GrxSettings (for LP 5.1+)

Overview for Devs (draft)

By Grouxho

www.esp-desarrolladores.com

1.- Features

GrxSettings is an application that aims to help developers of mods, providing a series of types of preferences and characteristics that during the last years I needed..

The main features that the application provides to developers are:

- 21 customizable type of preferences
- Preferences can be declared using their short class name.
- Resources (booleans, default values..) to define the default values of attributes for some preferences as well as several application features.
- Easy navigation panel configuration, with groups support.
- 13 themes available. Easy customization of themes. Enough styleable attributes to build three different styled areas. Moro to come
- Customizable dependency rules, which although simple, allow a more advanced behavior than the use of Android standard dependencies
- Nested screens keeping the toolbar.
- Group keys, allowing the developer both the saving of system resources in their mods and an easier update of them
- Up to two different broadcast can be sent when a preference value is changed.This feature combined with group keys allows the developer different strategies for their mods implementation.
- Customizable Rom Info section, with sliding tabs support. Mods developers can design as many screens as they need to show information about their works. Some extended views has been added in order to make this work easier.
- Floating Action Button, with a standardized operation throughout the application.
- Backup and restore of preferences with synchronization of stored values in settings system
- From your mods you can easily open a screen, sub-screen and even simulate a click on a preference through intents with predefined intent extras strings

Since LP, the concurrent access to a given shared preferences file is not reliable (it is a shame). So, this app try to keep all the time synchronized its shared preferences values with their corresponding values in Settings System. The first time the app is executed this synchronization will be carried out, in order to apply the default values the dev configured in the preferences. When the user restore a backup this task will be also executed. Instead of reading each preference from the saved backup and then to write read values to the settings

system db (or settings system xml) I have chosen an alternative way, in order to try to reproduce exactly defaults values configured by the dev. What this app does is to reproduce transparently to the user all the screens configured in the navigation panel menu. In that way a given key that was not saved in the backup will be initialized with the configured default value.

The application also support nested preferences screens, keeping the toolbar and memorizing first seen preference when the user leaves a given sub-screen (if several nested preferences screens exist in the same preference xml).

Several options can be configured by the user:

- Theme selection, if this feature is allowed by the developer
- Navigation panel header background (image), if this featured is allowed by the developer
- Exit Confirmation option
- Lists divider height
- Navigation Panel position (right or left)
- Show - hide floating action button
- Floating Action Button position (left, center, right)
- Option for keeping always open the groups of screens defined by the developer
- Show buttons for collapse - expands groups of options.
- Remember last screen
- Double clicking on most of the preferences allows the user to reset its value to the default value. In color pickers a long click on the selected color allows the user to copy to the clipboard its value

2.- Building the app

GitHub:

3.- Tabs for Rom Info o whatever you need

The app support as many tabs as you need. These tabs are showed the first time the app is executed or if the users select Rom Info in the user options menu.

To add tabs:

a.- Layouts

- Add to the string-array “tabs_layouts” the name of the layouts you want to show

Example:

```
<string-array name="tabs_layouts">
    <item>demo_info_about</item>
    <item>demo_info_changelog</item>
    <item>demo_info_credits</item>
</string-array>
```

In this example the app will look for the following layouts: demo_info_about.xml, demo_info_changelog.xml and demo_info_credits.xml

The sliding tabs will be showed in the order the layouts appear in the string-array.

b.- Titles

Add the corresponding titles in the string-array “tabs_names” . You can use strings, of course or write the tabs names directly.

Example

```
<string-array name="tabs_names">
    <item>About this app</item>
    <item>ChangeLog</item>
    <item>Credits and Thanks</item>
</string-array>
```

c.- Main Tilte

The main title for this screen is specified in the string:

gs_rom_name

d.- How to show no tabs.

If you do not want any tab, just leave the tabs_layouts like this:

<string-array name="tabs_layouts"/>

A empty screen is showed.

- If you only add one layout then the sliding tabs will be hidden.

e.- Info Layouts requirements.

In order to keep the floating action button behavior in the same way than in the rest of the app, the info layouts **must** be based on

com.mods.grx.settings.fab.ObservableScrollView

As any scrollview in android, only one main container should exist inside it, so, I recommend a LinearLayout inside the observablescrollview containing all the views you want to add to the layout.

<com.mods.grx.settings.fab.ObservableScrollView ..

<LinearLayout android:orientation=”vertical” …

PLACE HERE YOUR TEXTVIEWS, CARDVIEWS, IMAGEVIEWS, GRIDLAYOUT or wherever you need

</LinearLayout>

</com.mods.grx.settings.fab.ObservableScrollView>

If the layout is not based on ObservableScrollView, the app will fc.

f.- Added views to make easier the content creation process

The following view classes has been added to be used, if you want to, in these layouts.

- **TextViewWithLink** (com.mods.grx.settings.views.TextViewWithLink)

  Apart from the standard android xml attributes for TextViews you can use (name space null)

  grxAnimateText=”true” -> the text, if there is not enough space, will be horizontally scrolled.

  grxURL=”yoururl” -> the url will be opened when the text is clicked.

- **LinearLayoutWithLink .** (com.mods.grx.settings.views.LinearLayoutWithLink)

Apart from the standard android xml attributes for LinearLayout you can use (name space null)

grxURL=”yoururl” -> the url will be opened when the text is clicked.

- **ImageViewWithLink**. (com.mods.grx.settings.views.ImageViewWithLink)

  Apart from the standard android xml attributes for ImageView you can use (name space null)

  grxURL=”yoururl” -> the url will be opened when the text is clicked.

  grxCircular=”true” -> the icon will be rounded

- **CardViewWithLink**. (com.mods.grx.settings.views. CardViewWithLink)

  Apart from the standard android xml attributes for CardView you can use (name space null)

  grxURL=”yoururl” -> the url will be opened when the text is clicked.

- **ArrayTextList** -> (com.mods.grx.settings.views.ArrayTextList)

  Extends a LinearLayout, so you can use all the standard LinearLayout . The folloging att has been added (null name space):

  grxA_entries =” @array/your_string_array_name”

  Inside the linearlayout a TextView will be inflated. The text assigned to the textview will be spanned with the items contained in the string array. The class will add a little circular bullet on the left of every text line. Usefull for changelists o whatever other info you want to show in a texts list way.

For more info, look inside de demo app to see how to configure the tabs and how to use these views.

The app supports as many tabs as you need and provide the support for that. No pre-designed tab has been added since is the developer using this app who decides what to show. You can show your rom info, changelist, team, credits, donation links, whatever you want to.

4.- Themes and Styleable attrs.

GrxSettings v1 has got 13 themes. The default theme is defined in the manifest.

- Parent themes for Light Styles:
    - Main App

      <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    - Navigation Panel

      <style name=" SublimeTheme" parent="ThemeOverlay.AppCompat.Dark">
    - Dialogs

      <style name="GrxDialogStyle">

  It is blue light and contains all the style attributes used in light themes for now.

- The parent theme for Dark Styles is
    - Main App

      <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    - Navigation Panel

      <style name="SublimeTheme_dark" parent="ThemeOverlay.AppCompat.Dark">
    - Dialogs

      <style name="GrxDialogStyle">

  It is blue dark and contains all the style attributes used in dark themes for now.

You can easily modify the themes, and use all the defined attributes in your info layouts.

The main content layout of the app is

grx_coordinator_layout.xml

The toolbar uses the main colorPrimary as background Color.

You can easily make/modify themes for using different colors in the toolbar, main floating action button, preferences and dialogs, navigation panel header background and navigation panel options and highlighted menus… Looking at the main layouts of the application and these themes is easy to know how it works.

Look at the Green Orange theme to see how to combine two colors easily.

Widgets Layouts in Preferences

Some attributes are used on auxiliary elements in the app. For example, three different widget layouts used in the preferences were added.

- widget_arrow.xml
- widget_accent_arrow.xml
- widget_compl_accent.xml

These layouts are used in the preferences declarations to show an arrow black/white, with accentColor or with an added complementary accent color defined in the themes with the attribute complemnt_accent_color.

If you want to design your own widget layout, you have to keep at least the views and ids contained in the predefined (icon and imageview for arrow)

There is another widget layouts used in some preferences, widget_text, but it is not desgined to be used in the preferences declaration but internally by the app.

Navigation Panel header background

As it has been said, the main layout is grx_coordinator_layout.xml . You will find the navigation panel header layout in grx_nav_header.xml

If you have a look at it, you will see that there is a default header background color defined by the attr svn_nav_header_bg

This is a FrameLayout, so I have added an ImageView at the end in order to contain your Rom Logo.

If you enable the user option to change the header background, the default background color will be changed by the user selected image, but your logo will be shown anyway.

To make this easier the Rom logo creation **(drawable/ header_image**) an almost transparent example has been added with the right dimensions to use.

- **Leave some transparency for the mini floating action button which provides access to the user options menu or modify this layout keeping the ids!!**

For now if you want to make your own theme you have to modify one of the existing, because the logic to change the theme is related to the position of each theme in the string array

gsa_theme_list

5.- Menus

The following xml menus are used:

Backup and Restore: menu_grx_ajustes.xml

User Configuration Options: menu_grx_conf_nav.xml .

The above menus should be not be modified, since they are part of the logic of the application. If you do not want to use them, you will have to modify the source code.

You can disable the following options in the user menu:

- Selection of Background image: Change the Boolean allow_user_panel_header_bg to false if you do not want the user to change the navigation panel header with an image.

The navigation panel will use as the main navigation menu one of the following two xml, depending on the value of the Boolean "DEMO"

menu_grx_demo.xml : For demo purposes. (DEMO = true)

menu_grx_nav.xml: Since I will provide compiled demo apps, you should use this menu for your mods. If you want to clean all resources related to the demo, go to the source code and read comments in xmls0

6.- Navigation Menu

This app uses SublimeNavigationView library. This library allows us to have different menus managed in the same navigation panel. As said, two are used: user options, navigation options.

I wrote some minor modifications in the library to make it works the way I like.

You should not change anything inside menu_grx_conf_nav.xml (the user options menu) since the applications expects the things inside it to be as they are ☺

In order to create your preference screens, the used library supports several features, but the implementation done in the app expects the menu items to be configured as is described in this document.

6.1.- The preferences screens menu - menu_grx_nav.xml

- This is the menu you should use to configure your navigation panel options
- At least one item needs to be added, or the app will FC.
- We will use the following SublimeNavigationView library type of items in this menu.
    - Text : A item of type Text is used for every preference screen to be shown.
    - Separator: this simply add a line, for aesthetic purposes.
    - Group : to make Text groups. When you create a Group, you need to add a GroupHeader

6.2.- How to add a preference screen to the navigation panel.

Really easy to do:

- In menu_grx_nav.xml add a Text item with an id (mandatory)
- Create a preferences xml with the id name

Example

<Text android:id**="@+id/demo_check_switch**" android:title="GrxCheckBox - GrxSwitch " />

The navigation panel will show the text GrxCheckbox-GrxSwitch. If the user press it, the app will load the xml called demo_check_switch with the preferences added for this option.

As easy as this. You can create as many Text items (preferences screens) as you want to.

Of course you should use @string to make this as it should be, but this is just an example ☺

Other Attributes in Text items declarations

- android:icon=”@drawable/your_icon” -> will add an icon to the left of the text
- android:hint=”@string/your_string” -> it will add a subtitle with smaller text size thant the title.
- app:showIconSpace="true" -> This attribute leaves to the left of the text the space corresponding to an icon. It is usefull if you do not want to add an icon to the options but want to get the text with left margin (when the options belongs to a group, for example).

6.2.- How to add separators

Wherever inside menu_grx_nav.xml you can add

<Separator/>

This will add a line.

Example

<Separator/>

<Text android:id="@+id/demo_check_switch" android:title="GrxCheckBox - GrxSwitch " />

<Separator/>

6.3.- Groups of Menu Items

If you add many preference screens it is usefull to group them in the navigation panel. Groups are expandable and collapsible.

Adding a Group

<Group android:id="@+id/demo_grupo_apps" app:collapsible="true" app:collapsed="true">

The id is mandatory.

app:collapsible -> true or false, the group can be expanded/collapse or not.

app:collapsed -> how it is showed when the app starts.

GroupHeader

In SublimeNavigationView is mandatory to add a GroupHeader belonging to a group. The GroupHeader shows the group text. You can add the same attributes used in the Text items (icon, title, hint..). To assign an **id** is mandatory.

So a very simple group example with two options should contain at least the following elements.

```
<Group android:id="@+id/demo_grupo_listados"
        app:collapsible="true" app:collapsed="true">
         <GroupHeader  android:id="@+id/demo_head_apps"
                android:title="Apps,shortcuts..."/>
                <Text  android:id=”@+id/screen_1”….. />
                <Separator/>
                <Text  android:id=”@+id/screen_2”….. />

  </Group>
```

Look into the demo app (remember that it uses menu_grx_demo.xml instead of menu_grx_nav.xml) to see different ways of creating menu items and groups items.

7.- Opening the app from mods

From your mods you can easily open a screen, sub-screen and even simulate a click on a preference through intents with predefined intent extras strings

To do so, just create an new Intent, set it with the classname of this app / naub activity (com.mods.grx.settings / com.mods.grx.settings.GrxSettingsActivity) , add the following flags to the intent: FLAG_ACTIVITY_REORDER_TO_FRONT , FLAG_ACTIVITY_CLEAR_TOP , FLAG_ACTIVITY_NEW_TASK and add the following extras to the intent:

- **GrxScreen** -> specify the screen to open. As said before, this will be the id name of the preferences screen to open, defined in the navigation panel menu
- **GrxSubScreen** -> subscreen, if you want to reach a nested screen inside the specified GrxScreen preference XML.
- **GrxKey** -> if you specify a GrxKey in the intent, an onItemClick will be simulated, so, the preference will be opened or will change its state (f.e. on checkboxes or switches..)

If you just want to open a screen, it is only necessary to specify GrxScreen
If you want to simulate an onItemClick but the preference it is in the main preferences screen (not in a nested screen) GrxSubScreen should be the same as GrxScreen.
If you want just to open a nested screen you do not need to add the extra GrxKey

Java Example

```
@Override
public void onClick(View view) {

   Intent intent = new Intent();
   intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
   intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
   intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
   intent.setClassName("com.mods.grx.settings","com.mods.grx.settings.GrxSettingsActivity");
   intent.putExtra("GrxScreen","demo_nested");
   intent.putExtra("GrxSubScreen","nested_1");
   intent.putExtra("GrxKey","selectsort_3");
   try{
            getContext().startActivity(intent);
   }catch (Exception e){
      Toast.makeText(getContext(),e.getMessage(),Toast.LENGTH_SHORT).show();
   }
}
```

This code will try to open the preference selectsort_3 (a GrxMultiAccess preference in the demo app) which is placed in the nested screen nested_1 inside the preference screen demo_nested.xml

8.- Group Keys and broadcasts

When a preference value is changed by the user the app allows you:

- To send up to two different broadcast (predefined actions can be changed)
- To change the value of a shared group key.

The idea is to save the amount needed resources in your mods as well as to make your modding easier. These features are oriented to communicate to your mods that one or several preferences have changed.

8.1.- Group Keys

Attribute (string) in preferences:

grxGkey

Example

grxGkey=”gk_ram_recents”

A group key is a additional attribute you can add to the preferences declaration. The idea is to add the same group key to a group of preferences involved in the same mod, but you will find easily other uses..When one of the preferences sharing the same group key is changed by the user the app will change this group key.

Suppose a mod for showing RAM info where you allow the user to change the background color, text size, text color, updating time, etc.. You could do an observer for those preferences. Adding the same group key to those preferences you will just need to add to the observer the group key, so, when the group key changes that means that one of the preferences involved in the mod has changed. Then you can read all these preferences an apply the changes to the mod.

There are many ways of doing mods, this will allow you to add new ways. Following with the example, you could store the preferences values in a public class in static fields. Every time the group key changes, you could read the preferences for this mod, store their values in that class and then to apply the changes when needed. Or to apply the changes immediately but next time you do not need to read again the preferences, since they are available in the public class.

The use of this group keys allows you to reduce the needed resources to observe keys.

The stored values for group keys is of type int. The app increase its value every time a preference with the group key is changed from 1 to 32. When its value is 32, next change will reset the value to 1 again.

8.2.- Broadcasts

Up to two broadcast can be sent when a preference value changes. This is useful when you want to react to preference changes if you have previously registered the action in an existing receiver.

Attributes

grxBc1

grxBc2

Actions defined in strings.xml

gs_grxBc1

gs_grxBc2

Default values (bools.xml)

def_grxBc1

def_grxBc2

Example of Use in preferences

grxBc1=”true”

grxBc2=”false”

The actions corresponding to grxBc1 and grxBc2 are defined in the indicated strings inside strings.xml. The default values are defined in bools.xml. If you want, for example, the app to send a broadcast with the action defined in gs_grxBc1 you have to specify in the preference grxBc1=”true”. Be careful with the default values in these attributes. If you design your mods in such a way that you want the app always to send a broadcast, then define a true default value in def_grxBc1 or def_grxBc2, so will have less work in your preferences declarations.

I have been using this way for some years, making the most of existing receivers in systemui, services.jar or android.policy.jar. And I will do for some mods.

I always try to optimize the code of my mods, looking for a balance between features/battery drain. In my case I prefer to spend more cpu cycles in the user process of configuring the mod than when the mod is executing.

Anyway, this feature together with the group keys allows you to design each mod in the best possible way. In addition to the traditional way of reading user preferences you now have two new ways that allow you to make each mod as optimized as possible

9.- Stored values to System Settings and Strings Separators

As I said before it is a shame that on LP+ there is no way of to read concurrently reliably the preferences stored in an app sharedpreferences. So we cannot create in our mods a package context to access this app´s sharedpreferences. We can, but it will not work properly, sometimes it is ok sometimes nop. And it is a shame because I used to develop my mods in that way, making the most of the sharedpreferences type of stored values in it (sets, Boolean, ..)

Due to this, the app is designed to synchronize its internal shared preferences with the settings system db (now an xml, by the way).

9.1.- Global bool for Enabling to save values in settings system

In bools.xml you will find **enable_settingsdb**

Set it to true if you want to save your preferences values in system settings. I left that bool to false in the source code and in the demo app, so change it when you are writing your mods.

9.2.- Enabling in the preferences the saving of their values in settings system

You can specify in the preferences the **attribute grxCr**

if grxCr is true in a preference (grxCr="true") its value will be saved and synchronized in settings system

De default value for grxCr is specified in **bools.xml -> def_grxCr**

So, play with the enable_settingsdb, grxCr attributes and def_grxCr when you configure the app.

If def_grxCr is false you should have to explicitly specify you want the value to be saved in settings system the attribute grxCr="true"

If def_grxCr is true (normal use) you do not need to specify the attribute on each preference for their values to be saved. **But** remember, if the Boolean **enable_settingsdb** is false no value will be saved to settings system, even if def_grxCr is true or if you specify grxCr="true" on your preferences.

I have done this for allowing you not to mess your settings system while configuring your preferences.

The normal configuration should be enable_settingsdb = "true" and def_grxCr="true", specifying grxCr="false" just in those preferences you do not want to save their values in settings system.

I recommend you to play with the app to now how it works with enable_settingsdb="false" and def_grxCr="true" , system settings will not be messed while you configure your preferences. Once you have designed your preferences screens, change enable_setttingsdb to true. In this way you do not need to specify grxCr="true" on every preference.

9.2.- Type of stored values in settings and multivalue separators

- int : for preferences based on integer or Boolean values

- Strings : for preferences based on strings or preferences managing multivalues

Multivalues

As said they are allways stored in settings system (also in sharedpreferences) in strings.

The multivalue string always have the same format **(even for just one value)**

<value1><separator><value2><separator>

So in your mod you will split these strings to get an array of strings.

When you split a string it is a good practice to use Patter.quote(<string_separator>) to avoid problems

DEFAULT SEPARATOR

You can specify the default separator used by the app in strings.xml -> **gs_def_sep**

I have chosen the "|" as default separator. This is a very handy separator because in your smalies when splitting the string you do not need Pattern.quote, but simply scaping the | will work.

const-string v0, "\\|"

For other separator I recommend again the use of Pattern.quote

SEPARATOR ATTRIBUTE

In the multivalues preferences declarations you can specify a different separator by the use of the attribute **grxSep**

For example grxSep="="

If you do not specify grxSep, gs_def_sep will be used.

Pay attention on not to change the separator when the preference value is already stored or you will get problems!. If you need to change the separator in this circumstance you should use another preference key.

10.- Customizable Dependencies

Standard android dependencies are very limited: just check true, false (or 0, not 0..)… and are limited to the same preference screen.

A very simple system of configurable dependencies has been added to the program. This system works with basic rules. Rules are added to the preferences declaration through the attribute

grxDepRule

(with null namespace, as every added attribute to the preferences in this program) .

A dependency rule consist in the following **notation**:

result#dependency_key#type_of_value_of_the_dependency_key#vavlues to check

Result:	ENABLE or DISABLE
Dependency key	Key to check for enabling o disabling the preference with this rule
Type of Value:	Type of value of the dependency key: INT , STRING , BOOLEAN

I have added the BOOLEAN option to provide dependencies among different preferences screens (without app FCs, like it happens with standard dependencies)

Values to check:

One or more possible values (Separated by commas) of the dependency keys to be checked in order to enable or disable the preference.

Rules allows you to check exact values as well as partial values. To specify a partial value to check, write it between parenthesis **(value)**

Empty or null values **.** Many of the preferences designed for this application store their values as strings. In these cases the application for null values stores an empty value. To include the empty value in the rules use NULL

To be taken into account

- The parameters of the rule are separated with #
- If you just use one value to check, do not use ",".
- If you use a given key in more than one preference screen, add the rule to the preference in every place, if you want a coherent operation.
- If you use customizable dependencies, do not use standard dependency attributes in the involved preferences!!!.
- You can use uppercase or lowercase to specify the result of the rule and for the type of value,
- You must be exact when you use the NULL label (which must be capitalized).
- TRUE and FALSE are used with Boolean values (uppercase or lowercase)
- You must be exact with the name of the key and with the values (except for Boolean values as indicated above).

-

Example 1:

….. android:key=”seekbar_1” …..

Android:key=”key_b” grxDepRule=”ENABLE#seekbar_1#int#200,400,401”

Key_b will be disabled except when seekbar_1 value(which stored an int value) is one of the following values :200 or 400 or 401.

Example 2

android:key=”key_a” grxDepRule=”disable#key_c#string#NULL,5,(com.mods.grx.ajustes;),(56)

key_a will be disabled when key_c (which is stored as string) is empty, or its value is exactly the string 5 or contains the string com.mods.grx.ajustes; or contains the string 56

Example 3

android:key=”key_a” grxDepRule=”disable#key_d#boolean#true

key_a will be disabled when key_d, which is Boolean (switch or checkbox) is true.

11.- Type of preferences and their attributes

The standard PreferenceScreen is a final class so we cannot create a new preference based on it. Use the standard android attributes in PreferenceScreen

All the rest of preferences used in this app can be declared in the xml using their short class names. For example < GrxCheckBoxPreference …. /> instead of <android.preference.GrxCheckboxPreference.

To specify grx attributes do not use any name space!

11.1.- GrxCheckBoxPreference and GrxSwitchPreference

- Standard attributes:
    - android:defaultValue, android:icon, android:key, android:summaryOn, android:summaryOff, android:summary ..
- grx Attributes :
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)

- Stored value in system settings:

Integer value, so through content resolver read the int value . 1 -> checked, 0 -> not checked

11.2.- GrxPreferenceCategory

- Standard android Attributes: With appcompat just android:title and android:key seems to work, but you can use any standard one.

- grx Attributes
    - grxDepRule
    - grxTextColor (grxTextColor="#ffffff") Use it if you want to change de default text color.
    - grxBackGroundColor , use it if you want to change the default background color
    - grxHide (="true" or "false") Use it if you want to hide the category, not showing any text and without taking up screen space. This is usefull for wrapping other preferences applying standard o customized dependencies on them.

Of course, no stored value here

11.3.- GrxInfoText

It allows you to show text. For now the android:summary text is used.

- Standard attributes: android:icon, android:summar (the text here)
- grx Attributes

  grxDepRule

11.4.- GrxSeekBar

A seekbar preference.

- Standard attributes:
  - android:title, android:key, android:summary and android:defaultValue (use this to set de default value, an int value, of course)
- grx Attributes:
  - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
  - grxMax -> max value (int)
  - grxMin -> min value (int)
  - grxUni -> units, string
  - grxInter -> interval, int. Default value is 1. You can use this attribute to allow just certain values.
  - grxPopup -> true, false . By default true. It will show a popup while the user changes the pref value. If no summary is specified in the pref, the popup area will be reduced to make easier the access to the scrubber.
- Stored value: integer

11.5.- GrxNumberPicker

- - Standard attributes:

- android:icon, android:widgetLayout, android:title, android:key, android:summary, android:defaultValue (integer)

- grx attributes:
  - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
  - grxMin, grxMax and grxUni , same as GrxSeekBar
- Stored value: int

11.6.- GrxEditText

Edit text dialog

- Standard attributes:
  - android:icon, android:widgetLayout, android:title, android:key, android:summary, android:defaultValue (string)

- grx attributes:
  - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
- Stored value: String

11.7.- GrxOpenActivity

To open activities from the app

- Standard attributes:
  - android:icon, android:widgetLayout, android:title, android:key, android:summary (string)

- grx attributes:
  - grxDeprule (already explained)
  - grxActivity

    String, format pacakgename/activity name

    Example:

    com.android.settings/com.android.settings.Settings$WifiSettingsActivity

when the user click on the preference wifi settings will be opened. If the activity is not installed the preference is disabled.

11.8.- GrxOpenIntent

To open intents from the app

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary (string)

- grx attributes:
    - grxDeprule (already explained)

The intent is defined adding to the pref definition the intent tags. Example

```
<GrxOpenIntent android:icon="@drawable/demo_icon_2" android:title="Esp Desarrolladores"
android:summary="Click here to open">
        <intent
        android:action="android.intent.action.VIEW"
        android:data="http://www.esp-desarrolladores.com"
        />
</GrxOpenIntent>
```

When this pref is clicked will open the web page esp-desarrolladores.com

11.9.- GrxSingleSelection

To select one item in a list of items.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (string)

- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxA_entries -> array of entries (titles) in the selection list. (f.e. grxA_entries="@array/options_array_titles")
    - grxA_values -> array with the corresponding values
    - grxA_ics -> drawable array of icons (optional, just if you want to show icons in your list). If this attribute is used, the corresponding icon will be showed when a value is selected

- Stored value: String with the selected value (from grxA_values)

11.10.- GrxMultipleSelection

To select several item in a list of items.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (string)

- grx attributes:
    - Same attributes used in GrxSingleSelection and in addition:.
        - grxSep , optional, if you want to use a specific separator
        - grxMax , optional, max number of selectable values. If not specified or 0 all the values can be selected.

- Stored value: String with the selected values (from grxA_values) separated by the separatos (f.ex. -> 3|4|)

Remember that if you specify a default value, you should use the following format

<value1><separator><value2><separator>..

11.11.- GrxSortList

A preference to sort a list of items

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (string with values separated by the separator!!)

- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxSep , optional, if you want to use a specific separator
    - grxA_entries -> array of entries (titles)
    - grxA_values -> array with the corresponding values
    - grxA_ics -> drawable array of icons (optional)
    - grxSortIcon -> true if you want to show a sort icon.

  grxA_entries and grxA_values are mandatory, of course.
- Stored value: String with the sorted values separated by the used separator.

11.12.- GrxSelectSortItems

A preference to select items and sort them.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (string with values separated by the separator!!)
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxSep , optional, if you want to use a specific separator
    - grxA_entries -> array of entries (titles), mandatory
    - grxA_values -> array with the corresponding values , mandatory
    - grxA_ics -> drawable array of icons (optional)
    - grxMax , optional, max number of selectable values. If not specified or 0 all the values can be selected.

- Stored value: String with the sorted values separated by the used separator.

11.13.- GrxDatePicker

A picker dialog to select a date.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (specify a default value with the following format <day>/<month>/<year>)
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)

- Stored value: String with the following format

  Day/month/year

11.14.- GrxTimePicker

A time picker dialog to select a date. In landscape orientation the picker is not showed in the right way. This is because I am using the app compat time picker and is a reported bug…

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue** (specify a default value in integer format)
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)

- Stored value: integer

  The time is stored in minutes = (hour*60) + minutes

11.15.- GrxColorPicker

Color picker.

- Standard attributes:
    - android:icon, android:title, android:key, android:summary
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxDefColor -> default color for this prefererence. Use smali format

      grxDefColor=”0xffffffff” this is the same than #FFFFFFFF

If you do not specify a default color, the color defined in integers.xml -> def_grxDefColor will be used

- grxAuto -> if true, an additional button will be showed in the color picker, allowing to the user to generate a suggested colors palette from an image.

  The default value for this param is defined in bools.xml -> def_grxColorAuto
- grxAlpha -> if true the alpha slider is showed.

  The default value for this attribute is defined in bools.xml -> def_grxAlpha
- grxFlower -> if true the color picker selector will seem as a flower. If false it will be a circle.

  The default value por this attribute is defined in bools.xml -> def_grxFlower

- Stored value: an int value, ready to use

11.16.- GrxMultiAppColor

A preference to link apps with colors

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , **android:defaultValue**

- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)

- grxSep , optional, if you want to use a specific separator
- grxAllApps -> default false, if true all the installed apps are showed. If false just user apps.
- grxColor -> default true. If false, neither color selector will be used nor you are using this preference to link apps and colors. You also can use this app like an multiple app selector by setting this attribute to false. If true, in addition you can use the following attributes
  - grxDefColor, grxAuto, grxFlower and grxAlpha as indicated in GrxColorPicker
- grxSaveActName -> default is false. If true the activity name will be saved.

- Stored value: a String with selected values separated by the separator.

  Output Format:

  You have to do at leas two splits. The first one if for to know how many elements are stored (with separator).

  The on each element of the array and depending on the preference configuration you could find the following info:

  - Apps and colors

  <package name>=<color string><separator><package name 2>….

So, you have split again with "= " and to get the color just use Integer.valueOf(resultin_array[1]) in the resulting_array[0] you will get the package name.

- Apps and activity name and colors

  <package name>/<activity_name>=<color string><separator><package name 2>/<activity name2>=<color>

  The same as before, but if you want to process something with the activity name you have to split with / the first element in the above case.

If you do not use colors the output format for each selected element will be:

- <package name><separator><package name2><separator> in the case of you are not saving the activity name or
- <package name>/<activity name ><separator><package name2><activity name 2><separator>… if you use the activity name attribute.

11.17.- GrxPickImage

An image picker

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxSizeX and grxSizeY , optionals, resulting width and height in pixels (just write the number in this attributes). A proporcional cropper will be show while selecting the image
    - grxCircular , optional, default false, if true the resulting image will be rounded
- Stored value: a String representing the uri of the content.

If you specify grxSizeX and grxSizeY a new image will be saved on the icons directory of the app. If not specified, then it is supposed that a image from the gallery is going to be used in your mod.

Take into account that for now there is no EXIF information processed in this app. So you should tell your users to use screen shots or gallery images in the right orientation!.

If an cropped image is saved the orientation will be ok.

The easier way to get the bitmap is:

```
InputStream inputStream = context.getContentResolver().openInputStream(img_uri);
Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
```

11.18.- GrxSelectApp

A simple application selector.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , android:defaultValue
- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxAllApps -> default false. If true all installed apps will be selectable.
- Stored value: String, package name.

11.19.- GrxAccess

This is a multipurpose flexible preference. You can select an app shortcut (direct call, a chat in whatsapp, a specific contact..) , an app, an activity or a customized option.

Customized options are designed for developers. It is the same concept as in GrxSingleSelection, GrxMultipleSelection or in GrxSelectandSort : the developer design a list of options with their corresponding values (and icons, optional, of course).

The stored value is the string representation of an intent uri with extra values. So, for example, in the case you want to use this preference in a mod that launches a shortcuts, apps or activities you can use directly the value.

To create an intent from the stored value use

Intent.parseUri(STRING_VALUE,0)

And the you can launch the intent. That is all.

If you want to enrich your mod by using some of the extras added it will be easier to be processed through the intent.

If you add customized options, you also will get an string representation of the intent.

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , android:defaultValue (string representing the intent..uff..but possible)

- grx attributes:
    - grxCr, grxGkey, grxBc1, grxBc2, grxDeprule (already explained)
    - grxShc -> the default value is defined in bools.xml -> def_grxShc . If true in the selector will be show this option (select a shortcut)
    - grxApps - > the default value is defined in bools.xml -> def_grxApps . If true in the selector will be show this option (select an app)
    - grxAct -> the default value is defined in bools.xml -> def_grxAct . If true in the selector will be show this option (select an activity)
    - grxA_entries and grxA_values, if both are declared in the preference, then the option to select a customized option will be showed. In addition, of cours, you can add icons in this case by using grxA_ics. If you want to use the icon in your mod you can use the attribute grxSv_ics, setting it to true.

Notes:

The use of grxA_entries (titles), grxA_values (values) and grxA_ics (icons) is the same than in GrxSingleSelection, GrxMultipleSelection…

The shortcut icon is always saved. You can enrich your mods!.

The shortcut label is added to the saved value.

A formatted activity label is added to the saved value

- Output Value: string with its uri representation.

Intent.toUri(0);

<u>How to process</u>

1.- get the intent: Intent.parseUri(STRING_VALUE,0)

If you have defined the preference to select shortcuts, apps or activities in order to launch it from your mod, just launch the intent!

Extra values

2.- getting the extra info

a.- type of access

Int type = intent.getIntExtra("grx_type",-1);

- Types
    - 0 -> it correspond to a shortcut
    - 1 -> it correspond to an app
    - 2 -> it correspond to an activity
    - 3 -> it correspond to a customized option

b.- icon path

If you saved the customized icon or it is a shortcut you will find the path in:

String icon_path = intent.getStringExtra("grx_icon")

c.- access label

If the preference is of type shortcut , customized option or activity (formatted activity name, since there are many activities with no label) , the following extra will be stored.

String extra_label = intent.getStringExtra("grx_label")

d.- for customized options the name of the drawable icon, if used, is also stored. This may be usefull both to now very quickly what option was selected and to make easier to find the icon you need (if you use icons in your mods, of cours) .

String drawable_name= intent.getStringExtra("grx_drawable")

d.- for customized option you get the selected value (string) through the extra grx_value

String customized_value = intent.getStringExtra("grx_value")

In the case of apps and activities you will retrieve the icon in the usual ways.

Since it is complex to process this preference, **use the app code to do so!!**

Recommendation for smali mods

Decompile the app. You will find in com.mods.grx.settings.Utils everything you need to process this preference.

Use those methods to easily retrieve the icon and label, not worring about if it is the sd or you have to load them from the intent.

The same could be said for image processing, in com.mods.grx.settings.utils.GrxImageHelper you will find everything you need!!

11.20.- GrxMultiAccess

Multi GrxAccess selector

- Standard attributes:
    - android:icon, android:widgetLayout, android:title, android:key, android:summary , android:defaultValue (string representing the intent..uff..but possible)

- grx attributes: some attributes than in GrxAccess and in addition:
    - grxSep , optiona non default separator
    - grxMax , optional. If not specified or 0 (default value) then there will not be limit in the number of items.
- Stored value: GrxAccess values separated by the separator.

So, split the string and process each element as said in GrxAccess.

12.- Other configurable values

The app stores the backups an images used in preferences in the sd.

You can configure where to save this info.

Main dir : Strings.xml -> grx_dir_datos_app

Icons sub dir -> Strings.xml -> grx_ico_sub_dir

Backups extension -> String.xml ->gs_backup_ext

Backups are stored inside the main dir , in the folder backup. Inside this last folder, when the users backups the preferences, a new folder with the backup name is created to save the current images.

13.- How to quick start and Apktool.jar (2.0 .. +)

You can use apktool.jar with this app. In fact the demo apps I provide are signed with apktool.jar, so you will not get problems on using it.

To start using the app (both trough Android Studio or apktool.jar)

- Add a menu item to menu_grx_nav.xml (in res-menu)
- Create your preferences screen with the same file name than the specified id name in the menu item
- In bools.xml change the following values to fit your needs
    - DEMO
    - enable_settingsdb
    - def_grxCr
- Add your preferences to the new xml
- Configure the rom info tabs

and play with the app, adding preferences screens, broadcasts, group keys , changing the themes, etc..