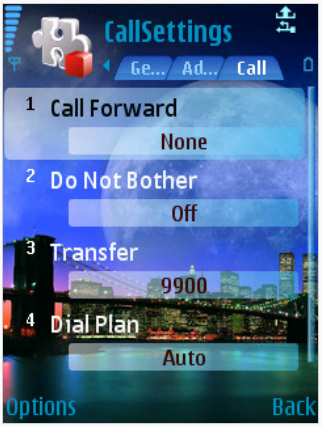
SettingList Usability

Introduction

Most of the applications have some sort of settings, either customizable or pre-populated. From a user's perspective it is essential to display those settings to the user so that they can change them if it's possible or otherwise they can at least view them to understand what values are being used by the application. Some examples of pre-populated settings could be username, self telephone number etc which are pre populated based on the settings on the server in case of a client-server application etc. Where as common user customizable settings could be volume level, auto start choice, language to use etc.

Settings list

Symbian C++ provides a setting list for the precise purpose of displaying and giving the choice of editing those settings to the user. The idea behind using a setting list is to be able to logically group all the related settings so that the user doesn't get confused if all of them were clustered together in one view/interface.



Example of settings list

Settings lists can be created from resources and they can also be created dynamically depending upon the usage pattern.

Examples of how to create setting list from resource

Settings Lists

Example of how to dynamically create a setting list

Create Dynamic Settings Pages using Symbian C++

Key points to consider while using setting list

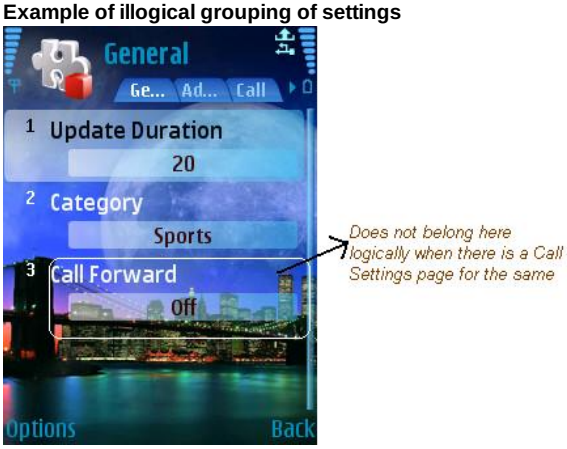
Like any other UI component, usability should be kept in mind while creating the setting lists as well. It should not happen that the user doesn't understand what settings s/he should enter and what are the respective limitations for each of the items on the setting page.

Some key points to consider while using setting list are as under:-

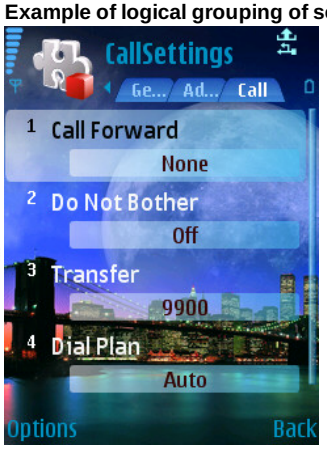
Ensure logical grouping of setting list items

The setting items being displayed on a setting page should belong together from a logical grouping point of view. For instance if the settings pertain to network then username/password and other non related fields should not be displayed there.

Example of illogical grouping of settings



Example of logical grouping of settings



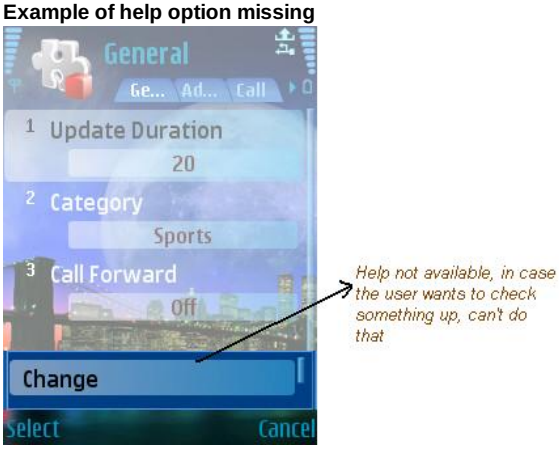
Include a specific help for the setting page/view

More often then not applications tend to have settings which are range bound or require some sort of conditions to be met for them

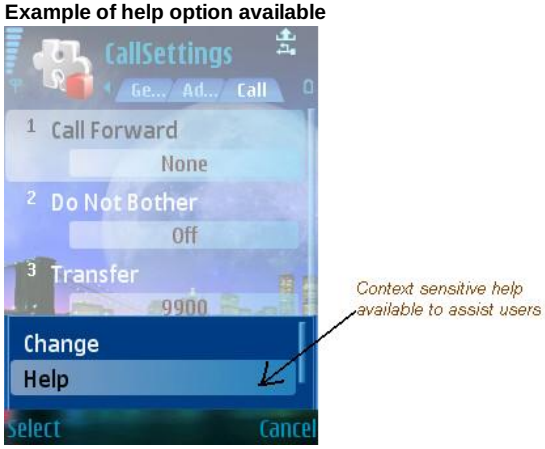
(C) Copyright Nokia 2014. All rights reserved.

to be valid. Hence it is imperative to provide a specific context sensitive help to the user on the settings page so that they can refer to it in case they so need it.

Example of help option missing



Example of help option available



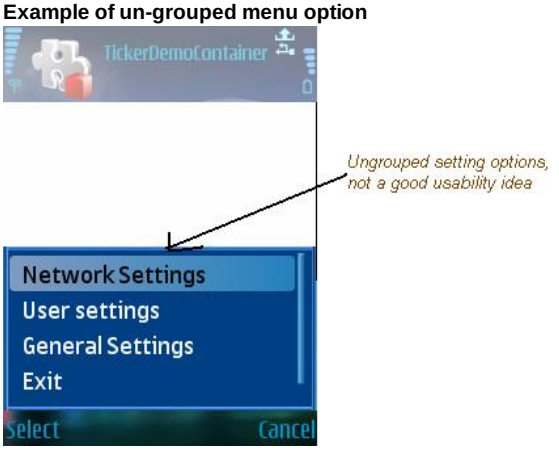
Indicate range/error conditions if any while setting the values

If there were any errors while setting values to the items, then the error should be indicated to the user and where possible help should be provided in terms of range considerations or any other checks that the user need to follow while setting the values. A generic error like **"Setting not saved"** should not be displayed as this would confuse the user.

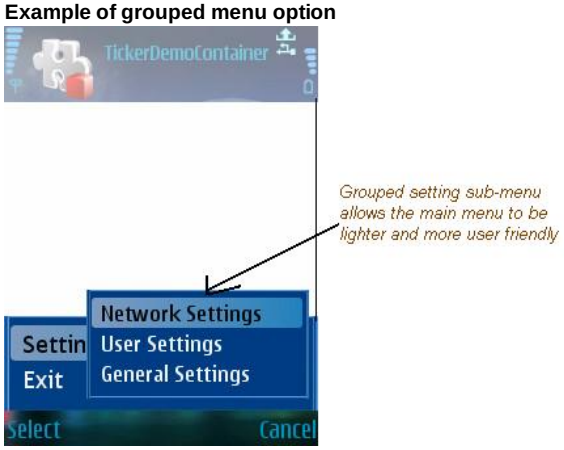
Group the option menu to navigate to the setting list

In case the application has more then one set of setting group and they are arranged in different views, then instead of providing multiple menu options, provide a single settings menu option which can have sub-menu for specific settings.

Example of un-grouped menu option



Example of grouped menu option



Provide tabbed interface where possible for settings

Where it is possible provide tabbed interface for the settings so that the user can select which set of settings s/he wants to view/modify. This gives the user a sense of uncluttered grouping of settings on the user interface. For instance an application which has multiple settings to deal with network, user rights, and media settings etc should have tabbed based settings list with each of them logically grouping each category of settings.

Example of un-tabbed setting list

Example of tabbed setting list

Hide/notify read only fields

There are lots of times when certain settings can not be modified by the user. In those cases either hide the corresponding setting items from the view or if you want to display them to the user make sure that you let them know that those fields are read-only both

(C) Copyright Nokia 2014. All rights reserved.

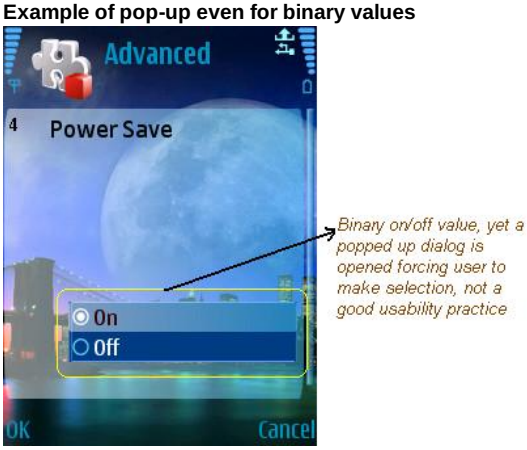
in the context sensitive help and also when they try to make changes to those fields by popping up a read only message.

Example of notifying read only

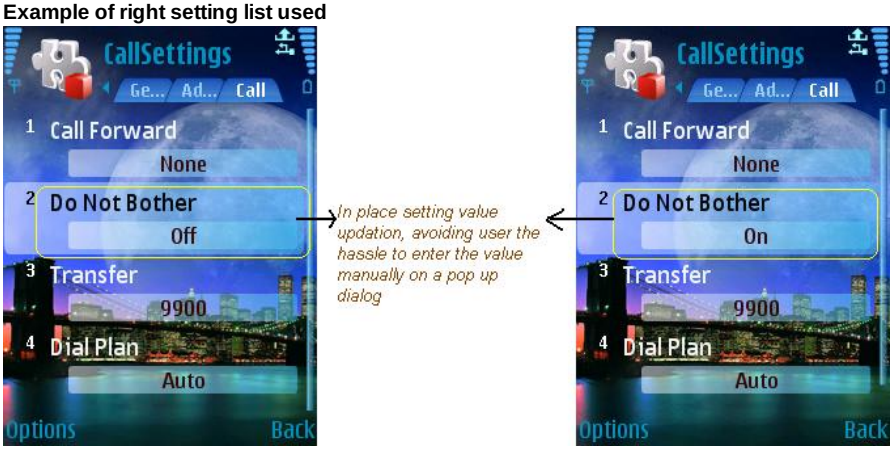
Save the settings to persistent storage

Use the right setting items, for instance in case of yes/no, on/off kind of binary values do not ask the user to explicitly enter the texts or navigate to a popup page only to give 2 user options. In those cases just toggle the values on the main setting page itself.

Example of pop-up even for binary values



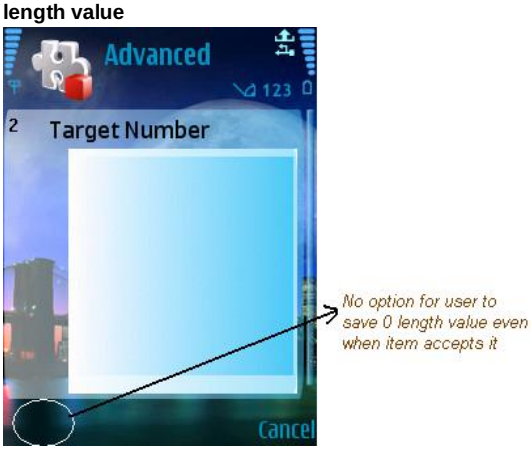
Example of right setting list used



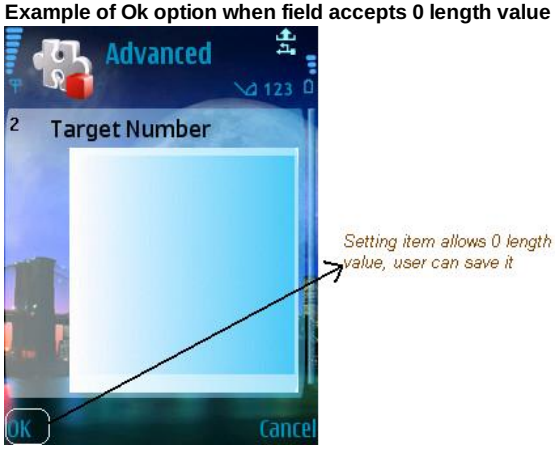
Allow user to save for 0 length values

In case the setting item accepts 0 length values, then provide the user the option to save the item by giving an 'Ok' option on the left soft key.

Example of Ok option missing even when field accepts 0 length value



Example of Ok option when field accepts 0 length value



Provide Back option for user to navigate to main view

Ideally setting views should not have soft key to allow the user to exit the application, they should instead provide the user the opportunity to navigate back to the main view of the application from where the settings view was invoked in the first place.

Example of back option not available

Example of back option available

(C) Copyright Nokia 2014. All rights reserved.

Save the settings to persistent storage

The settings should be loaded from and saved to a persistent storage so that the same can be retrieved when the application is started again. You certainly don't want the user to be entering all the values every time the application starts. Another thing to ensure is that the values are saved in a private folder so that they can't be altered by any other application.

Changes to the settings must be visible immediately

--- Added by Mayank on 23/06/2009 ---

(C) Copyright Nokia 2014. All rights reserved.