
 Android Public Tracker 36983666 ▾


LatinIME: fails to delete pre-existing characters if view has no InputConnection and uses keycodes to handle input

+121

Hotlists (2)

Mark as Duplicate





Comments (50)DependenciesDuplicates (0)Blocking (0)Resources (16)

FixedBugP3

+ Add Hotlist


[AOSP] assigned

[AOSP] FutureRelease

 STATUS UPDATE

No update yet.

Edit

 DESCRIPTION zh...@yahoo.com created issue #1

In the Android 4.4 branch of LatinIME, there is a regression in the backspace key handling that prevents users from deleting existing texts in textboxes if applications use keycodes to handle text

I've tracked down the line that caused the regression, and believe this should be fixed.

In LatinIME, if the application targets ICS or lower, keycodes are sent to the app when keys are pressed. Many application rely on this behaviour, especially apps that use native activities. I've mar world name).

Reproduce:

- Create a native activity that handles text editing through key events
- Create a text box with pre-existing text
- Press Backspace

On Android 4.3 version of Google Keyboard: backspace keycode sent, app deletes characters


On Android 4.4 version of Google Keyboard: app does not delete characters. If new characters are entered, those characters can be deleted, but not the preexisting characters

I'm running Android 4.3 build JWR66Y on a Nexus 7 2012. These tests were done with Google Keyboard 2.0.18933.905102a.


I believe I've tracked down the line of code that is causing this problem: https://android.googlesource.com/platform/packages/inputmethods/LatinIME/+android-4.4_r1.1/java/src/com/android

✓ Mentioned issues (1)

✓ Links (12)

 Mentioned issues (1)

P3 Android is not open Bluetooth 4.0 for developers "I've developed a workaround for this issue, and for the related [issue 36949180](#) , for use by those apps that for the present continue to use TYP

 Links (12)

" ... I've tracked down the line of code that is causing this problem: https://android.googlesource.com/platform/packages/inputmethods/LatinIME/+android-4.4_r1.1/java/src/com/android/inputmett
"<http://www.android-ide.com>"


"This is working as intended. The thing is that applications shouldn't assume that synthetic hardware key events are always sent for backspace. Please refer to API documents of relevant methods su

"As for null/empty InputConnection considerations, both are explicitly app bugs according to the documentation: <http://developer.android.com/reference/android/view/inputmethod/InputConnection>.

"Demo code: <https://github.com/appfour/LatinImeBackspaceTest>"

See all related links

COMMENTS




sp...@gmail.com <sp...@gmail.com> #2

I have the same problems in many apps.


Reproduce:

- open terminal emulator
- write something
- put the app terminal emulator in background
- put the app termianl emulator in foreground
- try to delete with new android keyboard what you had written previously.



id...@googlemail.com <id...@googlemail.com> #3

We have the same problem in AIDE (www.android-ide.com). For now we have worked around this issue but it breaks many apps.



id...@googlemail.com <id...@googlemail.com> #4

This problem does not only affect views without InputConnection. If you implement your own InputConnection and return an empty string for InputConnection.getTextBeforeCursor() the bac



en...@google.com <en...@google.com>

Assigned to kw...@google.com.



kw...@google.com <kw...@google.com> [#5](#)

This is working as intended. The thing is that applications shouldn't assume that synthetic hardware key events are always sent for backspace. Please refer to API documents of relevant met



sp...@gmail.com <sp...@gmail.com> [#6](#)

Someone have the last LatinIME recompiled without this bad behaviour ? in this way, waiting for the correction, we can use the latest google keyboard without problems.



kw...@google.com <kw...@google.com>

Status: Won't Fix (Intended Behavior)



zh...@yahoo.com <zh...@yahoo.com> [#7](#)

I disagree with your opinion that this is working as intended. This is correct behaviour if the application is targeting Jelly Bean or higher, since those apps do not have keycodes sent to them,

Spotgra..., if your device is not Android 4.4, couldn't you revert to an older version of Google Keyboard?



sp...@gmail.com <sp...@gmail.com> [#8](#)

No i want the last version with the good behaviour for all. thanks to your comment:

"I believe I've tracked down the line of code that is causing this problem: https://android.googlesource.com/platform/packages/inputmethods/LatinIME/+android-4.4_r1.1/java/src/com/an

i solved recompiling the last latinime. i removed the LatinImeGoogle.apk from /system/app and reinstalled my recompiled version with different signing key as a standard apk

I disagree with the opinion that this is working as intended.



sp...@gmail.com <sp...@gmail.com> [#9](#)

i deleted this part from the source file LatinIME.java:

```
if (codePointBeforeCursor == Constants.NOT_A_CODE) {  
    // Nothing to delete before the cursor. We have to revert the deletion states  
    // that were updated at the beginning of this method.  
    mDeleteCount--;  
    mExpectingUpdateSelection = false;  
    return;  
}
```

and all work well.



id...@googlemail.com <id...@googlemail.com> [#10](#)

How is this supposed to work for terminal emulators or remote desktop apps which can not return the text before the cursor because they don't have access to this information? Should they

And how many chars is an editor supposed to return for InputConnection.getTextBeforeCursor()? According to the doc it should return up to n characters but as it is currently implemented th

This should be fixed at least for InputType.TYPE_NULL text fields unless it is Google's position that the default keyboard does not support InputType.TYPE_NULL text fields anymore in the Ki



kw...@google.com <kw...@google.com> [#11](#)

Status: Assigned (reopened)

We'll fix this for InputType.TYPE_NULL.



id...@googlemail.com <id...@googlemail.com> [#12](#)

Thanks for fixing that for InputType.TYPE_NULL.

But also please note that the documentation for getTextBeforeCursor(int n, int flags) states:

"IME authors: [...] please keep in mind the Editor may choose to return less characters than requested even if they are available for performance reasons."

With the KitKat LatinIME backspace only works for as many times in a row as characters are returned here, clearly not taking into account that the number of characters returned can be less



kw...@google.com <kw...@google.com> [#13](#)

I don't think that's the case here. A call to handleBackspace() basically just takes care of a deletion of one letter before the cursor. The method gets called repeatedly as the user keeps deleti

jc...@google.com <jc...@google.com> [#14](#)

There are several things at work there and many considerations to be had. Let's start with theoretical points, and see what we should do in Google Keyboard afterwards.

First, the most important thing to understand here is: on a mobile device, expecting a software input method to even *have keys at all* is broken. There is voice, there is handwriting, there are working with them or forcing users into an outdated/ill-adapted method instead of their method of choice is bad user interaction. And we want to avoid bad user interaction in Android. So in short: if a mobile app requires key events, be it backspace or otherwise, there is something wrong with the app and it should be fixed.

As for null/empty InputConnection considerations, both are explicitly app bugs according to the documentation: <http://developer.android.com/reference/android/view/inputmethod/InputConnection> - As the doc states, "The InputConnection interface is the communication channel from an InputMethod back to the application that is receiving its input": so if you have no InputConnection, you must return null. - InputConnections that return empty strings are also explicitly non-compliant according to KitKat's documentation for InputConnection: "Also, you may return less than n characters if the cursor is really at the start of the text."

This doc has been clarified in KK about this very point, but the workings are the exact same as in JB and before. Explicitly, if the app returns 0 characters when there are characters before the cursor, just ignore the function and never return anything.

Also, Google keyboard will not stop deleting after n backspaces unless the app is also lying about the cursor position in updateSelection() or EditorInfo.initialSel{Start,End}. Then again, if the

Actually, Google Keyboard never supported TYPE_NULL at all. You can notice, for example, you never get a key event when pressing alphabetic keys under any circumstances. This is both un

In the long run, we want the apps to fix their unreasonable assumptions about input on a mobile device. For an app like Minecraft PE, I think it should just use a TextView and get correct behavior. For a terminal emulator, the situation is a bit sad, but the fact is, the protocol assumes a technology that's slowly becoming irrelevant and the app will have to cope with this generation gap. If buttons even if some devices may have them. So a terminal emulator will have to either supply its own keyboard interface that's not an IME, or to correctly implement the API and map modifier

Now, for actual action on the Google Keyboard side, I believe the call for backward compatibility is a reasonable one to make. Really, Google Keyboard should never have sent key events for the expectation and embrace new, not key-based input styles.

Likewise, though Google Keyboard does not really support TYPE_NULL, I think it's marginally better to send the event in the TYPE_NULL case, so let's do this too.

sp...@gmail.com <sp...@gmail.com> [#15](#)

jchal...@google.com

An app that was made to control another android device, (type on your tablet with the keyboard of your phone via wifi or bluetooth, when the tablet is attached to tv via hdmi) can't use the Android

Why if a phone not have a physical keyboard can't send keyevents ? i'm only asking.

All previous android keyboard send keyevents what are the problems ?

jc...@google.com <jc...@google.com> [#16](#)

If you require something that is like a hardware keyboard, yes you'll need to code it yourself. The platform won't help you make a bad user interface.

If a phone doesn't have a physical keyboard, it's natural it can't send key events, right? It doesn't have keys.

Previous android keyboards did not send key events, except for backspace. One of the problems is: people make apps that require key events, and that breaks IMEs that don't look like keyboards

sp...@gmail.com <sp...@gmail.com> [#17](#)

[Comment deleted]

sp...@gmail.com <sp...@gmail.com> [#18](#)

jchal...@google.com

Sorry for this other question, i hope is the last and is only to understand.

when i write this code in my Activity and it work well when i use all previous android keyboard and also the actual 4.4 keyboard, the android keyboards are sending key events seems.

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch(keyCode)
    {
        case KeyEvent.KEYCODE_A:

            // do something

            return true;
        }
    return super.onKeyDown(keyCode, event);
}
```

and this work for all letters. only the delete key not work with the last latinIme.

in this case the Android keyboards are sending key events or am I wrong ?

and in the future will continue to work ?

jc...@google.com <jc...@google.com> [#19](#)

I don't see how this "works well".

- It does not work if the focus is in a text field.

- It requires you to force the keyboard to display by hand, though this is explicitly discouraged. This breaks users habits that the keyboard should pop to enter text, in usual components. It can also break with dead keys for French or German.

So essentially, it does not work for 90% of users.

It's actually not implemented by the keyboard ; it's a fallback mechanism that dates back to the G1, and that is rather obsolescent. I am not aware of specific plans about this, but I would certainly like to see it implemented.

sp...@gmail.com <sp...@gmail.com> [#20](#)

To understand look at an app like DroidMote Client

Also gesture, other language supported by the app, special characters and voice dictation are possible

```
public boolean onKeyMultiple(final int inKeyCode, int repeatCount, KeyEvent event)
```

Why you want to force to use a text editor if in a remote control app is possible to write directly with the beautiful Google Keyboard ? A remote app writes directly to the remote system and not in the text editor.

There are thousands of apps for remote control also Splashtop, VNC and others and you want to break all this beautiful apps usability ?

sb...@gmail.com <sb...@gmail.com> [#21](#)

this backspace issue is very frustrating I cannot correct text I put in by voice. It worked perfectly on 4.3. Please help us fix this make finish you a system update patch it's really annoying

to...@gmail.com <to...@gmail.com> [#22](#)

I have the same problems too! I experienced it with an application that I developed with Kivy and Python for Android. I cannot modify or delete a predefined text in my TextInput widget and I cannot delete the text before the cursor.

kw...@google.com <kw...@google.com> [#23](#)

Marked as fixed.

Will be fixed for InputType.TYPE_NULL.

jc...@google.com <jc...@google.com> [#24](#)

> you want to break all this beautiful apps usability ?

No, we want them to implement input correctly so that it works with Japanese, Chinese, gestures, voice and future input methods.

> Kivy and Python work with native coding, could it be that?

Probably not. The issue is likely that the app is not responding correctly to `getTextBeforeCursor` and does not call `updateSelection` properly.

to...@gmail.com <to...@gmail.com> [#25](#)

ok but what is the solution? a fix from Android or a workaround in the code?

kw...@google.com <kw...@google.com> [#26](#)

Stay tuned for the next update of Google Keyboard on the Play Store.

id...@googlemail.com <id...@googlemail.com> [#27](#)

- Show quoted text -

- Show quoted text -

This is not working for us in our tests. I have created a widget which works just like `EditText` except that it only returns the last character before the cursor. With this example you can easily reproduce the issue.

Demo code: <https://github.com/appfour/LatinImeBackspaceTest>

Demo APK: <https://github.com/appfour/LatinImeBackspaceTest/releases/download/v0.1/LatinImeBackspaceTest.apk>

kw...@google.com <kw...@google.com> [#28](#)

Looks like you have:

```
private static class GetTextBeforeCursorLimitedInputConnectionWrapper
    implements InputConnection {
```

that may not work.

<http://developer.android.com/reference/android/view/inputmethod/InputConnection.html> says "Applications should never directly implement this interface, but instead subclass from `BaseInputConnection`".

id...@googlemail.com <id...@googlemail.com> [#29](#)

Re [comment #31](#):

I don't see how this is relevant because `GetTextBeforeCursorLimitedInputConnectionWrapper` is only an adapter around the `(Base-)InputConnection` returned by `EditText.onCreateInputConne`

Nevertheless I have done that and still get the same incorrect behavior as expected.

Demo code: <https://github.com/appfour/LatinImeBackspaceTest>

Updated demo APK: <https://github.com/appfour/LatinImeBackspaceTest/releases/download/v0.2/LatinImeBackspaceTest.apk>

kw...@google.com <kw...@google.com> [#30](#)

Apparently you are hitting a different issue with this sample app. The text appended by the IME seems to be deletable. Also, can you explicitly specify `inputType` for your `EditText`? Google Ke

jc...@google.com <jc...@google.com> [#31](#)

[- Show quoted text -](#)

There are two things I want to say on this.

1) You rightfully point out that contrary to what I said, your sample app does update the cursor position correctly, and the keyboard still stops after one character in this case. I had forgotten cursor position but the keyboard does not account for it correctly in this case. It's fixed in development versions; it will ultimately be published with a future release.

Now in the practice, if the app returns a reasonable amount of text, there won't be any problems. If the app limits itself to returning 1024 chars or more, it will work in all cases; even with a fe

2) I want to point out that your sample app is still absolutely not compliant. In effect, you subclass the edit text, and you break one of the methods it uses for communication with the IME. It's same time for example will be very confused and offer a very bad experience.

As per the documentation, the editor may limit the amount of text returned `_for_performance_reasons_`. Stopping at 1 character is certainly not a sensible performance limit.

If your app handles edition of very large amounts of text, go ahead and return only a few kB. But don't limit to 1 char, or 5, or one line. The char limit clause is an exception for maintaining goc

to...@gmail.com <to...@gmail.com> [#32](#)

In previous comments I read that we should wait for this issue to be fixed by an update. Since android kitkat 4.4.1 is official I read the fixes that were made and could not find any fix for the i

jc...@google.com <jc...@google.com> [#33](#)

Again, an app relying on this to be sent is broken. You have `*no*` guarantee to receive key events, and it's not going to change.

Apps should stop relying on key events for text input because it causes a sub-par user experience, and use the mechanisms Android offers them for rich text edition instead.

sp...@gmail.com <sp...@gmail.com> [#34](#)

Reading some of this answer you can understand why Android is not a real open source OS but is the OS of Google. The default keyboard of an os is a sensible part and google should not de This story seem the same of Microsoft with Internet Explorer. You want force people to use what you want. In an os without phisical keyboard that the default os keyboard send keyevent is n

go...@gmail.com <go...@gmail.com> [#35](#)

In my own app I am using a non-`TextEdit`-based view with an `onCreateInputConnection()` method that returns a `BaseInputConnection` having `false` as its second constructor parameter ("`dumr` Android 4.3 (I have not yet loaded the pending OTA update to bring the device itself to 4.4).

My view is derived from `GridView` and represents a word game board. The user can tap anywhere on the board and that square will be selected, and the soft keyboard will appear (unless a h

So, in this configuration, there is no simple editor buffer with a linear sequence of characters. Characters go wherever the user has last tapped, and can overwrite each other. These characte letters only. Here is a link to the app in case you want to see how it interacts with the keyboard:

<https://play.google.com/store/apps/details?id=com.goalstate.WordGames.FullBoard.trialsuite>

The released app has its `targetSdkVersion` set to 9. Thus, it has no problems obtaining key events, including `KEYCODE_DEL` events. I'm in the process of updating it to `targetSdkVersion 19` (;

* All alpha keys work just as before. There is no problem with these keys.

* The backspace key also works fine (i.e., sends `KEYCODE_DEL` events), so long as I have typed more letters onto the board (since the last time I selected the board) than the number of times: backspace key fails to send any `KEYCODE_DEL` events.

I understand the desire to force apps to support more complex kinds of user input than the keyboard, but for my particular app, the maintenance of an editing buffer would be a very unnatura

Furthermore, I'm not aware that any decision has been made to simply not support dummy mode, and so long as dummy mode is being supported (and it `*is*` presently being supported - the result in a `KEYDOWN` event, and every key released would result in a `KEYUP` event, when in "dummy" mode, regardless of the position within what is presently an Editable buffer created by de

I hope that this is a useful example. This behavior has stalled my upgrade to API level 19. I'm looking at a workaround to generate my own key events by overriding `getEditable()` to return a v

I'm willing to do things the "right" way, but what is the "right" way in this case if key events are no longer supplied by the default soft keyboard?

go...@gmail.com <go...@gmail.com> [#36](#)

I should add that when I use the third party SwiftKey IME, my app (the unreleased, API Level 19 version presently in development) has no problem with receiving KEYCODE_DEL events.

jc...@google.com <jc...@google.com> [#37](#)

> I understand the desire to force apps to support more complex kinds of user input than the keyboard, but for my particular app, the maintenance of an editing buffer would be a very unnatural

Yes: this is the key point ! An IME is for inputting text.

Your app is not expecting text input. It's expecting button presses. IMEs are tools for text input. Not for button presses. I believe your app should be implementing what style is appropriate for

To illustrate what the problem is, let's consider the following case. Your app is only using Latin characters, right? So what happens when a user enters the app with a Korean keyboard that can

This is what we are talking about here. If your app needs text, it should be able to accept any input method. If your app needs button-like keys, you cannot expect the IME to do this for you: it has modes that will make your app painful/impossible to use.

go...@gmail.com <go...@gmail.com> [#38](#)

@spotgr...: for your fix, I'm wondering whether instead of removing that entire clause as you described, if you would want to just scope it so that it excludes TYPE_NULL?

```
if ((codePointBeforeCursor == Constants.NOT_A_CODE) && !currentSettings.mInputAttributes.isTypeNull()) {  
    // Nothing to delete before the cursor. We have to revert the deletion states  
    // that were updated at the beginning of this method.  
    mDeleteCount--;  
    mExpectingUpdateSelection = false;  
    return;  
}
```

go...@gmail.com <go...@gmail.com> [#39](#)

@jchal...: I do understand your point, but I'd like to distinguish between that position, which certainly has merit from a human interface standpoint, and what is presently implemented in the code

Because the present code does in fact support a second "fullEditor" argument for the BaselineInputConnection constructor (along with an EditorInfo.TYPE_NULL value), which, when passed as false,

So what we presently have is not a design decision to cease support for an event-generating IME mode; rather, we have *support* for an event-generating IME mode that has a small but critical

Leaving that bug in place would certainly not be an optimal way to force conformance to a new policy regarding allowable IME usage. To implement a design change in the use of the IME constructor would be provided. Then, at some point, the code would actually cease to support the capability (except for backward compatibility support of code targeted to earlier API levels)

Regarding whether a keyboard should have a dummy mode at all, well, if it didn't, then I'd have to implement a panel of buttons that would behave very much like the default keyboard but provide an

How to introduce this change is a design decision that hasn't been announced, even if there is some evolving consensus regarding how it ought to be done. It seems reasonable to expect that

go...@gmail.com <go...@gmail.com> [#40](#)

Anyway, I'm confused, because it looks like your commit right here from 10/9/2013 "Send backspace as an event when TYPE_NULL" incorporates the fix:

<https://android.googlesource.com/platform/packages/inputmethods/LatinIME/+/-/b41bea65502ce7339665859d3c2c81b4a29194e4>

So maybe I'm arguing for something to which everyone has already agreed, and it's just a matter of time before it is released. If so, I apologize for any confusion that this may have caused.

sp...@gmail.com <sp...@gmail.com> [#41](#)

Google thanks thanks thanks for the update.
I'm at your disposal :)

jc...@google.com <jc...@google.com> [#42](#)

Note that this change is for TYPE_NULL, not for dummy connections.

For TYPE_NULL, the API explicitly states the IME should try to send key events, and that's what Google Keyboard will be doing. Note that still, TYPE_NULL support is very spotty among all IMEs

Still, if this was breaking your app, you should really consider fixing it. Any app broken by the lack of backspace event -- even when TYPE_NULL -- should be fixed, as there are many IMEs out there

sp...@gmail.com <sp...@gmail.com> [#43](#)

I'm speaking about apps that simulate keyboard in remote android device. And the best user experience is to write directly to the remote system and not first on text box and after send to remote system
For apps that simulate keyboard in remote android device, you must support language one by one also if you use the TEXT BOX compliant version.

I personally also have a version with a different implementation that is no key event employee, but for my users the current implementation is the best.

However, I understand the needs of google, and I accept and agree with, but I hope that also in future, at least the default open source android keyboard, will send key-event for TYPE_NULL as well

I just try to explain with my bad english, but I'm very happy with how google supports its developers now.

Thanks for the UPDATE.



go...@gmail.com <go...@gmail.com> [#44](#)

[Comment deleted]



go...@gmail.com <go...@gmail.com> [#45](#)

I've developed a workaround for this issue, and for the related [issue 36949180](#), for use by those apps that for the present continue to use TYPE_NULL behavior to trap key events.

The presentation of this workaround describes the concerns that Jean has pointed out, while at the same time providing a way for apps that presently rely upon TYPE_NULL behavior to continue. <http://stackoverflow.com/questions/18581636/android-cannot-capture-backspace-delete-press-in-soft-keyboard/19980975#19980975>

I would appreciate any comments regarding this workaround, as well as hearing about any results obtained if people use it, and also any improvements that might be made to it.



[Deleted User] <[Deleted User]> [#46](#)

I have an opengl application that uses a SurfaceView. In other platforms that my code runs (iOS and WP8), i put an Edit field in the screen at a hidden point and gather text from it, passing all characters on it, i can no longer delete them. Can someone give me an insight of how to bypass this problem? thanks.



lo...@gmail.com <lo...@gmail.com> [#47](#)

@regis If you are using Unity C#, this can be walked around by using `UnityEngine.Input.GetKeyDown(KeyCode.Backspace)`, if it's true then remove the last character in the `TextEdit`.



[Deleted User] <[Deleted User]> [#48](#)

lorry
`UnityEngine.Input.GetKeyDown(KeyCode.Backspace)`

don't work

what can i do?



lo...@gmail.com <lo...@gmail.com> [#49](#)

@rafael
It's a little bit complicate, but you can write a unity java jar plugin,

<http://docs.unity3d.com/Manual/PluginsForAndroid.html>

then use the answer

<http://stackoverflow.com/questions/4886858/android-edittext-deletembackspace-key-event>

to handle it.

I've did this in 2 days and it works well on devices include Nexus(In java we can do anything, but not in JNI layer that Unity provided)



sp...@gmail.com <sp...@gmail.com> [#50](#)

With the last version in Google Play, with Android versions < 5 if you use the keyboard in landscape on apps with `InputType.TYPE_NULL`, you can't write anything

E/InputMethodService(2755): Unexpected null in startExtractingText : mExtracted
Text = null, input connection = com.android.internal.view.InputConnectionWrapper
@4208f910