



Contents

- [1 Description](#)
- [2 Advantages](#)
- [3 Disadvantages](#)
 - ◆ [3.1 Fallback option](#)
- [4 Use when](#)
- [5 Use how](#)
 - ◆ [5.1 Design guidelines](#)
 - ◆ [5.2 Technology how-to](#)
 - ◇ [5.2.1 Symbian C++](#)
 - ◇ [5.2.2 Python](#)
 - ◇ [5.2.3 Java ME](#)
 - ◇ [5.2.4 Flash Lite](#)
- [6 Design tips](#)

Description

Mobile applications could offer to users the **ability to send SMS messages directly from within the application**. In order to offer this kind of functionality, a form for SMS composing is usually presented to the user, with characteristics that are easily recognizable.

Advantages

SMS Forms integrated in a mobile application give to the user the possibility to **send SMS messages without the need to leave the application** and open the default SMS composer. Also, being integrated within the application, the SMS Form can be **customized** in order to offer to the user a **richer or smoother user experience** (e.g.: filling fields with **pre-compiled data**, allowing to send a SMS message to **multiple recipients**).

Disadvantages

In order to offer the SMS sending functionality, the used technology must offer programmatic **access to Messaging** (to send the SMS message) **and (optionally) to Contacts** (in order to choose the recipient number from the phonebook). Depending on the used technology, this kind of access could or could not be available.

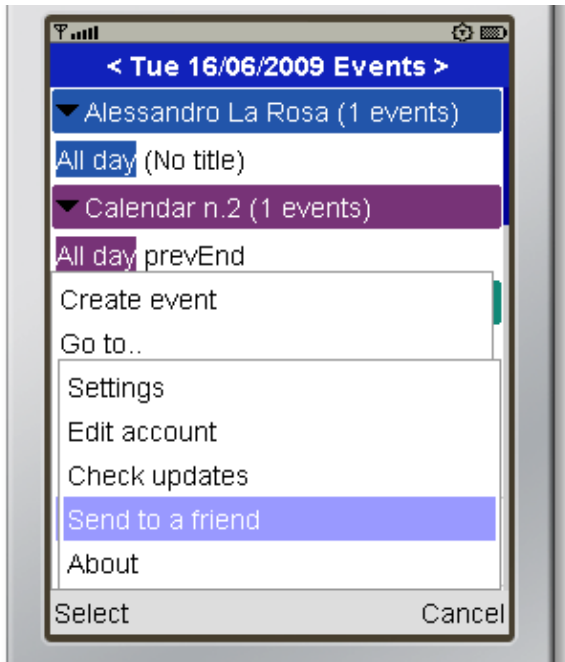
Fallback option

When **direct SMS sending from a mobile application is not possible** because of technology limits, a fallback option is, when available, to **open the default SMS editor**. If possible, the SMS editor **fields are pre-compiled** with data passed from the mobile application, in order to provide a **better user experience to the final user**.

Use when

The SMS Form Design Pattern is useful in all these situations where you want to allow users to compose and send SMS messages from within a mobile application. This can include scenarios like:

- **Messaging applications**
- **Sending an application download link** to a friend (the classical "Send to a friend" functionality), as shown in the following screenshot.



Use how

Design guidelines

SMS Forms are actually a **common pattern for most mobile users**, since SMS messaging is a functionality always present on mobile phones. For this reason, when adding a SMS Form to a mobile application, it is good practice to respect some **common layout and usability guidelines** that allow users to easily understand the various interface functionalities.



An SMS Form is typically composed of **few, well distinguishable UI elements**. These are:

- A **single-line field for the receiver's phone number**
- A **button** (typically placed near the phone number's field, or in the soft-keys' menu) to allow users to **choose a phone number from the device's phonebook**
- A **multi-line field to allow users to type the message body**
- A **button to send the SMS**

If the SMS sending is a core functionality of the application, it is also good practice to provide extra functionalities to the user, such as:

- **saving an SMS draft**, and the ability re-open and send saved drafts
- **sending a single SMS message** to multiple recipients, without the need to re-type the message for each number

Technology how-to

In order to correctly implement the SMS Form design pattern, the **used technology must allow the access to these device functionalities**:

- **SMS sending**: this functionality is usually offered through some sort of **Messaging API**
- **Access to phonebook**: in order to choose a phone number of a stored contacts

Depending on the used technology, these functionalities are offered through different methods and APIs.

Symbian C++

The following articles explain how to access stored Contacts, and how to send SMS messages from a Symbian C++ application:

- [Getting contact info from default database](#)
- [CS001307 - Symbian C++: Sending an SMS](#)

Python

The following articles explain how to access stored Contacts, and how to send SMS messages from a Python application:

- [How to choose a phone number](#)
- [How to send SMS](#)

Java ME

The following articles explain how to access stored Contacts, and how to send SMS messages from a Java ME application:

- [How to read contacts using JSR 75](#)
- [How to Send Text SMS in Java ME](#)

Flash Lite

The following articles explain how to access stored Contacts, and how to send SMS messages from a Flash Lite application, by using **Platform Services**:

- [CS001223 - Listing contacts in Flash Lite](#)
- [CS001249 - Sending an SMS in Flash Lite](#)

If Platform Services are not available, the default SMS editor can be opened with pre-compiled data, as shown in this Forum Nokia Wiki article:

- [How to send SMS/MMS messages](#)

Design tips

- Give to the user the ability to **directly type the number in the number field**, without requiring an extra click to access a separate view. If this feature is not natively supported by the used technology, check if it can be programmatically implemented.

Mobile_Design_Pattern:_SMS_Form

- When possible, give to the user the ability to **directly type the message in the message field**, without requiring an extra click to access a separate view.
- Always **support T9 functionality** where available.
- Provide a clear **notification about the sending status**