Effects of Android's SMS-URI parsing bug on NFC applications

Michael Roland
Upper Austria University of Applied Sciences
NFC Research Lab Hagenberg
{ michael.roland } @fh-hagenberg.at

This problem corresponds to issues #12142, #15866, and #15868 of Android. A hotfix is available on Android Market: SMS-fix for NFC.

Smart posters for SMS applications

Fig. 1 shows a typical scenario for the use of an SMS-URI record in an NFC application. The URI record contains a ready-made SMS message (that consists of the destination phone number and the message text). Additionally, the URI record is packed into a smart poster to add a textual description.

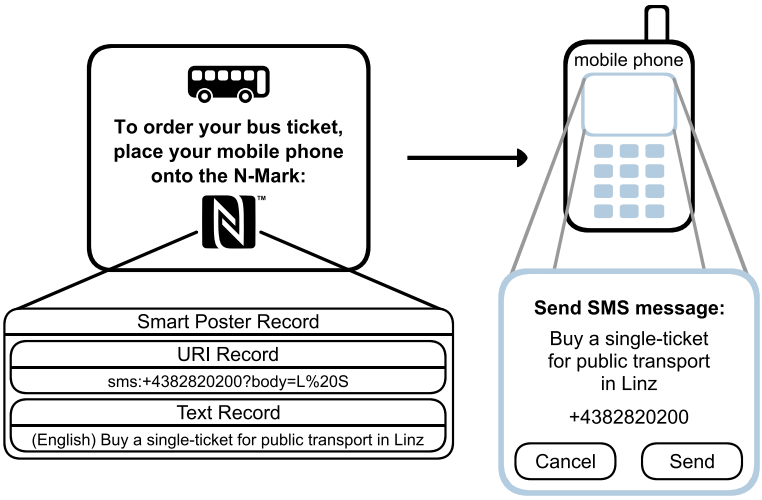

Fig. 1. This example of a smart poster record is located a t bus stop of a public transport system. The smart poster contains a ready-made SMS message to order a single-trip ticket. When the user touches the NFC tag with the phone, the message is transferred to the phone and the user only has to confirm to actually send the message.

The expected behavior for this scenario would be that the smart poster triggers a dialog with the user asking for confirmation. As soon as the user accepts to send the message, the predefined message will be sent to the predefined number (both taken from the URI record).

Format of the SMS URI record

The SMS URI record consists of the scheme prefix "sms:", one or more phone numbers (separated by ",") and an optional parameter named "body" that contains the message's text. Certain special characters in the message are represented by a percent sign followed by the hexadecimal character code (e.g. a space is encoded as "%20").

To send the message "L S" to the phone number "+4382820200", the URI would look like this:

```
sms:+4382820200?body=L%20S
```

Current behavior on Android

On Android, by default, the Tags application processes certain types of standard NDEF records received from NFC tags. Among these NDEF record types are Smart posters and URI records.

For our smart poster example from Fig. 1, the Tags application correctly processes the record, extracts the textual description and detects, that the URI record instructs the phone to send a text

message (SMS) to the phone number +4382820200 (see Fig. 2). In the next step, the user can click on the button "Text +4382820200" to start sending the SMS message.

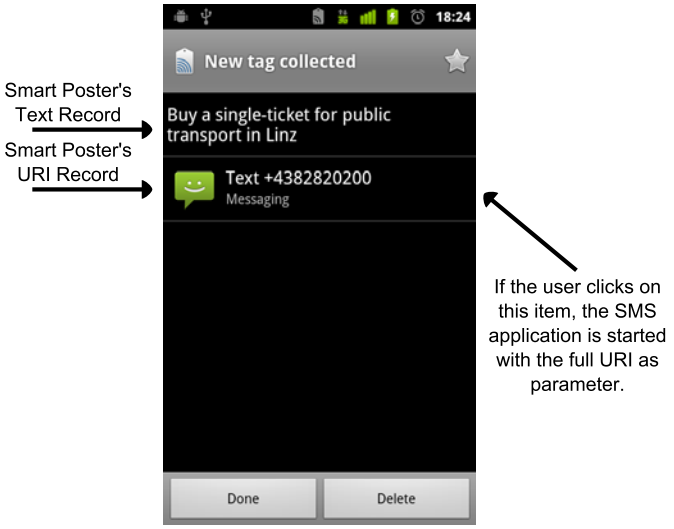

Fig. 2. The Tags application correctly processes the smart poster record, extracts the textual description and detects, that the URI instructs the phone to send a text message (SMS) to the phone number +4382820200. The user can then click on the button "Text +4382820200" to start the text messaging application.

When the user clicked on the button, the expected behavior would be that the messaging application is opened, the recipient's phone number is inserted into the recipient field and the message's text is inserted into the SMS text field. On Android 2.3.4, the actual behavior is different (see Fig. 3): Instead of the recipient's phone number, the whole scheme-specific part of the URI (i.e. everything after the "sms:") is inserted into the recipient field. The SMS text field remains empty. Therefore, the SMS is sent to an invalid phone. Moreover, the message does not contain the predefined text.

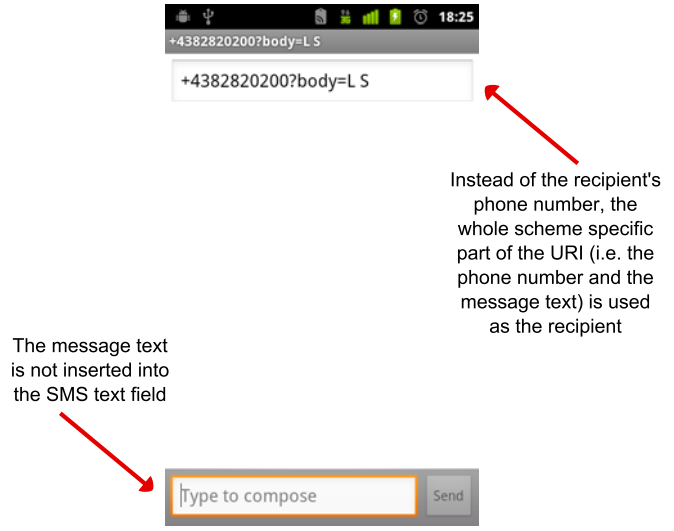

Fig. 3. On Android 2.3.4, the URI triggers the messaging application. This application extracts the whole scheme-specific part of the URI, "+4382820200?body=L%20S", and uses this as the recipient's phone number. The message text remains empty.

Expected behavior and a hotfix for this issue

A hotfix was created to avoid the issue[1]. Fig. 4 shows how the Tags application integrates this hotfix. If the user uses the new button that is created by the hotfix, the messaging application is started, and the recipient and text fields are filled correctly.

Fig. 5 shows the expected behavior for the SMS URI "sms:+4382820200?body=L%20S". In the messaging application the recipient field is prefilled with the phone number and the message text field is prefilled with the message text.

[1] Available on Android Market: https://market.android.com/details?id=at.mroland.android.apps.smsfixfornfc

Fig. 4. The hotfix adds a new handler for the SMS URI. This handler is now shown in the Tags application besides the messaging application for the SMS URI. When the user clicks on this new button, the messaging application is started in a way that avoids the problem.

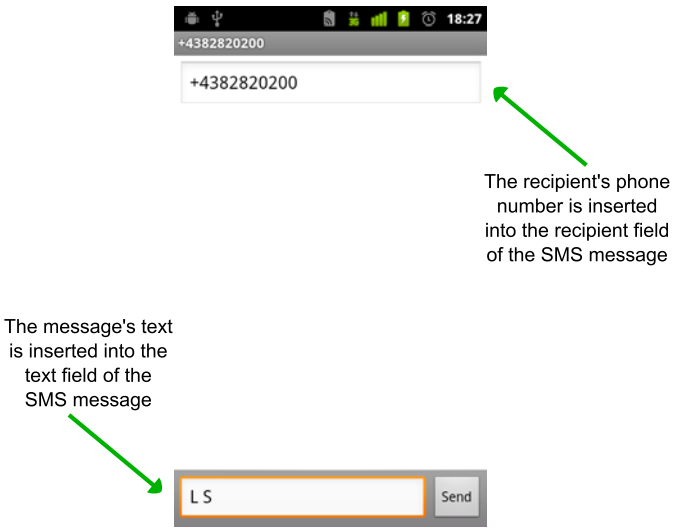

Fig. 5. This is the expected behavior. The messaging application is started, the phone number from the URI is used for the recipient field and the value of the "body" parameter is used for the message text field.

How the hotfix works

The hotfix installs a handler for SMS URIs (i.e. an activity with an intent-filter that triggers on the SENDTO action with a URI that has the "sms:" scheme):

```
<intent-filter android:priority="-1000">
  <action android:name="android.intent.action.SENDTO"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <data android:scheme="sms"/>
</intent-filter>
```

The activity then parses the SMS URI record. The parameters are stripped from the URI. Thus the remaining URI contains only the scheme and the recipient's phone number. A new intent is created with the action SENDTO and this URI. The extracted "body" parameter is then added to the new intent as the extra "sms_body". The best matching activity for the new intent is then started.

"sms_body" is an undocumented intent-extra that is processed by the messaging application. Therefore, this hotfix is a temporary solution that will only work with the current version of the messaging application that is delivered with the base Android system.