obtain** (long downTime, long eventTime, int action, float x, float y, int metaState)
Create a new MotionEvent, filling in a subset of the basic motion values.

static MotionEvent **obtainNoHistory** (MotionEvent other)
Create a new MotionEvent, copying from an existing one, but not including any historical point information.

final void **offsetLocation** (float deltaX, float deltaY)
Adjust this event's location.

final void recycle()
Recycle the MotionEvent, to be re-used by a later caller.
final void setAction(int action)
Sets this event's action.
final void setEdgeFlags(int flags)
Sets the bitfield indicating which edges, if any, were touched by this MotionEvent.
final void setLocation(float x, float y)
Set this event's location.
final void setSource(int source)
Modifies the source of the event.
String toString()
Returns a string containing a concise, human-readable description of this object.
final void transform(Matrix matrix)
Applies a transformation matrix to all of the points in the event.
void writeToParcel(Parcel out, int flags)
Flatten this object in to a Parcel.

Protected Methods

void finalize()
Invoked when the garbage collector has detected that this instance is no longer reachable.

Inherited Methods [Expand]

- From class android.view.InputEvent
- From class java.lang.Object
- From interface android.os.Parcelable

Constants

public static final int **ACTION_CANCEL** Added in API level 1

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): The current gesture has been aborted. You will not receive any more points in it. You should treat this as an up event, but not perform any action that you normally would.

Constant Value: 3 (0x00000003)

public static final int **ACTION_DOWN** Added in API level 1

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): A pressed gesture has started, the motion contains the initial starting location.

This is also a good time to check the button state to distinguish secondary and tertiary button clicks and handle them appropriately. Use `getButtonState()` (/reference/android/view/MotionEvent.html#getButtonState()) to retrieve the button state.

Constant Value: 0 (0x00000000)

public static final int **ACTION_HOVER_ENTER** Added in API level 14

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): The pointer is not down but has entered the boundaries of a window or view.

This action is always delivered to the window or view under the pointer.

This action is not a touch event so it is delivered to `onGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#onGenericMotionEvent(android.view.MotionEvent)) rather than `onTouchEvent(MotionEvent)` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)).

Constant Value: 9 (0x00000009)

public static final int **ACTION_HOVER_EXIT** Added in API level 14

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): The pointer is not down but has exited the boundaries of a window or view.

This action is always delivered to the window or view that was previously under the pointer.

This action is not a touch event so it is delivered to `onGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#onGenericMotionEvent(android.view.MotionEvent)) rather than `onTouchEvent(MotionEvent)` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)).

Constant Value: 10 (0x0000000a)

public static final int **ACTION_HOVER_MOVE** Added in API level 12

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): A change happened but the pointer is not down (unlike `ACTION_MOVE` (/reference/android/view/MotionEvent.html#ACTION_MOVE)). The motion contains the most recent point, as well as any intermediate points since the last hover move event.

This action is always delivered to the window or view under the pointer.

This action is not a touch event so it is delivered to `onGenericMotionEvent(MotionEvent)` (/reference/android/view/View.html#onGenericMotionEvent(android.view.MotionEvent)) rather than `onTouchEvent(MotionEvent)` (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)).

Constant Value: 7 (0x00000007)

public static final int **ACTION_MASK** Added in API level 5

Bit mask of the parts of the action code that are the action itself.

Constant Value: 255 (0x000000ff)

public static final int **ACTION_MOVE** Added in API level 1

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): A change has happened during a press gesture (between `ACTION_DOWN` (/reference/android/view/MotionEvent.html#ACTION_DOWN) and `ACTION_UP` (/reference/android/view/MotionEvent.html#ACTION_UP)). The motion contains the most recent point, as well as any intermediate points since the last down or move event.

Constant Value: 2 (0x00000002)

public static final int **ACTION_OUTSIDE** Added in API level 3

Constant for `getActionMasked()` (/reference/android/view/MotionEvent.html#getActionMasked()): A movement has happened outside of the normal bounds of the UI element. This does not provide a full gesture, but only the initial location of the movement/touch.

Constant Value: 4 (0x00000004)

public static final int **ACTION_POINTER_1_DOWN** Added in API level 5

This constant was deprecated in API level 8.
Use `ACTION_POINTER_INDEX_MASK` (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with `ACTION_POINTER_DOWN` (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN).

Constant Value: 5 (0x00000005)

public static final int **ACTION_POINTER_1_UP** Added in API level 5

This constant was deprecated in API level 8.
Use `ACTION_POINTER_INDEX_MASK` (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with `ACTION_POINTER_UP` (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP).

Constant Value: 6 (0x00000006)

public static final int **ACTION_POINTER_2_DOWN** Added in API level 5

This constant was deprecated in API level 8.
Use `ACTION_POINTER_INDEX_MASK` (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with `ACTION_POINTER_DOWN` (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN).

Constant Value: 261 (0x00000105)

public static final int **ACTION_POINTER_2_UP** Added in API level 5

This constant was deprecated in API level 8.
Use `ACTION_POINTER_INDEX_MASK` (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with `ACTION_POINTER_UP` (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP).

Constant Value: 262 (0x00000106)

public static final int **ACTION_POINTER_3_DOWN** Added in API level 5

This constant was deprecated in API level 8.

Use ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with ACTION_POINTER_DOWN (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN).

Constant Value: 517 (0x00000205)

public static final int **ACTION_POINTER_3_UP** Added in API level 5

This constant was deprecated in API level 8.
Use ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to retrieve the data index associated with ACTION_POINTER_UP (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP).

Constant Value: 518 (0x00000206)

public static final int **ACTION_POINTER_DOWN** Added in API level 5

Constant for getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()): A non-primary pointer has gone down.

Use getActionIndex() (/reference/android/view/MotionEvent.html#getActionIndex()) to retrieve the index of the pointer that changed.

The index is encoded in the ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) bits of the unmasked action returned by getAction() (/reference/android/view/MotionEvent.html#getAction()).

Constant Value: 5 (0x00000005)

public static final int **ACTION_POINTER_ID_MASK** Added in API level 5

This constant was deprecated in API level 8.
Renamed to ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) to match the actual data contained in these bits.

Constant Value: 65280 (0x0000ff00)

public static final int **ACTION_POINTER_ID_SHIFT** Added in API level 5

This constant was deprecated in API level 8.
Renamed to ACTION_POINTER_INDEX_SHIFT (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_SHIFT) to match the actual data contained in these bits.

Constant Value: 8 (0x00000008)

public static final int **ACTION_POINTER_INDEX_MASK** Added in API level 8

Bits in the action code that represent a pointer index, used with ACTION_POINTER_DOWN (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN) and ACTION_POINTER_UP (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP). Shifting down by ACTION_POINTER_INDEX_SHIFT (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_SHIFT) provides the actual pointer index where the data for the pointer going up or down can be found; you can get its identifier with getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) and the actual data with getX(int) (/reference/android/view/MotionEvent.html#getX(int)) etc.

See Also
getActionIndex()

Constant Value: 65280 (0x0000ff00)

public static final int **ACTION_POINTER_INDEX_SHIFT** Added in API level 8

Bit shift for the action bits holding the pointer index as defined by ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK).

See Also
getActionIndex()

Constant Value: 8 (0x00000008)

public static final int **ACTION_POINTER_UP** Added in API level 5

Constant for getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()): A non-primary pointer has gone up.

Use getActionIndex() (/reference/android/view/MotionEvent.html#getActionIndex()) to retrieve the index of the pointer that changed.

The index is encoded in the ACTION_POINTER_INDEX_MASK (/reference/android/view/MotionEvent.html#ACTION_POINTER_INDEX_MASK) bits of the unmasked action returned by getAction() (/reference/android/view/MotionEvent.html#getAction()).

Constant Value: 6 (0x00000006)

public static final int **ACTION_SCROLL** Added in API level 12

Constant for getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()): The motion event contains relative vertical and/or horizontal scroll offsets. Use getAxisValue(int) (/reference/android/view/MotionEvent.html#getAxisValue(int)) to retrieve the information from AXIS_VSCROLL (/reference/android/view/MotionEvent.html#AXIS_VSCROLL) and AXIS_HSCROLL (/reference/android/view/MotionEvent.html#AXIS_HSCROLL). The pointer may or may not be down when this event is dispatched.

This action is always delivered to the window or view under the pointer, which may not be the window or view currently touched.

This action is not a touch event so it is delivered to onGenericMotionEvent(MotionEvent) (/reference/android/view/View.html#onGenericMotionEvent(android.view.MotionEvent)) rather than onTouchEvent(MotionEvent) (/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)).

Constant Value: 8 (0x00000008)

public static final int **ACTION_UP** Added in API level 1

Constant for getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()): A pressed gesture has finished, the motion contains the final release location as well as any intermediate points since the last down or move event.

Constant Value: 1 (0x00000001)

public static final int **AXIS_BRAKE** Added in API level 12

Axis constant: Brake axis of a motion event.

- For a joystick, reports the absolute position of the brake control. The value is normalized to a range from 0.0 (no braking) to 1.0 (maximum braking).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 23 (0x00000017)

public static final int **AXIS_DISTANCE** Added in API level 14

Axis constant: Distance axis of a motion event.

- For a stylus, reports the distance of the stylus from the screen. A value of 0.0 indicates direct contact and larger values indicate increasing distance from the surface.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 24 (0x00000018)

public static final int **AXIS_GAS** Added in API level 12

Axis constant: Gas axis of a motion event.

- For a joystick, reports the absolute position of the gas (accelerator) control. The value is normalized to a range from 0.0 (no acceleration) to 1.0 (maximum acceleration).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 22 (0x00000016)

public static final int **AXIS_GENERIC_1** Added in API level 12

Axis constant: Generic 1 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 32 (0x00000020)

public static final int **AXIS_GENERIC_10** Added in API level 12

Axis constant: Generic 10 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 41 (0x00000029)

public static final int **AXIS_GENERIC_11** Added in API level 12

Axis constant: Generic 11 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 42 (0x0000002a)

public static final int **AXIS_GENERIC_12** Added in API level 12

Axis constant: Generic 12 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 43 (0x0000002b)

public static final int **AXIS_GENERIC_13** Added in API level 12

Axis constant: Generic 13 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 44 (0x0000002c)

public static final int **AXIS_GENERIC_14** Added in API level 12

Axis constant: Generic 14 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 45 (0x0000002d)

public static final int **AXIS_GENERIC_15** Added in API level 12

Axis constant: Generic 15 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 46 (0x0000002e)

public static final int **AXIS_GENERIC_16** Added in API level 12

Axis constant: Generic 16 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 47 (0x0000002f)

public static final int **AXIS_GENERIC_2** Added in API level 12

Axis constant: Generic 2 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 33 (0x00000021)

public static final int **AXIS_GENERIC_3** Added in API level 12

Axis constant: Generic 3 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 34 (0x00000022)

public static final int **AXIS_GENERIC_4** Added in API level 12

Axis constant: Generic 4 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 35 (0x00000023)

public static final int **AXIS_GENERIC_5** Added in API level 12

Axis constant: Generic 5 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 36 (0x00000024)

public static final int **AXIS_GENERIC_6** Added in API level 12

Axis constant: Generic 6 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 37 (0x00000025)

public static final int **AXIS_GENERIC_7** Added in API level 12

Axis constant: Generic 7 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 38 (0x00000026)

public static final int **AXIS_GENERIC_8** Added in API level 12

Axis constant: Generic 8 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 39 (0x00000027)

public static final int **AXIS_GENERIC_9** Added in API level 12

Axis constant: Generic 9 axis of a motion event. The interpretation of a generic axis is device-specific.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 40 (0x00000028)

public static final int **AXIS_HAT_X** Added in API level 12

Axis constant: Hat X axis of a motion event.

- For a joystick, reports the absolute X position of the directional hat control. The value is normalized to a range from -1.0 (left) to 1.0 (right).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 15 (0x0000000f)

public static final int **AXIS_HAT_Y** Added in API level 12

Axis constant: Hat Y axis of a motion event.

- For a joystick, reports the absolute Y position of the directional hat control. The value is normalized to a range from -1.0 (up) to 1.0 (down).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 16 (0x00000010)

public static final int **AXIS_HSCROLL** Added in API level 12

Axis constant: Horizontal Scroll axis of a motion event.

- For a mouse, reports the relative movement of the horizontal scroll wheel. The value is normalized to a range from -1.0 (left) to 1.0 (right).

This axis should be used to scroll views horizontally.

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 10 (0x0000000a)

public static final int **AXIS_LTRIGGER** Added in API level 12

Axis constant: Left Trigger axis of a motion event.

- For a joystick, reports the absolute position of the left trigger control. The value is normalized to a range from 0.0 (released) to 1.0 (fully pressed).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 17 (0x00000011)

public static final int **AXIS_ORIENTATION** Added in API level 12

Axis constant: Orientation axis of a motion event.

- For a touch screen or touch pad, reports the orientation of the finger or tool in radians relative to the vertical plane of the device. An angle of 0 radians indicates that the major axis of contact is oriented upwards, is perfectly circular or is of unknown orientation. A positive angle indicates that the major axis of contact is oriented to the right. A negative angle indicates that the major axis of contact is oriented to the left. The full range is from -PI/2 radians (finger pointing fully left) to PI/2 radians (finger pointing fully right).
- For a stylus, the orientation indicates the direction in which the stylus is pointing in relation to the vertical axis of the current orientation of the screen. The range is from -PI radians to PI radians, where 0 is pointing up, -PI/2 radians is pointing left, -PI or PI radians is pointing down, and PI/2 radians is pointing right. See also AXIS_TILT.

See Also
getOrientation(int)
getHistoricalOrientation(int, int)
orientation
getMotionRange(int)

Constant Value: 8 (0x00000008)

public static final int **AXIS_PRESSURE** Added in API level 12

Axis constant: Pressure axis of a motion event.

- For a touch screen or touch pad, reports the approximate pressure applied to the surface by a finger or other tool. The value is normalized to a range from 0 (no pressure at all) to 1 (normal pressure), although values higher than 1 may be generated depending on the calibration of the input device.
- For a trackball, the value is set to 1 if the trackball button is pressed or 0 otherwise.
- For a mouse, the value is set to 1 if the primary mouse button is pressed or 0 otherwise.

See Also
getPressure(int)
getHistoricalPressure(int, int)
pressure
getMotionRange(int)

Constant Value: 2 (0x00000002)

public static final int **AXIS_RTRIGGER** Added in API level 12

Axis constant: Right Trigger axis of a motion event.

- For a joystick, reports the absolute position of the right trigger control. The value is normalized to a range from 0.0 (released) to 1.0 (fully pressed).

See Also
getAxisValue(int, int)
getHistoricalAxisValue(int, int, int)
getAxisValue(int)
getMotionRange(int)

Constant Value: 18 (0x00000012)

public static final int **AXIS_RUDDER** Added in API level 12

Axis constant: Rudder axis of a motion event.

- For a joystick, reports the absolute position of the rudder control. The value is normalized to a range from -1.0 (turn left) to 1.0 (turn right).

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 20 (0x00000014)

public static final int **AXIS_RX** Added in API level 12

Axis constant: X Rotation axis of a motion event.

- For a joystick, reports the absolute rotation angle about the X axis. The value is normalized to a range from -1.0 (counter-clockwise) to 1.0 (clockwise).

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 12 (0x0000000c)

public static final int **AXIS_RY** Added in API level 12

Axis constant: Y Rotation axis of a motion event.

- For a joystick, reports the absolute rotation angle about the Y axis. The value is normalized to a range from -1.0 (counter-clockwise) to 1.0 (clockwise).

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 13 (0x0000000d)

public static final int **AXIS_RZ** Added in API level 12

Axis constant: Z Rotation axis of a motion event.

- For a joystick, reports the absolute rotation angle about the Z axis. The value is normalized to a range from -1.0 (counter-clockwise) to 1.0 (clockwise). *On game pads with two analog joysticks, this axis is often reinterpreted to report the absolute Y position of the second joystick instead.*

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 14 (0x0000000e)

public static final int **AXIS_SIZE** Added in API level 12

Axis constant: Size axis of a motion event.

- For a touch screen or touch pad, reports the approximate size of the contact area in relation to the maximum detectable size for the device. The value is normalized to a range from 0 (smallest detectable size) to 1 (largest detectable size), although it is not a linear scale. This value is of limited use. To obtain calibrated size information, use `AXIS_TOUCH_MAJOR` or `AXIS_TOOL_MAJOR`.

See Also
`getSize(int)`
`getHistoricalSize(int, int)`
`size`
`getMotionRange(int)`

Constant Value: 3 (0x00000003)

public static final int **AXIS_THROTTLE** Added in API level 12

Axis constant: Throttle axis of a motion event.

- For a joystick, reports the absolute position of the throttle control. The value is normalized to a range from 0.0 (fully open) to 1.0 (fully closed).

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 19 (0x00000013)

public static final int **AXIS_TILT** Added in API level 14

Axis constant: Tilt axis of a motion event.

- For a stylus, reports the tilt angle of the stylus in radians where 0 radians indicates that the stylus is being held perpendicular to the surface, and PI/2 radians indicates that the stylus is being held flat against the surface.

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int, int)`
`getMotionRange(int)`

Constant Value: 25 (0x00000019)

public static final int **AXIS_TOOL_MAJOR** Added in API level 12

Axis constant: ToolMajor axis of a motion event.

- For a touch screen, reports the length of the major axis of an ellipse that represents the size of the approaching finger or tool used to make contact.
- For a touch pad, reports the length of the major axis of an ellipse that represents the size of the approaching finger or tool used to make contact. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.

When the touch is circular, the major and minor axis lengths will be equal to one another.

The tool size may be larger than the touch size since the tool may not be fully in contact with the touch sensor.

See Also
`getToolMajor(int)`
`getHistoricalToolMajor(int, int)`
`toolMajor`
`getMotionRange(int)`

Constant Value: 6 (0x00000006)

public static final int **AXIS_TOOL_MINOR** Added in API level 12

Axis constant: ToolMinor axis of a motion event.

- For a touch screen, reports the length of the minor axis of an ellipse that represents the size of the approaching finger or tool used to make contact.
- For a touch pad, reports the length of the minor axis of an ellipse that represents the size of the approaching finger or tool used to make contact. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.

When the touch is circular, the major and minor axis lengths will be equal to one another.

The tool size may be larger than the touch size since the tool may not be fully in contact with the touch sensor.

See Also
`getToolMinor(int)`
`getHistoricalToolMinor(int, int)`
`toolMinor`
`getMotionRange(int)`

Constant Value: 7 (0x00000007)

public static final int **AXIS_TOUCH_MAJOR** Added in API level 12

Axis constant: TouchMajor axis of a motion event.

- For a touch screen, reports the length of the major axis of an ellipse that represents the touch area at the point of contact. The units are display pixels.
- For a touch pad, reports the length of the major axis of an ellipse that represents the touch area at the point of contact. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.

See Also
`getTouchMajor(int)`
`getHistoricalTouchMajor(int, int)`
`touchMajor`
`getMotionRange(int)`

Constant Value: 4 (0x00000004)

public static final int **AXIS_TOUCH_MINOR** Added in API level 12

Axis constant: TouchMinor axis of a motion event.

- For a touch screen, reports the length of the minor axis of an ellipse that represents the touch area at the point of contact. The units are display pixels.
- For a touch pad, reports the length of the minor axis of an ellipse that represents the touch area at the point of contact. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.

When the touch is circular, the major and minor axis lengths will be equal to one another.

See Also
`getTouchMinor(int)`
`getHistoricalTouchMinor(int, int)`
`touchMinor`
`getMotionRange(int)`

Constant Value: 5 (0x00000005)

public static final int **AXIS_VSCROLL** Added in API level 12

Axis constant: Vertical Scroll axis of a motion event.

- For a mouse, reports the relative movement of the vertical scroll wheel. The value is normalized to a range from -1.0 (down) to 1.0 (up).

This axis should be used to scroll views vertically.

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 9 (0x00000009)

public static final int **AXIS_WHEEL** Added in API level 12

Axis constant: Wheel axis of a motion event.

- For a joystick, reports the absolute position of the steering wheel control. The value is normalized to a range from -1.0 (turn left) to 1.0 (turn right).

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 21 (0x00000015)

public static final int **AXIS_X** Added in API level 12

Axis constant: X axis of a motion event.

- For a touch screen, reports the absolute X screen position of the center of the touch contact area. The units are display pixels.
- For a touch pad, reports the absolute X surface position of the center of the touch contact area. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.
- For a mouse, reports the absolute X screen position of the mouse pointer. The units are display pixels.
- For a trackball, reports the relative horizontal displacement of the trackball. The value is normalized to a range from -1.0 (left) to 1.0 (right).
- For a joystick, reports the absolute X position of the joystick. The value is normalized to a range from -1.0 (left) to 1.0 (right).

See Also
`getX(int)`
`getHistoricalX(int, int)`
`x`
`getMotionRange(int)`

Constant Value: 0 (0x00000000)

public static final int **AXIS_Y** Added in API level 12

Axis constant: Y axis of a motion event.

- For a touch screen, reports the absolute Y screen position of the center of the touch contact area. The units are display pixels.
- For a touch pad, reports the absolute Y surface position of the center of the touch contact area. The units are device-dependent; use `getMotionRange(int)` to query the effective range of values.
- For a mouse, reports the absolute Y screen position of the mouse pointer. The units are display pixels.
- For a trackball, reports the relative vertical displacement of the trackball. The value is normalized to a range from -1.0 (up) to 1.0 (down).
- For a joystick, reports the absolute Y position of the joystick. The value is normalized to a range from -1.0 (up or far) to 1.0 (down or near).

See Also
`getY(int)`
`getHistoricalY(int, int)`
`y`
`getMotionRange(int)`

Constant Value: 1 (0x00000001)

public static final int **AXIS_Z** Added in API level 12

Axis constant: Z axis of a motion event.

- For a joystick, reports the absolute Z position of the joystick. The value is normalized to a range from -1.0 (high) to 1.0 (low). *On game pads with two analog joysticks, this axis is often reinterpreted to report the absolute X position of the second joystick instead.*

See Also
`getAxisValue(int, int)`
`getHistoricalAxisValue(int, int, int)`
`getAxisValue(int)`
`getMotionRange(int)`

Constant Value: 11 (0x0000000b)

public static final int **BUTTON_BACK** Added in API level 14

Button constant: Back button pressed (mouse back button).

The system may send a `KEYCODE_BACK` (/reference/android/view/KeyEvent.html#KEYCODE_BACK) key press to the application when this button is pressed.

See Also
`getButtonState()`

Constant Value: 8 (0x00000008)

public static final int **BUTTON_FORWARD** Added in API level 14

Button constant: Forward button pressed (mouse forward button).

The system may send a `KEYCODE_FORWARD` (/reference/android/view/KeyEvent.html#KEYCODE_FORWARD) key press to the application when this button is pressed.

See Also
`getButtonState()`

Constant Value: 16 (0x00000010)

public static final int **BUTTON_PRIMARY** Added in API level 14

Button constant: Primary button (left mouse button). This button constant is not set in response to simple touches with a finger or stylus tip. The user must actually push a button.

See Also
`getButtonState()`

Constant Value: 1 (0x00000001)

public static final int **BUTTON_SECONDARY**

Button constant: Secondary button (right mouse button, stylus first button).

See Also
`getButtonState()`

Constant Value: 2 (0x00000002)

public static final int **BUTTON_TERTIARY** Added in API level 14

Button constant: Tertiary button (middle mouse button, stylus second button).

See Also
`getButtonState()`

Constant Value: 4 (0x00000004)

public static final int **EDGE_BOTTOM** Added in API level 1

Flag indicating the motion event intersected the bottom edge of the screen.

Constant Value: 2 (0x00000002)

public static final int **EDGE_LEFT** Added in API level 1

Flag indicating the motion event intersected the left edge of the screen.

Constant Value: 4 (0x00000004)

public static final int **EDGE_RIGHT** Added in API level 1

Flag indicating the motion event intersected the right edge of the screen.

Constant Value: 8 (0x00000008)

public static final int **EDGE_TOP** Added in API level 1

Flag indicating the motion event intersected the top edge of the screen.

Constant Value: 1 (0x00000001)

public static final int **FLAG_WINDOW_IS_OBSCURED** Added in API level 9

This flag indicates that the window that received this motion event is partly or wholly obscured by another visible window above it. This flag is set to true even if the event did not directly pass through the obscured area. A security sensitive application can check this flag to identify situations in which a malicious application may have covered up part of its content for the purpose of misleading the user or hijacking touches. An appropriate response might be to drop the suspect touches or to take additional precautions to confirm the user's actual intent.

Constant Value: 1 (0x00000001)

public static final int **INVALID_POINTER_ID** Added in API level 14

An invalid pointer id. This value (-1) can be used as a placeholder to indicate that a pointer id has not been assigned or is not available. It cannot appear as a pointer id inside a `MotionEvent` (/reference/android/view/MotionEvent.html).

Constant Value: -1 (0xffffffff)

public static final int **TOOL_TYPE_ERASER** Added in API level 14

Tool type constant: The tool is an eraser or a stylus being used in an inverted posture.

See Also
`getToolType(int)`

Constant Value: 4 (0x00000004)

public static final int **TOOL_TYPE_FINGER** Added in API level 14

Tool type constant: The tool is a finger.

See Also
`getToolType(int)`

Constant Value: 1 (0x00000001)

public static final int **TOOL_TYPE_MOUSE** Added in API level 14

Tool type constant: The tool is a mouse or trackpad.

See Also
`getToolType(int)`

Constant Value: 3 (0x00000003)

public static final int **TOOL_TYPE_STYLUS** Added in API level 14

Tool type constant: The tool is a stylus.

See Also
`getToolType(int)`

Constant Value: 2 (0x00000002)

public static final int **TOOL_TYPE_UNKNOWN** Added in API level 14

Tool type constant: Unknown tool type. This constant is used when the tool type is not known or is not relevant, such as for a trackball or other non-pointing device.

See Also
`getToolType(int)`

Constant Value: 0 (0x00000000)

Fields

public static final Creator<MotionEvent> **CREATOR** Added in API level 1

Public Methods

public static String **actionToString** (int action) Added in API level 19

Returns a string that represents the symbolic name of the specified unmasked action such as "ACTION_DOWN", "ACTION_POINTER_DOWN(3)" or an equivalent numeric constant such as "35" if unknown.

Parameters
action The unmasked action.

Returns
The symbolic name of the specified action.

See Also
`getAction()`

public final void **addBatch** (long eventTime, PointerCoords[] pointerCoords, int metaState) Added in API level 9

Add a new movement to the batch of movements in this event. The event's current location, position and size is updated to the new values. The current values in the event are added to a list of historical values.
Only applies to `ACTION_MOVE` (/reference/android/view/MotionEvent.html#ACTION_MOVE) or `ACTION_HOVER_MOVE` (/reference/android/view/MotionEvent.html#ACTION_HOVER_MOVE) events.

Parameters
eventTime The time stamp (in ms) for this data.
pointerCoords The new pointer coordinates.
metaState Meta key state.

public final void **addBatch** (long eventTime, float x, float y, float pressure, float size, int metaState) Added in API level 1

Add a new movement to the batch of movements in this event. The event's current location, position and size is updated to the new values. The current values in the event are added to a list of historical values.
Only applies to `ACTION_MOVE` (/reference/android/view/MotionEvent.html#ACTION_MOVE) or `ACTION_HOVER_MOVE` (/reference/android/view/MotionEvent.html#ACTION_HOVER_MOVE) events.

Parameters
eventTime The time stamp (in ms) for this data.
x The new X position.
y The new Y position.

pressure The new pressure.
size The new size.
metaState Meta key state.

public static int **axisFromString** (String symbolicName) Added in API level 12

Gets an axis by its symbolic name such as "AXIS_X" or an equivalent numeric constant such as "42".

Parameters
symbolicName The symbolic name of the axis.

Returns
The axis or -1 if not found.

See Also
keyCodeToString(int)

public static String **axisToString** (int axis) Added in API level 12

Returns a string that represents the symbolic name of the specified axis such as "AXIS_X" or an equivalent numeric constant such as "42" if unknown.

Parameters
axis The axis.

Returns
The symbolic name of the specified axis.

public final int **findPointerIndex** (int pointerId) Added in API level 5

Given a pointer identifier, find the index of its data in the event.

Parameters
pointerId The identifier of the pointer to be found.

Returns
Returns either the index of the pointer (for use with getX(int) et al.), or -1 if there is no data available for that pointer identifier.

public final int **getAction** () Added in API level 1

Return the kind of action being performed. Consider using getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()) and getActionIndex() (/reference/android/view/MotionEvent.html#getActionIndex()) to retrieve the separate masked action and pointer index.

Returns
The action, such as ACTION_DOWN or the combination of ACTION_POINTER_DOWN with a shifted pointer index.

public final int **getActionIndex** () Added in API level 8

For ACTION_POINTER_DOWN (/reference/android/view/MotionEvent.html#ACTION_POINTER_DOWN) or ACTION_POINTER_UP (/reference/android/view/MotionEvent.html#ACTION_POINTER_UP) as returned by getActionMasked() (/reference/android/view/MotionEvent.html#getActionMasked()), this returns the associated pointer index. The index may be used with getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)), getX(int) (/reference/android/view/MotionEvent.html#getX(int)), getY(int) (/reference/android/view/MotionEvent.html#getY(int)), getPressure(int) (/reference/android/view/MotionEvent.html#getPressure(int)), and getSize(int) (/reference/android/view/MotionEvent.html#getSize(int)) to get information about the pointer that has gone down or up.

Returns
The index associated with the action.

public final int **getActionMasked** () Added in API level 8

Return the masked action being performed, without pointer index information. Use getActionIndex() (/reference/android/view/MotionEvent.html#getActionIndex()) to return the index associated with pointer actions.

Returns
The action, such as ACTION_DOWN or ACTION_POINTER_DOWN.

public final float **getAxisValue** (int axis) Added in API level 12

getAxisValue(int) (/reference/android/view/MotionEvent.html#getAxisValue(int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters
axis The axis identifier for the axis value to retrieve.

See Also
AXIS_X
AXIS_Y

public final float **getAxisValue** (int axis, int pointerIndex) Added in API level 12

Returns the value of the requested axis for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index).

Parameters
axis The axis identifier for the axis value to retrieve.
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

Returns
The value of the axis, or 0 if the axis is not available.

See Also
AXIS_X
AXIS_Y

public final int **getButtonState** () Added in API level 14

Gets the state of all buttons that are pressed such as a mouse or stylus button.

Returns
The button state.

See Also
BUTTON_PRIMARY
BUTTON_SECONDARY
BUTTON_TERTIARY
BUTTON_FORWARD
BUTTON_BACK

public final int **getDeviceId** () Added in API level 1

Gets the id for the device that this event came from. An id of zero indicates that the event didn't come from a physical device and maps to the default keymap. The other numbers are arbitrary and you shouldn't depend on the values.

Returns
The device id.

public final long **getDownTime** () Added in API level 1

Returns the time (in ms) when the user originally pressed down to start a stream of position events.

public final int **getEdgeFlags** () Added in API level 1

Returns a bitfield indicating which edges, if any, were touched by this MotionEvent. For touch events, clients can use this to determine if the user's finger was touching the edge of the display. This property is only set for ACTION_DOWN (/reference/android/view/MotionEvent.html#ACTION_DOWN) events.

See Also
EDGE_LEFT
EDGE_TOP
EDGE_RIGHT
EDGE_BOTTOM

public final long **getEventTime** () Added in API level 1

Retrieve the time this event occurred, in the uptimeMillis() (/reference/android/os/SystemClock.html#uptimeMillis()) time base.

Returns
Returns the time this event occurred, in the uptimeMillis() time base.

public final int **getFlags** () Added in API level 9

Gets the motion event flags.

See Also
FLAG_WINDOW_IS_OBSCURED

public final float **getHistoricalAxisValue** (int axis, int pointerIndex, int pos) Added in API level 12

Returns the historical value of the requested axis, as per `getAxisValue(int, int)` (/reference/android/view/MotionEvent.html#getAxisValue(int, int)), occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

axis	The axis identifier for the axis value to retrieve.
pointerIndex	Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.
pos	Which historical value to return; must be less than `getHistorySize()`

Returns
The value of the axis, or 0 if the axis is not available.
See Also
AXIS_X
AXIS_Y

public final float **getHistoricalAxisValue** (int axis, int pos) Added in API level 12

`getHistoricalAxisValue(int, int, int)` (/reference/android/view/MotionEvent.html#getHistoricalAxisValue(int, int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

axis	The axis identifier for the axis value to retrieve.
pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getAxisValue(int)
AXIS_X
AXIS_Y

public final long **getHistoricalEventTime** (int pos) Added in API level 1

Returns the time that a historical movement occurred between this event and the previous event, in the `uptimeMillis()` (/reference/android/os/SystemClock.html#uptimeMillis()) time base.

This only applies to ACTION_MOVE events.

Parameters

pos	Which historical value to return; must be less than `getHistorySize()`

Returns
Returns the time that a historical movement occurred between this event and the previous event, in the `uptimeMillis()` time base.
See Also
getHistorySize()
getEventTime()

public final float **getHistoricalOrientation** (int pointerIndex, int pos) Added in API level 9

Returns a historical orientation coordinate, as per `getOrientation(int)` (/reference/android/view/MotionEvent.html#getOrientation(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex	Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.
pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getOrientation(int)
AXIS_ORIENTATION

public final float **getHistoricalOrientation** (int pos) Added in API level 9

`getHistoricalOrientation(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalOrientation(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getOrientation()
AXIS_ORIENTATION

public final void **getHistoricalPointerCoords** (int pointerIndex, int pos, MotionEvent.PointerCoords outPointerCoords) Added in API level 9

Populates a `MotionEvent.PointerCoords` (/reference/android/view/MotionEvent.PointerCoords.html) object with historical pointer coordinate data, as per `getPointerCoords(int, MotionEvent.PointerCoords)` (/reference/android/view/MotionEvent.html#getPointerCoords(int, android.view.MotionEvent.PointerCoords)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex	Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.
pos	Which historical value to return; must be less than `getHistorySize()`
outPointerCoords	The pointer coordinate object to populate.

See Also
getHistorySize()
getPointerCoords(int, MotionEvent.PointerCoords)
MotionEvent.PointerCoords

public final float **getHistoricalPressure** (int pos) Added in API level 1

`getHistoricalPressure(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalPressure(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getPressure()
AXIS_PRESSURE

public final float **getHistoricalPressure** (int pointerIndex, int pos) Added in API level 5

Returns a historical pressure coordinate, as per `getPressure(int)` (/reference/android/view/MotionEvent.html#getPressure(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex	Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.
pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getPressure(int)
AXIS_PRESSURE

public final float **getHistoricalSize** (int pos) Added in API level 1

`getHistoricalSize(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalSize(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos	Which historical value to return; must be less than `getHistorySize()`

See Also
getHistorySize()
getSize()
AXIS_SIZE

public final float **getHistoricalSize** (int pointerIndex, int pos) Added in API level 5

Returns a historical size coordinate, as per `getSize(int)` (/reference/android/view/MotionEvent.html#getSize(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getSize(int)`
`AXIS_SIZE`

public final float **getHistoricalToolMajor** (int pointerIndex, int pos) Added in API level 9

Returns a historical tool major axis coordinate, as per `getToolMajor(int)` (/reference/android/view/MotionEvent.html#getToolMajor(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getToolMajor(int)`
`AXIS_TOOL_MAJOR`

public final float **getHistoricalToolMajor** (int pos) Added in API level 9

`getHistoricalToolMajor(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalToolMajor(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getToolMajor()`
`AXIS_TOOL_MAJOR`

public final float **getHistoricalToolMinor** (int pointerIndex, int pos) Added in API level 9

Returns a historical tool minor axis coordinate, as per `getToolMinor(int)` (/reference/android/view/MotionEvent.html#getToolMinor(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getToolMinor(int)`
`AXIS_TOOL_MINOR`

public final float **getHistoricalToolMinor** (int pos) Added in API level 9

`getHistoricalToolMinor(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalToolMinor(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getToolMinor()`
`AXIS_TOOL_MINOR`

public final float **getHistoricalTouchMajor** (int pointerIndex, int pos) Added in API level 9

Returns a historical touch major axis coordinate, as per `getTouchMajor(int)` (/reference/android/view/MotionEvent.html#getTouchMajor(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getTouchMajor(int)`
`AXIS_TOUCH_MAJOR`

public final float **getHistoricalTouchMajor** (int pos) Added in API level 9

`getHistoricalTouchMajor(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalTouchMajor(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getTouchMajor()`
`AXIS_TOUCH_MAJOR`

public final float **getHistoricalTouchMinor** (int pointerIndex, int pos) Added in API level 9

Returns a historical touch minor axis coordinate, as per `getTouchMinor(int)` (/reference/android/view/MotionEvent.html#getTouchMinor(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getTouchMinor(int)`
`AXIS_TOUCH_MINOR`

public final float **getHistoricalTouchMinor** (int pos) Added in API level 9

`getHistoricalTouchMinor(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalTouchMinor(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getTouchMinor()`
`AXIS_TOUCH_MINOR`

public final float **getHistoricalX** (int pos) Added in API level 1

`getHistoricalX(int, int)` (/reference/android/view/MotionEvent.html#getHistoricalX(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters

pos Which historical value to return; must be less than `getHistorySize()`

See Also

`getHistorySize()`
`getX()`
`AXIS_X`

public final float **getHistoricalX** (int pointerIndex, int pos) Added in API level 5

Returns a historical X coordinate, as per `getX(int)` (/reference/android/view/MotionEvent.html#getX(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters

pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

pos Which historical value to return; must be less than getHistorySize()

See Also
getHistorySize()
getX(int)
AXIS_X

public final float **getHistoricalY** (int pos) Added in API level 1

getHistoricalY(int, int) (/reference/android/view/MotionEvent.html#getHistoricalY(int, int)) for the first pointer index (may be an arbitrary pointer identifier).

Parameters
pos Which historical value to return; must be less than getHistorySize()

See Also
getHistorySize()
getY()
AXIS_Y

public final float **getHistoricalY** (int pointerIndex, int pos) Added in API level 5

Returns a historical Y coordinate, as per getY(int) (/reference/android/view/MotionEvent.html#getY(int)), that occurred between this event and the previous event for the given pointer. Only applies to ACTION_MOVE events.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.
pos Which historical value to return; must be less than getHistorySize()

See Also
getHistorySize()
getY(int)
AXIS_Y

public final int **getHistorySize** () Added in API level 1

Returns the number of historical points in this event. These are movements that have occurred between this event and the previous event. This only applies to ACTION_MOVE events -- all other actions will have a size of 0.

Returns
Returns the number of historical points in the event.

public final int **getMetaState** () Added in API level 1

Returns the state of any meta / modifier keys that were in effect when the event was generated. This is the same values as those returned by KeyEvent.getMetaState (/reference/android/view/KeyEvent.html#getMetaState()).

Returns
an integer in which each bit set to 1 represents a pressed meta key

See Also
getMetaState()

public final float **getOrientation** (int pointerIndex) Added in API level 9

Returns the orientation of the touch area and tool area in radians clockwise from vertical for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). An angle of 0 radians indicates that the major axis of contact is oriented upwards, is perfectly circular or is of unknown orientation. A positive angle indicates that the major axis of contact is oriented to the right. A negative angle indicates that the major axis of contact is oriented to the left. The full range is from -PI/2 radians (finger pointing fully left) to PI/2 radians (finger pointing fully right).

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_ORIENTATION

public final float **getOrientation** () Added in API level 9

getOrientation(int) (/reference/android/view/MotionEvent.html#getOrientation(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_ORIENTATION

public final void **getPointerCoords** (int pointerIndex, MotionEvent.PointerCoords outPointerCoords) Added in API level 9

Populates a MotionEvent.PointerCoords (/reference/android/view/MotionEvent.PointerCoords.html) object with pointer coordinate data for the specified pointer index.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.
outPointerCoords The pointer coordinate object to populate.

See Also
MotionEvent.PointerCoords

public final int **getPointerCount** () Added in API level 5

The number of pointers of data contained in this event. Always >= 1.

public final int **getPointerId** (int pointerIndex) Added in API level 5

Return the pointer identifier associated with a particular pointer data index is this event. The identifier tells you the actual pointer number associated with the data, accounting for individual pointers going up and down since the start of the current gesture.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

public final void **getPointerProperties** (int pointerIndex, MotionEvent.PointerProperties outPointerProperties) Added in API level 14

Populates a MotionEvent.PointerProperties (/reference/android/view/MotionEvent.PointerProperties.html) object with pointer properties for the specified pointer index.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.
outPointerProperties The pointer properties object to populate.

See Also
MotionEvent.PointerProperties

public final float **getPressure** () Added in API level 1

getPressure(int) (/reference/android/view/MotionEvent.html#getPressure(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_PRESSURE

public final float **getPressure** (int pointerIndex) Added in API level 5

Returns the current pressure of this event for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). The pressure generally ranges from 0 (no pressure at all) to 1 (normal pressure), however values higher than 1 may be generated depending on the calibration of the input device.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_PRESSURE

public final float **getRawX** () Added in API level 1

Returns the original raw X coordinate of this event. For touch events on the screen, this is the original location of the event on the screen, before it had been adjusted for the containing window and views.

See Also
getX(int)
AXIS_X

public final float **getRawY** () Added in API level 1

Returns the original raw Y coordinate of this event. For touch events on the screen, this is the original location of the event on the screen, before it had been adjusted for the containing window and views.

See Also
getY(int)
AXIS_Y

public final float **getSize** (int pointerIndex) Added in API level 5

Returns a scaled value of the approximate size for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). This represents some approximation of the area of the screen being pressed; the actual value in pixels corresponding to the touch is normalized with the device specific range of values and scaled to a value between 0 and 1. The value of size can be used to determine fat touch events.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_SIZE

public final float **getSize** () Added in API level 1

getSize(int) (/reference/android/view/MotionEvent.html#getSize(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_SIZE

public final int **getSource** () Added in API level 9

Gets the source of the event.

Returns
The event source or SOURCE_UNKNOWN if unknown.

public final float **getToolMajor** (int pointerIndex) Added in API level 9

Returns the length of the major axis of an ellipse that describes the size of the approaching tool for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). The tool area represents the estimated size of the finger or pen that is touching the device independent of its actual touch area at the point of contact.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_TOOL_MAJOR

public final float **getToolMajor** () Added in API level 9

getToolMajor(int) (/reference/android/view/MotionEvent.html#getToolMajor(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_TOOL_MAJOR

public final float **getToolMinor** () Added in API level 9

getToolMinor(int) (/reference/android/view/MotionEvent.html#getToolMinor(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_TOOL_MINOR

public final float **getToolMinor** (int pointerIndex) Added in API level 9

Returns the length of the minor axis of an ellipse that describes the size of the approaching tool for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). The tool area represents the estimated size of the finger or pen that is touching the device independent of its actual touch area at the point of contact.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_TOOL_MINOR

public final int **getToolType** (int pointerIndex) Added in API level 14

Gets the tool type of a pointer for the given pointer index. The tool type indicates the type of tool used to make contact such as a finger or stylus, if known.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

Returns
The tool type of the pointer.

See Also
TOOL_TYPE_UNKNOWN
TOOL_TYPE_FINGER
TOOL_TYPE_STYLUS
TOOL_TYPE_MOUSE

public final float **getTouchMajor** () Added in API level 9

getTouchMajor(int) (/reference/android/view/MotionEvent.html#getTouchMajor(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_TOUCH_MAJOR

public final float **getTouchMajor** (int pointerIndex) Added in API level 9

Returns the length of the major axis of an ellipse that describes the touch area at the point of contact for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index).

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_TOUCH_MAJOR

public final float **getTouchMinor** () Added in API level 9

getTouchMinor(int) (/reference/android/view/MotionEvent.html#getTouchMinor(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_TOUCH_MINOR

public final float **getTouchMinor** (int pointerIndex) Added in API level 9

Returns the length of the minor axis of an ellipse that describes the touch area at the point of contact for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index).

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_TOUCH_MINOR

public final float **getX** (int pointerIndex) Added in API level 5

Returns the X coordinate of this event for the given pointer *index* (use getPointerId(int) (/reference/android/view/MotionEvent.html#getPointerId(int)) to find the pointer identifier for this index). Whole numbers are pixels; the value may have a fraction for input devices that are sub-pixel precise.

Parameters
pointerIndex Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to getPointerCount()-1.

See Also
AXIS_X

public final float **getX** () Added in API level 1

getX(int) (/reference/android/view/MotionEvent.html#getX(int)) for the first pointer index (may be an arbitrary pointer identifier).

See Also

AXIS_X

public final float **getXPrecision** ()

Added in API level 1

Return the precision of the X coordinates being reported. You can multiply this number with `getX()` to find the actual hardware value of the X coordinate.

Returns
Returns the precision of X coordinates being reported.
See Also
AXIS_X

public final float **getY** (int pointerIndex)

Added in API level 5

Returns the Y coordinate of this event for the given pointer *index* (use `getPointerId(int)` to find the pointer identifier for this index). Whole numbers are pixels; the value may have a fraction for input devices that are sub-pixel precise.

Parameters

pointerIndex	Raw index of pointer to retrieve. Value may be from 0 (the first pointer that is down) to `getPointerCount()`-1.

See Also
AXIS_Y

public final float **getY** ()

Added in API level 1

`getY(int)` for the first pointer index (may be an arbitrary pointer identifier).

See Also
AXIS_Y

public final float **getYPrecision** ()

Added in API level 1

Return the precision of the Y coordinates being reported. You can multiply this number with `getY()` to find the actual hardware value of the Y coordinate.

Returns
Returns the precision of Y coordinates being reported.
See Also
AXIS_Y

public static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, PointerProperties[] pointerProperties, PointerCoords[] pointerCoords, int metaState, int buttonState, float xPrecision, float yPrecision, int deviceId, int edgeFlags, int source, int flags)

Added in API level 14

Create a new MotionEvent, filling in all of the basic values that define the motion.

Parameters

downTime	The time (in ms) when the user originally pressed down to start a stream of position events. This must be obtained from `uptimeMillis()`.
eventTime	The the time (in ms) when this specific event was generated. This must be obtained from `uptimeMillis()`.
action	The kind of action being performed, such as `ACTION_DOWN`.
pointerCount	The number of pointers that will be in this event.
pointerProperties	An array of *pointerCount* values providing a `MotionEvent.PointerProperties` property object for each pointer, which must include the pointer identifier.
pointerCoords	An array of *pointerCount* values providing a `MotionEvent.PointerCoords` coordinate object for each pointer.
metaState	The state of any meta / modifier keys that were in effect when the event was generated.
buttonState	The state of buttons that are pressed.
xPrecision	The precision of the X coordinate being reported.
yPrecision	The precision of the Y coordinate being reported.
deviceId	The id for the device that this event came from. An id of zero indicates that the event didn't come from a physical device; other numbers are arbitrary and you shouldn't depend on the values.
edgeFlags	A bitfield indicating which edges, if any, were touched by this MotionEvent.
source	The source of this event.
flags	The motion event flags.

public static MotionEvent **obtain** (long downTime, long eventTime, int action, float x, float y, float pressure, float size, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags)

Added in API level 1

Create a new MotionEvent, filling in all of the basic values that define the motion.

Parameters

downTime	The time (in ms) when the user originally pressed down to start a stream of position events. This must be obtained from `uptimeMillis()`.
eventTime	The the time (in ms) when this specific event was generated. This must be obtained from `uptimeMillis()`.
action	The kind of action being performed, such as `ACTION_DOWN`.
x	The X coordinate of this event.
y	The Y coordinate of this event.
pressure	The current pressure of this event. The pressure generally ranges from 0 (no pressure at all) to 1 (normal pressure), however values higher than 1 may be generated depending on the calibration of the input device.
size	A scaled value of the approximate size of the area being pressed when touched with the finger. The actual value in pixels corresponding to the finger touch is normalized with a device specific range of values and scaled to a value between 0 and 1.
metaState	The state of any meta / modifier keys that were in effect when the event was generated.
xPrecision	The precision of the X coordinate being reported.
yPrecision	The precision of the Y coordinate being reported.
deviceId	The id for the device that this event came from. An id of zero indicates that the event didn't come from a physical device; other numbers are arbitrary and you shouldn't depend on the values.
edgeFlags	A bitfield indicating which edges, if any, were touched by this MotionEvent.

public static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, float x, float y, float pressure, float size, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags)

Added in API level 5

This method was deprecated in API level 9.
Use `obtain(long, long, int, float, float, float, int, float, float, int, int)` instead.

Create a new MotionEvent, filling in all of the basic values that define the motion.

Parameters

downTime	The time (in ms) when the user originally pressed down to start a stream of position events. This must be obtained from `uptimeMillis()`.
eventTime	The the time (in ms) when this specific event was generated. This must be obtained from `uptimeMillis()`.
action	The kind of action being performed, such as `ACTION_DOWN`.
pointerCount	The number of pointers that are active in this event.
x	The X coordinate of this event.
y	The Y coordinate of this event.
pressure	The current pressure of this event. The pressure generally ranges from 0 (no pressure at all) to 1 (normal pressure), however values higher than 1 may be generated depending on the calibration of the input device.
size	A scaled value of the approximate size of the area being pressed when touched with the finger. The actual value in pixels corresponding to the finger touch is normalized with a device specific range of values and scaled to a value between 0 and 1.
metaState	The state of any meta / modifier keys that were in effect when the event was generated.
xPrecision	The precision of the X coordinate being reported.
yPrecision	The precision of the Y coordinate being reported.
deviceId	The id for the device that this event came from. An id of zero indicates that the event didn't come from a physical device; other numbers are arbitrary and you shouldn't depend on the values.
edgeFlags	A bitfield indicating which edges, if any, were touched by this MotionEvent.

public static MotionEvent **obtain** (long downTime, long eventTime, int action, int pointerCount, int[] pointerIds, PointerCoords[] pointerCoords, int metaState, float xPrecision, float yPrecision, int deviceId, int edgeFlags, int source, int flags)

Added in API level 9

This method was deprecated in API level 14.
Use `obtain(long, long, int, int, PointerProperties[], PointerCoords[], int, int, float, float, int, int, int, int)` instead.

Create a new MotionEvent, filling in all of the basic values that define the motion.

Parameters

downTime	The time (in ms) when the user originally pressed down to start a stream of position events. This must be obtained from `uptimeMillis()`.
eventTime	The the time (in ms) when this specific event was generated. This must be obtained from `uptimeMillis()`.
action	The kind of action being performed, such as `ACTION_DOWN`.
pointerCount	The number of pointers that will be in this event.

pointerIds	An array of *pointerCount* values providing an identifier for each pointer.
pointerCoords	An array of *pointerCount* values providing a `MotionEvent.PointerCoords` coordinate object for each pointer.
metaState	The state of any meta / modifier keys that were in effect when the event was generated.
xPrecision	The precision of the X coordinate being reported.
yPrecision	The precision of the Y coordinate being reported.
deviceId	The id for the device that this event came from. An id of zero indicates that the event didn't come from a physical device; other numbers are arbitrary and you shouldn't depend on the values.
edgeFlags	A bitfield indicating which edges, if any, were touched by this MotionEvent.
source	The source of this event.
flags	The motion event flags.

public static MotionEvent **obtain** (MotionEvent other) Added in API level 1

Create a new MotionEvent, copying from an existing one.

public static MotionEvent **obtain** (long downTime, long eventTime, int action, float x, float y, int metaState) Added in API level 1

Create a new MotionEvent, filling in a subset of the basic motion values. Those not specified here are: device id (always 0), pressure and size (always 1), x and y precision (always 1), and edgeFlags (always 0).

Parameters

downTime	The time (in ms) when the user originally pressed down to start a stream of position events. This must be obtained from `uptimeMillis()`.
eventTime	The the time (in ms) when this specific event was generated. This must be obtained from `uptimeMillis()`.
action	The kind of action being performed, such as `ACTION_DOWN`.
x	The X coordinate of this event.
y	The Y coordinate of this event.
metaState	The state of any meta / modifier keys that were in effect when the event was generated.

public static MotionEvent **obtainNoHistory** (MotionEvent other) Added in API level 5

Create a new MotionEvent, copying from an existing one, but not including any historical point information.

public final void **offsetLocation** (float deltaX, float deltaY) Added in API level 1

Adjust this event's location.

Parameters

deltaX	Amount to add to the current X coordinate of the event.
deltaY	Amount to add to the current Y coordinate of the event.

public final void **recycle** () Added in API level 1

Recycle the MotionEvent, to be re-used by a later caller. After calling this function you must not ever touch the event again.

public final void **setAction** (int action) Added in API level 1

Sets this event's action.

public final void **setEdgeFlags** (int flags) Added in API level 1

Sets the bitfield indicating which edges, if any, were touched by this MotionEvent.

See Also
`getEdgeFlags()`

public final void **setLocation** (float x, float y) Added in API level 1

Set this event's location. Applies `offsetLocation(float, float)` (/reference/android/view/MotionEvent.html#offsetLocation(float, float)) with a delta from the current location to the given new location.

Parameters

x	New absolute X location.
y	New absolute Y location.

public final void **setSource** (int source) Added in API level 12

Modifies the source of the event.

Parameters

source	The new source.

public String **toString** () Added in API level 1

Returns a string containing a concise, human-readable description of this object. Subclasses are encouraged to override this method and provide an implementation that takes into account the object's type and data. The default implementation is equivalent to the following expression:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

See Writing a useful `toString` method (/reference/java/lang/Object.html#writing_toString) if you intend implementing your own `toString` method.

Returns
a printable representation of this object.

public final void **transform** (Matrix matrix) Added in API level 11

Applies a transformation matrix to all of the points in the event.

Parameters

matrix	The transformation matrix to apply.

public void **writeToParcel** (Parcel out, int flags) Added in API level 1

Flatten this object in to a Parcel.

Parameters

out	The Parcel in which the object should be written.
flags	Additional flags about how the object should be written. May be 0 or `PARCELABLE_WRITE_RETURN_VALUE`.

Protected Methods

protected void **finalize** () Added in API level 1

Invoked when the garbage collector has detected that this instance is no longer reachable. The default implementation does nothing, but this method can be overridden to free resources.

Note that objects that override `finalize` are significantly more expensive than objects that don't. Finalizers may be run a long time after the object is no longer reachable, depending on memory pressure, so it's a bad idea to rely on them for cleanup. Note also that finalizers are run on a single VM-wide finalizer thread, so doing blocking work in a finalizer is a bad idea. A finalizer is usually only necessary for a class that has a native peer and needs to call a native method to destroy that peer. Even then, it's better to provide an explicit `close` method (and implement `Closeable` (/reference/java/io/Closeable.html)), and insist that callers manually dispose of instances. This works well for something like files, but less well for something like a `BigInteger` where typical calling code would have to deal with lots of temporaries. Unfortunately, code that creates lots of temporaries is the worst kind of code from the point of view of the single finalizer thread.

If you *must* use finalizers, consider at least providing your own `ReferenceQueue` (/reference/java/lang/ref/ReferenceQueue.html) and having your own thread process that queue.

Unlike constructors, finalizers are not automatically chained. You are responsible for calling `super.finalize()` yourself.

Uncaught exceptions thrown by finalizers are ignored and do not terminate the finalizer thread. See *Effective Java* Item 7, "Avoid finalizers" for more.

Throws

Throwable	