GrxSettings-v2.1

(MPL 2.0 - LP 5.1+)

App for configuring mods

Developers Guide (Ag. 2020)

1.- Main Features

- 30 types of preferences which can be referenced using their short name. Some of them have several behaviours depending on added xml attributes.

- Three Dots Menú: Backup and restore preferences, Tools (restart UI, restart Phone, Go to Recovery), Reset all values to default.

- Root support. The app checks if it has been su granted when it needs it. So is not a must, except if you want to run some features.

- Navigation panel through preferences screens and groups of preferences screens, with nested screen support.

- Floating action button.

- Several themes available. Easy creation of new themes.

- Up to 7 actions can be run after a preference value change: reboot, restart an app, run a script (file, file+arguments,string-array commands), change a group key, send up to 2 custom broadcasts, send common broadcast with extras and to simulate an onclick event.

You can change the default order of execution of the actions by changing an app´s string, but you can override this order on every preference by adding an attribute.

- Customizable dependency rules. Enable or disable preferences based on other preference´s values, even if the preference is not in the same preference screen.

- Customizable Build Prop rules, allowing to hide/show individual preferences, preference screens or groups of preferences screens, based on build prop properties values.

- Full support for standard Secure, Global and System Settings.

- Support for easy rom info creation, through easily configurable sliding tabs.

- User options: floating area with access to last seen screens, several themes (if the dev allows it), 3 color pickers styles, image selection for navigation panel header background (if the dev allows it), divider height selection in list views, show/hide and position of the floating action button, navigation panel position, force expanded groups of preferences screens, remember screen, exit confirmation….

2.- Main Configuration Options - values-configapp.xml

- **grxs_app_name** : string for app name

- **grxb_demo_mode** : boolean. If true the app is configured in demo mode. Demo mode means that navigation panel will use the menu demo_navigation_screens.

When yo build your app set it to false, then the menu rom_navigation_screens will be used

- **grxb_global_enable_settingsdb**: boolean. This boolean is useful when you are building your screens. By default is set to false, this means that no value will be saved to Settings System, not messing your settings system values. Do not forget to set it to true, or your mods will not work!!
- **grxb_saveSettings_default** : boolean. This boolean sets the default value for the common atribute grx:saveSettings .

So, grx_global_enable_settingsdb is mandatory, meaning that if set to false no value will be saved to settings system, but grx:saveSettings is a common preference attribute and grxb_saveSettings_default is the default value for this attribute. If you set this default value to true you do not need to add the attribute in the preferences. Normal configuration for your app will be grxb_global_enable_settingsdb set to true and grxb_saveSettings_default set to true. You can play with these attributes while developing your mods and screen for not to mess too much your settings system.

- **grxb_allow_panelheader_changes**: boolean value. If you set it to false the user will not be able to add a custom header in navigation panel.
- **grxb_allow_theme_change**: boolean value which determines if the user will be able to change of theme. If you set to false, the user will not be able to change of theme.
- **grxb_allow_user_colorpicker_selection**: boolean value. If set to true the user will be able to select his/her favourite color picker style.
- **grxs_process_actions_order**: A string. This app allows the dev to execute up to 6 different actions when a preference value is changed (run a script, send broadcast, change a group key, reboot, kill and a app, send broadcast with actions..). This strings defines the default order of execution of these actions.
- **grxs_kill_command** : this is a string indicating to the app how to execute the app kill action. Depending on your rom / environment you may need to change it.
- **grxs_data_base_folder** : string. This value defines the base folder for the user data managed by this app: icons, backups, etc. The default folder is placed in the internal sd.
- **grxs_backups_files_extension** : string defining the backups files name extensión.
- **grxi_default_list_divider_height** : integer. Default divider height in lists.
- **grxs_default_theme** : string defining the default theme for the app. This is the style name you find in sytles.xml.
- **grxi_floatingrecents_max_items_seen**: integer defining max number of seen screens links in the quick Access floating área window.
- **grxi_floatingrecents_total_items**: max number of last visited screens to manage.

- **grxi_default_colorPicker_color**: default color for color pickers when we do not define a default color
- **grxs_separator_default**: default separator value in multivalues preferences
- **grxs_colorPickerStyle_default**: default style for color pickers (circle, square, flower)
- **grxb_remember_screen_default**: Boolean. This is the default value of the user checkbox to remember last screen, in the user navigation panel options.

3.- NameSpace declaration for using grx attributes in preferences screens

- Added preferences attributes use the name space “grx”.

- Add the following to your new preferences screens:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:grx="http://schemas.android.com/apk/res-auto">
```

4.- Preferences

- **GrxCheckBoxPreference**:
    - same standard attributes than CheckBoxPreference.
    - grx: checkboxColor : Checkbox color. I recommend you to use a reference attribute added to styles.

  See demo_check_swtich_file.xml for examples

- **GrxSwitchPreference:**
    - same standard attributes than SwitchPreference.
    - grx: switchColor: Switch color. I recommend you to use a reference attribute added to styles.

  See demo_check_swtich_file.xml for examples

- **GrxFilePreference:** Extends GrxSwitchPreference, instead of creating a preference key in settings, it creates an empty file. Same attributes than GrxSwitchPreferences.

  See demo_check_swtich_file.xml for examples

- **GrxPreferenceCategory:** Extends standard PreferenceCategory.
    - Same standard attributes than PreferenceCategory.
    - grx: hiddenCategory: If set to true, the category text is not shown. Useful for wrapping other preferences you want to enable – disable based in dependency (standard or based in rules)
    - grx: textColor: Category text color. I recommend you the use of reference attributes added to styles.
    - grx:backgroundColor: category text background color. I recommend you the use of reference attributes added to styles.
    - grx: centerHorizontal: if it is set to true the text is cantered horizontally.

  You can see all this new attributes working through the demo_... .xml files.

- **GrxInfoText:** This preference does not store any value in settings. It is useful to show info text to the user.
    - android:summary : The text to show , you can also use android:icon to show an icon on left.
    - grx: rightIcon: drawable reference to show an additional icon on the right side.

- Grx: rightIconTint: tint color for the right icon. I recommend you the use of reference attributes added to styles.

- **GrxDatePicker:** Date Picker preference.

  - **Format:** The preference manages string values with the format dd/mm/yyyy
  - android:title, android:summary, android:defaultValue

  Have a look to demo_time_date.xml

- **GrxTimePicker:** A time picker preference.
  - **Format**: This preference manages int values. The int value represent the number of minutes in a 24 hrs basis. For example, a value of 128 means 02:08 (AM).
  - android:title, android:summary, android:defaultValue

  Have a look to demo_time_date.xml

- **GrxColorPicker:** Color picker, 3 styles available.

  - **Format:** Int value.
  - android:title, android:summary
  - android:defaultValue I recommend you to use the format 0xffffffff to define colors in this app. If this attribute is not defined in the preference, the default value is taken, as said before, from the value defined in the integer grxi_default_colorPicker_color, in values-configapp.xml.

  - grx: colorPickerStyle : You can set this attribute to circle, square or flower. If you do not add this attribute the default style defined in the string grxs_colorPickerStyle_default inside values-configapp.xml is taken.

  - grx: showAlphaSlider , true or false, shows or not the alpha slider. The default value of this attribute is defined in the boolean grxb_showAlphaSlider_default (values-configapp.xml)

  - grx: showAutoButton , true or false. If true, in the color picker an additional button is added. When the user press this button, an auto color palette is generated based on an image selected by the user. The default value for this attribute is set in the Boolean grxb_showAutoButton_default in values-configapp.xml

  - grx: saveValueOnFly , default false. If true the value will be saved on fly (while the user change the color).

  Have a look to demo_colorpicker.xml

- **GrxAppSelection** . This is an app selector. It allows to save only the package name or packagename/activityname

    - android:title, android:summary
    - grx: saveActivityname , true or false. If true the activity name will be saved. The default value for this attribute is defined in the boolean grxb_saveActivityname_default that you will find in values-configapp.xml

    - **Format:** As said it depends on the grx:saveActivityname value. The resulting string will be either the package namer or package name/activity name

    - grx: showSystemapps , true or false, if true system apps are shown in the selector. The default value for this attribute is set in the Boolean grxb_showSystemapps_default in values-configapp.xml

    - android:defaultValue , be careful, this value should have the right format depending on the saveActivityname attribute.

  Have a look to demo_selectapp.xml

- **GrxMultipleAppSelection .** This preference allows to select several apps.

    - android:title, android:summary
    - grx: saveActivityname, grx: showSystemapps , same than in GrxAppSelection
    - grx: maxChoices, an int value. If 0 (default) no limit.
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.
    - **Format** This pref uses a string value. Depending on the separator and the saveActivityname attribute. F.e. “com.android.setting|com.grx.settings|” when we do not save the activity name
    - android:defaultValue, string with the format according to the separator and saveActivityname attributes

  Have a look to demo_selectapp.xml

- **GrxEditText** , A preference to edit text (string)
    - android:title, android:summary, android:defaultValue

  Have a look in demo_num_text_seekbar.xml

- **GrxNumberPicker,** Number picker. Int value.

    - android:title, android:summary, android:defaultValue

- grx:minValue, an int value representing the min value shown in the number picker.
- grx:maxValue, an int value representing the max value shown in the number picker
- grx:units, units

Have a look in demo_num_text_seekbar.xml

- **GrxSeekBar**, Seekbar preference. It works with int values

  - android:title, android:summary, android:defaultValue
  - grx:showPopup , true or false. If true a popup is shown with the current value. The default value is true.
  - grx:minValue, an int value representing the min value possible in the seekbar.
  - grx:maxValue, an int value representing the min value possible in the seekbar
  - grx:units, units
  - grx: interval , interval taken when moving the seekbar. Default value is 0.
  - grx: seekbarColor, optional color attribute for the seekbar. I recommend you to use a referenced attribute valued added to the styles.

  Have a look in demo_num_text_seekbar.xml

- **GrxOpenIntent** , Open intents when the user clicks on it. Useful for example to open Web pages. This preference does not save any value to settings system.

  - android:title, android:summary, android:defaultValue

  <GrxOpenIntent android:title …>

  <intent android:action="android.intent.action.VIEW"
  android:data="https://www.espdroids.com" />

  </GrxOpenIntent>

  The previous example would open the web www.espdroids.com

Have a look to demo_open_intent_activity.xml

- **GrxOpenActivity** , Launch an activity when the user clicks on it. This preference does not save any value to settings system.

  - android:title, android:summary,
  - grx:activity . This attribute sets the activity to be launched. You can launch an app (just adding the package name) or an specific activity using the format packagename/activityname . If the package or the package/activity does not exists in that moment, the preference is disabled

- **GrxSingleSelection** , Use this preference instead of the standard ListPreference. Given a list of items this preference allows to select one of them. It saves a string value, as ListPreference, but you still can use the value as an integer, if your options values have got that format.

    - android:title, android:summary, android:defaultValue
    - grx:optionsArray , the array reference for the options titles.
    - grx: valuesArray, the array reference for the values.
    - grx: iconsArray, optional, you can set an array of drawables that will be shown in the dialog.
    - grx: iconsValueTint, optional if you are using iconsArray attribute. The icons will be tinted with this color. As always, I recommend you to use a reference attribute defined in the themes styles definition.

  Have a look to demo_select_items.xml

- **GrxMultipleSelection**, this preference is the one you will use instead of the standard MultiSelectPreference. This preference saves a string value with the selected items separated by the defined separator.
  It uses the same attributes than GrxSingleSelection and these additional:

    - android:title, android:summary,
    - grx: maxChoices, an int value. If 0 (default) no limit.
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.
    - android:defaultValue, set your default value according to the separator you selected. The default value should finish with this separator to make it easier to process your mods.

Have a look to demo_select_items.xml

- **GrxSortList** , this preferences allows to sort a list of items out. The value of this preference is of type String.

    - android:title, android:summary,
    - grx:optionsArray , the array reference for the options titles.
    - grx: valuesArray, the array reference for the values.
    - grx: iconsArray, optional, you can set an array of drawables that will be shown in the dialog.
    - grx: iconsValueTint, optional if you are using iconsArray attribute. The icons will be tinted with this color. As always, I recommend you to use a reference attribute defined in the themes styles definition
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.

- android:defaultValue, set your default value according to the separator you selected. The default value should finish with this separator to make it easier to process your mods.

Have a look to demo_sort_items.xml

- **GrxSelectSortItems** . This preference is of type String. It allows to select items from a list and to sort the selection out.

  - android:title, android:summary,
  - grx: maxChoices, Max number of selectable items, an int value. If 0 (default) no limit.
  - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.
  - grx:optionsArray , the array reference for the options titles.
  - grx: valuesArray, the array reference for the values.
  - grx: iconsArray, optional, you can set an array of drawables that will be shown in the dialog.
  - grx: iconsValueTint, optional if you are using iconsArray attribute. The icons will be tinted with this color. As always, I recommend you to use a reference attribute defined in the themes styles definition
  - android:defaultValue, set your default value according to the separator you selected. The default value should finish with this separator to make it easier to process your mods.

  Have a look to demo_select_sort.xml

- **GrxPickImage** , an image picker preference. Images are saved to the path formed through the above explained options in values-configapp.xml (grxs_data_base_folder and grxs_data_icons_subfolder).

  - android:title, android:summary,
  - grx:sizeX and grx:sizeY , optional, the saved image will be resized to the values (int value, pixels) defined in these 2 params.
  - grx: circularImage , set this attribute with value true if you want the image to be processed in order to be saved as a circular image.

  - **Saved Value,** This preference save a string value representing either the path to the processed image (if you added size / circular attributed) or a string representing an uri (no processed images).

  New security implemented in nougat and Oreo related to sd card might give you many headaches while retrieving the images from your mods, depending from where

you are running the mod. Since this new security policies seems to be settled, in the future I will add this app a File Provider.

Have a look to demo_image_picker.xml

- **GrxPerItemColor** . This preference allows to link items values with colors. It is a preference of type string.
    - android:title, android:summary
    - grx:optionsArray , the array reference for the options titles.
    - grx: valuesArray, the array reference for the values.
    - grx: iconsArray, optional, you can set an array of drawables that will be shown in the dialog.
    - grx: iconsValueTint, optional if you are using iconsArray attribute. The icons will be tinted with this color. As always, I recommend you to use a reference attribute defined in the themes styles definition
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator. This is the separator for the <item/color> values.
    -

      The default colors of the items can be set in two ways:

        - grx:defaultColor , same default color for all items
        - grx:colorsArray , an array of colors

      Format: The format of the saved value is
      Item_value/color<separator>item2_value/colorforitem2<sep….

      The color in the string value is a string with the exact int representation of the color value

  Have a look to demo_items_color.xml

- **GrxPerItemSingleSelection**. This preferernce allows to link items – values and to sort them out.

    - android:title, android:summary
    - grx:optionsArray , the array reference for the options titles.
    - grx: valuesArray, the array reference for the values.
    - grx: iconsArray, optional, you can set an array of drawables that will be shown in the dialog.
    - grx: iconsValueTint, optional if you are using iconsArray attribute. The icons will be tinted with this color. As always, I recommend you to use a reference attribute defined in the themes styles definition
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator. This is the separator for the <item;value> values.

- grx: spinnerOptionsArray , array of the single selection titles
- grx: spinnerValuesArray , array of the single selection values
- grx: allowSortOut , if this attribute is set to true, the user will be able to sort the list out. The default value of this attribute is false

Format:

- value1;selection<separator>…

- android:defaultValue, set your default value according to the separator you selected. The default value should finish with this separator to make it easier to process your mods.

Have a look to demo_peritem_single_sel.xml

- **GrxPerAppColor.** This preference allows to link apps and colors.

  - android:title, android:summary
  - grx: showSystemapps , true or false, if true system apps are shown in the selector. The default value for this attribute is set in the Boolean grxb_showSystemapps_default in values-configapp.xml
  - grx: maxChoices, max number of apps to select, an int value. If 0 (default) no limit.
  - grx:defaultColor , default color when a new app is selected
  - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator. This is the separator for the <item;value> values.

  Format. The output format of the string is:

  Package_name=color<separator>….

  The color in the output string is a string representation of the color int value.

  Have a look to demo_multiappcolor.xml

- **GrxLedPulse** . This is a preference to help the devs to make mods on the phone´s led. It allows to select a led color, and the led on time and led off time. This preference generates a notification. This notifications could be used by devs to add on-fly support for color selection. Anyway, in most of the devices while configuring this preference, the led will take the current values when the screen is off (because of the generated notification).

  - android:title, android:summary

- grx: showPulseOptions . Default value is true. Set it to false if you do not want pulse options to show

Format: The output format of this preference is:

a.- if pulse options are shown

#RGBcolor;pulse_on_time;pulse_of_time

Pulse times in miliseconds. The stored value is the exact representation of these values.

b.- if pulse options are not shown.

#RGBcolor (for example #ff3300)

- android:defaultValue. Set the default value according to the expected format depending on the used attributes.

The arrays for times on and off of this preference are configured in values-grxsettings xmls . You can modify the existing values in the following arrays: grxa_ledpulse_ton_options, grxa_ledpulse_ton_values, grxa_ledpulse_toff_options, grxa_ledpulse_toff_values

Have a look to demo_color_pulse.xml

- **GrxPerAppLedPulse.** This preference allows to link an app and a led configuration (color, time on and time off). Pulse options are always shown.
-
  - android:title, android:summary
  - grx: maxChoices, max number of apps to select, an int value. If 0 (default) no limit.
  - grx: showSystemapps , true or false, if true system apps are shown in the selector. The default value for this attribute is set in the Boolean grxb_showSystemapps_default in values-configapp.xml
  - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator. This is the separator among individual values (color;timeon;timeoff)

Format: The output of this preference is a string with the following format

#COLOR1;t1on;t1off<separator>#COLOR2…

- android:defaultValue. Set your default value according to the expected format depending on the used attributes.

Have a look to demo_color_pulse.xml

- **GrxSingleWidget** . This preference allows the selection of a widget. Since in some manufactures depending on the OS version is not possible to retrieve some installed widgets, this preference also allows to add additional widgets through arrays.

    - android:title, android:summary
    - grx: widgetsArray , an optional array with widgets not shown by the preference, but you want show.

    Format: The output value of this preference is a string with the following format

    Packagename/widget

    Example:
    com.sec.android.daemonapp/com.sec.android.daemonapp.appwidget.WeatherAppWidget2x1

    - android:defaultValue, set the default value (optional) with the explained format.

      The items in the optional array array of widgets will have the same above format.

  Have a look to demo_widgets.xml

- **GrxMultipleWidgets**. This preference allows to select multiple widgets.

    - android:title, android:summary
    - grx: maxChoices, an int value. If 0 (default) no limit. Max number of selectable items
    - grx: widgetsArray , an optional array with widgets not shown by the preference, but you want show.
    - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.

      Format: The string of this preference value has got the following format:

      Packagename/widget<separator>….

  Have a look to demo_widgets.xml

- **GrxAccess** . This is a multiuse preference. It allows the selection of apps, activities, shortcuts or custom values based in customizable list of items. The result of this preference is a string, representing an uri with extras values.

    - android:title, android:summary
    - grx:showShortcuts , if this attribute is set to true, the option to select a shortcut will be shown (a shortcut can be a direct call, direct message, etc.). The default value for this attribute is set in the Boolean grxb_showShortcuts_default in values-configapp.xml
    - grx:showApplications, if this attribute is set to true, the option to select an application is shown. The default value for this attribute is set in the Boolean grxb_showApplications_default in values-configapp.xml
    - grx:showActivities, if this attribute is set to true, the option to select an application is shown. The default value for this attribute is set in the Boolean grxb_showActivities_default in values-configapp.xml

      You can also to show a list of items by adding this attributes

    - grx:optionsArray, grx:valuesArray the arrays of options and values.

      Additionally you could add for the custom options arrays the following attributes

    - grx:iconsArray (array of icons), grx:iconsValueTint (tint color for the array of icons) , grx:saveActionsicons (if set to true, the icon will be saved to the data-icons configured folder).

    Format: The saved string contains the uri string of an intent with the following extra values:

        - grx_type : 0 -> shortcut , 1 -> app, 2 -> activitie, 3 -> custom action
        - grx_label: a string with the label of the selected item. The language of the saved label will be the language configured in the moment the value was saved. Depending on the type of access this label will correspond to the shortcut label, app label, activity label or the custom option label. For apps and activities you should retrieve the label in the normal way.

        - grx_value, for values corresponding to custom actions, you can get the selected value from this extra
        - grx_icon: This extra will provide you the path for the icon associated to the selected option. If it is a custom action and you enabled the saving of icons you will get the path for the saved icon. For values corresponding to shortcuts the icon is always saved. There is no need

to save apps or activity icons, you can easily retrieve them in the normal way.

- grx_drawable, the name of the drawable associated to the custom option selected. This extra could help you to decide in your mod which icon to show, for example.

If your mod need to launch the selected option in the cases of shortcuts, apps and activities you only need to create an intent from the saved value. If you need the icons or the label (for shortcuts / custom actions) , use the extras values.

Have a look to demo_access.xml

- **GrxMultiAccess**. This preference allows to select multiple GrxAccess. So, all the attributes of GrxAccess will be used and additionally we can use the following:

  - grx_maxChoices, number max of GrxAccess selected. If 0, no limit.
  - grx: separator , string defining the separator of the values. The default value is set in the string grxs_separator_default in values-configapp.xml. I set this value | as separator.

  The output format is a string containing uri strings representation of GrxAccess values separated by the selected separator.

  Have a look to demo_access.xml

- **GrxButtonPreference.** This preference does not save any value. This preference is usefull for executing one or several actions. You can run any of the available actions that can be run when the value of a normal preference is changed. So, you can run a script, to change a group key value, to send broadcast, … These actions will be explained in a specific section of this doc.

  - android:title, android:summary
  - grx:button_Style, set this attribute value to small, big or none.
  - grx: button_Text, Button text. If this attribute is not set, the title is taken.
  - grx: button_textColor , you can set the button text color through this attribute.
  - grx: button_backgroundColor, the default background color is the colorAccent defined in the current theme. By using this attribute you can change this color.

As always I recommend you to use attributes referencing a color in your themes.

Have a look to demo_buttons.xml to see the different configurations of this preference.

5.- Other styling attributes for preferences

In the previous section was shown specific styling attributes for some preferences (checkbox and switch colors, seekbars colors, info right icon color, etc..)

There are two more common attributes that can be applied to most of the preferences.

If you add the standard attribute android:icon you can tint it by using **grx:iconTint** .

You can add in many preferences **grx:arrowColor .** If you add this attribute, an arrow will be shown on the right side (left side if in RTL config) with the specified color.

It is highly recommended to set this colors in preferences by using an attribute referencing a color, as I did in most of the demo examples. You should do this if you allow to your users to select different themes or it will be really difficult to find colors complementing main themes colors.

6.- Actions after a preference Change

You can add some attributes to the preferences managing a key value in order to run some action when the value is changed. These are the attributes that also have to be used with the special preference GrxButtonPreference.

If you add more than one action to your preference, the execution order is defined by the string grxs_process_actions_order in values-configapp.xml.

However, in every preference you could override this order by using the attribute grx: processActionsOrder. You can define the order and actions to be run in this specific preference.

6.1- Actions

- **Run Scripts (needs su granted)**
    - Script File in assets folder:
        - grx:script="file_name_in_assets"
    - Script File in assets folder + Arguments
        - grx:script="file_name_in_assets" grx:scriptArguments="string with arguments"
    - Script based on string-array
        - grx:script="@array/array_name"

Other Attributes for Scripts (Common for the 3 cases)

- Confirmation dialog to run the script
  - grx:scriptConfirm="true" (or false). (by default is false)

- Toast after the script is run
  - grx:scriptToast="string text to show in the toast"

Have a look to demo_scripts.xml for more details.

- **Kill Package (needs su granted)**

  - grx:killpackage="package_name_to_kill"
  - grx:confirmKillpackage="true" ("false). (show confirmation dialog)

  Default value for confirmation dialog

  Bools.xml -> grxb_confirmKillpackage_default (false by default)

  Command for kinlling package

  String -> grxs_kill_command (change it if you need it. By default :

  pkill -l TERM -f

Have a look to demo_reboot_killapp.xml

- **Reboot the device (in some versions we will need su granted)**

  - grx:reboot="true" ("false").
  - grx: confirmReboot="true" ("false). (show confirmation dialog). By default true. You can set default in bools.xml -> grxb_confirmReboot_default

Have a look to demo_reboot_killapp.xml

- **Send BroadCasts**

  - grx:broadCast1="string_action"
  - grx:broadCast2="string_action"

  broadCast2 is sent with a delay of half a second.

Have a look to demo_groupkey_bcs.xml

- **Send Common broadcast with extras**

The common broadcast action for all preferences is defined in the string grx_common_extra_broadcast in values-configapp.xml.

To send the broadcast with that action you have to add the following attributes to your app:

- grx: commonBcExtra , set the key for the extra value sent
- grx: commonBcExtraValue , the value to be sent.

Have a look to demo_groupkey_bcs.xml

You can use your broadcast for update mods, or to carry out some kind of action you need in your mods.

- **Change a group key**

  - grx:groupKey="name_of_group_key"

  Group keys are very usefull if you want to update a group of preferences which all have the same groupkey declared. It will helps you to reduce system resources (content observers). By adding a groupKey to several preferences, apart from changing their corresponding android keys, the group key is changed (it is a counter from 1 to 32). So, you can register just this observed key in your observer. Once you detect it changes, read all these keys in your mod and update your mods. I think it is better to spend some more battery in reading groups of keys than registering hundreds of observed keys.

  The user will spend more time using your mods than configuring then, and you can design your mod´s infrastructure in several ways.

Have a look to demo_groupkey_bcs.xml

- **Simulate onClick on other preference**

  When a preference value changes, you can simulate an onclick action on other preference present in the screen based in simple rules.

  - grx:clickRules, By adding this attribute to a preference , once this preference value is changed, this rule will be processed in order to decide if a onclick event on other preference has to be done.

  The format of the rule is quite simple

  values to check#target preference key#values to check in dest pref

  The onclick rule will be processed when this preference value changes to any of the specified values. The onclick will on the target preference will be done if the dest pref current value is one of the specified in the rule.

Have a look to demo_pref_actions_click.xml to see simple examples of use of clickRules.

6.2.- Order for executing the actions

Default order

De default order is set in the following string:

<string name="**grxs_process_actions_order**">scripts|broadcasts|groupkey|reboot|kill|</string>

If you want to change it, please keep the names, and ends it with |

If you want to delete some action (because you will never use it) delete it and do not forget to delete its corresponding |

Overriding actions Order

You can override in individual preferences the default order for executing the actions.

grx:processActionsOrder=”actions to process separted by |”

example: grx:processActionsOrder=”groupkey|kill|broadcasts|”

7.- Stock Settings preferences: System, Global and Secure support

You can add Settings preferences used in your secsettings or other app, by adding to the preference declaration the following attribute:

grx: systemType=”type_of_settings_pref” where type can be: “global” , “secure” or “system”

These preferences are updated in a different way to normal app 1preferences. Every time the preference screen is resumed the app update all declared settings preferences, so if the user goes to the stock settings app and change the value, the preference in the app is updated when he is back to it.

In these preferences, the initialization is done using the settings system value except when restoring a preferences backup.

IMPORTANT: If you want to use this feature, you have to know that according to android security, only privileged apps can write and read the settings system. If you compile this app with the default target sdk (22) may be you are able to write in settings system, but for sure not in secure and global unless you install the app in priv-app .

8.- Understanding Preferences Storage, backup, restoration and Synchronization.

This app stored all the preferences values in:

- its shared preferences xml
- Settings$System table

In shared preference Booleans values are stored as Boolean, but the corresponding value in Settings system will be, of course, an integer value (0 or 1, in this case).

To make the app consistent and to support settings preferences addition to your preferences screens the behaviour is the following:

- Backup: shared preferences, preferences images are backed up.

- First Run and Preferences Restoration: On first run and in preferences restoration, the app will synchronize settings system values with its stored shared preferences.

  Settings preferences (stock global, secure, system prefs) are also synchronized, since we are storing them.

9.- Dependency rules

You can enable or disable a preference based on other preference value(s). The referred preference may or not to be in the same preference screen.

Dependency rules use the app shared preferences values

Checked Values in the referred preference

- The current preference value is empty/null
- The current preference value is one of a set of values to compare
- The current preference value contains a value to compare.

The involved logic is an OR logic.

How to add a dependency rule

grx: depRule =”Result#referred_key#type_of_referred_key#values-to_check”

Where:

- Result must be one of the two following strings: ENABLE or DISABLE
- referred_key is the key name to check.
- Type_of_reffered_key can be (we need to help the app to know how to check.. ☺)
    - INT (for preferences stored in the app´s shared preferences as int)
    - STRING (for preferences stored in the app´s shared preferences as strings)
    - BOOLEAN (for preferences stored in the app´s shared preferences as boolean, this is checkbox, switch and file preferences)
- values-to_check
    - One or more possible values, separated by commas can be checked.
    - The NULL value checks if the referred
    - Values embedded in parenthesis means “contains”
    - Values not embedded in parenthesis means “is”
    - The comma character means OR
    - For BOOLEAN values to check you have to write “TRUE” or “FALSE”

Some examples to will clarify this:

- DISABLE#key_1#INT#1

  The preference with this dependency rule will be disabled if the preference´s value of the preference with key key_1 is 1. Int values are expected.

- ENABLE#key_1#INT#1,4,5

  The preference with this dependency rule will be enabled if the preference´s value of the preference with key key_1 is 1 or it is 4 or it is 5.

- ENABLE#key_1#STRING#1,(4g),NULL

  The preference with this dependency rule will be enabled if the preference´s value of the preference with key key_1 is 1 or contains the string 4g or it is empty. String values are expected to check.

- ENABLE#key_1#BOOLEAN#FALSE

  The preference with this dependency rule will be enabled if the preference´s value of the preference with key key_1 is false. A Boolean stored value in shared preference is expected.

  Have a look to demo_dependency.xml for updated info.

10.- Creating your navigation menu

GrxSettings uses the following menu xmls (res-menu)

- menu_app_main.xml -> 3 dots menu (backup, restore and build prop simulator when in demo mode)
- menu_user_options.xml -> user options menu (divider height, panel header bg, remember screen, etc.).
- menu_grx_demo.xml -> this is the navigation menu for the preference screens of the demo. Use this menu as a reference for your menu.

- **menu_app_preferences_screens.xml**

  Add your menu navigation in this xml.

  - **Adding a preference screen menu link**

    <Text android:id="@+id/your_preference_screen_xml_name"

    android:title="Your Title" android:hint ="Your SubTitle" />

    - android:id is a must. The id name should be the same as the xml preference screen name that you want to show when the user clicks on this option. For example, If your preferences xml is called my_screen.xml then android:id="@+id/my_screen"

      android:hint is optional.

    - Other optional attributes:

      - android:icon="@drawable/drawable_name" (shows an icon)
      - android:showIconSpace="true" (or "false") (if true an empty space will be showed.)

  This app uses SublimeNavigationView library. It supports other attributes but the app is assuming that the indicated attributes are the only used to make all the options consistent.

  - **Adding a Separator line**

    Add the following in any place to show a separator line in the navigation panel.

    <Separator/>

  - **Adding a Centered Text**

    I have added this option for styling purposes, which shows a centered text that can be used to make clearer for the user the navigation menu.

<CenteredText android:title="Your title" android:hint ="your subtitle" android:checked="true"/>

Both android:hint and android:checked are optional. If android:checked is set to true, the text is coloured (see styling section for more details)

- **Adding Grouped items**

You can easily show an expandable/collapsible group of items in the navigation panel through using the following tags:

```
<Group
    android:id="@+id/group_id"  app:collapsible="true" app:collapsed="true">
    <GroupHeader android:id="@+id/header_id"
      android:title="Your group title"
      android:hint="Your group subtitle"/>
                <!-- YOUR ITEMS (Text, Separator, CenteredText) here -->
</Group>
```

- android:id has to be set both in group and header tag
- GroupHeader has to be added when you add a group
- app:collapsible true indicates that the user can expand – collapse the group by clicking on the header. app:collapsed true indicates that the initial state of the group is collapsed

You can add as many groups as you need to your navigation menu.

Nested groups are not supported atm.

Have a look to the demo_navigation_screens.xml menu

11.- Adding Build Prop rules to individual preferences, navigation items and groups of screens

You can show or hide individual preferences, individual navigation menu items (screens) and navigation gropus based on values of properties contained in the rom´s build.prop.

You can use this feature for example to auto-configure the app navigation and preferences availability based on your aroma installation if you have different mods depending on the aroma user selections, or simply based on other existing buid.prop properties (device name, or whatever you want to check)

The logic behind the build prop rules is the same than the dependency rules with the following rule format in this case:

- In your menu_app_preferences_screens.xml use the following notation

  app:grxBPRule=”YOUR_BP_RULE”

  You can add a rule to Group and Text tags

- In preferences screen add to the desired preferences the following

  grx:bpRule=”YOUR_BP_RULE”

The logic and format of the BP rules are the same in both cases. I have kept the original sublimenavigationview used namespace, that is why the xml attribute is different for now..

YOUR_BP_RULE

HIDE_SHOW#property_name#values-to_check

Where

- HIDE_SHOW has to be HIDE or SHOW
- property_name the property name to check in build.prop
- values-to_check , with the same logic than in dependency rules

Examples

- SHOW#ro.myprop#TEST,TEST2,(FULL)

  The preference, navigation item or navigation gropu will be shown if the value of the property ro.myprop is exactly TEST or it is exactly TEST2 or it contains the word FULL

- HIDE#grx.demo.prop#(LITE)

  The preference, navigation item or navigation gropu will be hidden if the property grx.demo.prop contains the string LITE

If you need inside a preferences screen to hide several preferences based on a BP rule, wrap them inside a GrxPreferenceCategory

Have a look to demo_bprule_screen1.xml , demo_bprule_screen2.xml , demo_navigation_screens.xml menu and run the app in demo mode to see how it works.

12.- Nested Screens

Have a look to demo_nested.xml to see how this feature works. In contrast to standard nested screens in this app the toolbar is kept.

13.- Tabs for Rom Info o whatever you need

When the app is run the first time or on every execution if the user did not enabled the remember app option, the screen showed is the Rom Info.

This section can be built with tabbed screens, as many as needed.

When in demo mode the Rom info section is taken from the arrays demo_tabs_names and demo_tabs_layouts, in values-demo.xml

The first array contains the tabs titles and the second one the name of the layouts to be loaded on each tab.

When you set the app not in demo mode, the app will look for the following arrays (you will find them in values-rom.xml:

- **rom_tabs_names and rom_tabs_layouts**

Have a look to the demo tabs layouts to see witch kind of resources you can add to them in order to design your own screens.

Summarizing:

- The main container of these layouts has to be of type ObservableScrollView
- You can add any standard android view : ImageViews, TextView, etc… and containers (FrameLayout, LinearLayout, etc..)
- I have added to the app some views in order to make both easier and more powerful the creation of screens.
    - **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=”yourурl” -> 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=”yourурl” -> 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=”yourурl” -> 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.

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

<string-array name="tabs_layouts"/> . An empty screen is showed.

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

14.- Themes

All themes are based in 2 main themes:

- Theme.Base.Light , for themes with light backgrounds
- Theme.Base.Dark, for themes with dark backgrounds.

Use these two themes to build your own themes. Also have a look to the exising themes as a reference.

It is important you to keep the consistency of the new themes, in terms of attributes used by them, or you could experience unexpected FCs.

It is quite easy and very quick to add a new theme taking as reference one of the existing.

If you add a new them and you want it to be shown in the Theme selection user dialog, you have to:

- Create the theme
- Add the theme title/description to the grxa_theme_list array in values-grxsettings.xml (pay attention if you are using different languages..)
- Add the theme name used in the styles.xml to the array grxa_theme_values that you will find in values-grxsettings.xml

15.- Summary of the main xml used in the app.

- values-configapp.xml : main configuration options and default values of the app

- values-grxsettings.xml (and associated languages) : values used in the preferences and some app features

- values-demo: values used in demo mode

- values-rom : you should use this xml to add your strings, arrays, etc..

- demo_navigation_screens : Navigation Panel menu in demo mode

- rom_navigation_screens : Menu when not in demo mode. Use this menu to add your screens.

- three_dots_menu :

  - backup and restore menu. When in demo mode it also shows the build prop rules test utility.

  - Reset all Values to their defaults.

  - Tools: restart UI, restart Phone, go to recovery

- user_options_menu : This is the navigation panel user menu.

16.- Grouped Values Functionality.

This is a spcial feature with its special rules that allows us to save in one settings key a set of individual keys. The individual keys are not saved in settings system but in the app´s shared preferences system. The output format is a string, conatining each individual key – value.

With the goal of to optimize the code and to make easier for modders to design their mods, we can use alias instead of the real key names.

The idea behind this feature is to make easier for modders the personalization of views that are added to different places. For example, let´s imagine a custom battery that we place in home screen, notification panel and in lock screen. Let´s say, for example, that our battery class uses 14 options. If we want to allow to the users to customize the battery on every place we added it (3 in this example) we would need to manage and add 42 different keys. With this feature we will use only 3, if we design properly our view. Now, let´s think in our battery view code. To make the code easier to develop we could use the alias feature. So the 3 grouped values keys will use inside the same alias for the similar keys. For example, let´s say main switches for each of the 3 vies are: ms_home, ms_panel, ms_lockscreen, and let´s think that we are defined the same alias for in the 3 cases, let´s say ms_alias. These real keys names and values will be saved in GrxSettings Shared Preferences, but, in settings system grouped values we will find the same alias. So, our view code doe not need to be customized taking into account the real name of the preferences in grx app.

You can alos use this feature to reduce the number of inserted keys to settings system, grouping all the keys inside a single key saved to system. This will allow us to design new ways of modding.

The following attributes has been added to GrxSettings in order to manage this functionality.

- **grx:groupedValueKey**

  Each individual key you want to be inserted in the resulting grouped value key needs to define this attribute.

  You can define per preferece screen as many grouped values you need.

- **grx:groupedValueSystemType**

  With this attribute we define where the grouped value will be saved in settings. You can use

  “system”, “secure” or “global”. While developing the mod you could use “” for testing purposed.

- **grx:groupedValueBroadCast**

Define a broadcast action to be sent after the gropued value is saved to settings. The grouped value is sent in the broadcast as an extra string with the same key name than the involved grouped value key.

In your mods either a contentobserver or a bc receivier can know that the grouped value changed. You could read the key on boots and to receive the changes via bc, that is up to you.

- **grx: groupedValueMyAlias**

  The alias to be used instead of the real key in the resulting gropued value.

Apart from this, in **values-configapp.xml** we found the following default configuration options used in the logic of this feature.

- **grxb_groupedValuekey_savekeysnames**

  it´s a bool, by default set to true. If you set it to false, then in the resulting gropued values string the keys names (or aliases) will be not saved. Then your mod will need to know the order you wrote the preferences to "decode" the values.

- **grxb_groupedValuekey_savekeysnames**

  String defining how the key or alias and the associated value will be separated. By default I have set **~~**

- **grxs_groupedValuekey_valuesSeparator**

  String defining how each pair of keys-values will be separated in the grouped value. By default I have set it to |||

  When you define these 2 separators take into account the separators you use normally in your preferences!!!.

Logic and Rules

Grouped Values can be used inside the same preference screen. If you use the same grouped value key in other preferences screens or subscreens the values will be overwritten when we go to those screens. So, use one grouped value key name just in one preference screen.

You have to add the same **groupedValueKey** attribute value to each preference you want to be saved in the same grouped value.

groupedValueSystemType and groupedValueBroadCast are evaluated only the first time they are found for a given groupedValueKey. This means that if for example pref1, pref2 and pref3 uses same gropuedValueKey and pref1 and pref2 have got defined

gropuedValueSystemType, then only the valued found in the first preference, this is pref1, is taken into account.

All changes in individual preferences makes the grouped Value to be updated. Every time we change a value belonging to a grouped value, the resulting value is calculated and saved to settings and the broadcast sent, if defined. **If you add a GrxButtonPreference** using an existing groupedValueKey attribute, then the behavior is different. In this case, only when the user press the button the resulting grouped value key is calculated and saved in settings and the broadcast sent. You can adda as many buttons as you need. The moment one button is detected the logic will stop saving to settings on every single change.

Output Value

The grouped value is saved as a string, with the following format. Assuming ~~ is the key-value separator you are using and the separator among key-values pair is ||| the resulting string will look like:

Key1~~value1||| Key2~~value2||| Key2~~value2|||…

Remember that KeyX will be the alias used intead of the real key name if you use the alias attribute.

- **BOOLEANS Values:** for Grx CheckBox and Switch preferences the true is saved as a 1 and false is saved as 0
- **INTEGERS Values.** For preferences as seekbars, number pickers, color picker.. the string representing that integer value will be saved. The program uses the String.valueOf(int) instruction.

In your mod or in your class for support this grouped values feature you will need to split by the main separator ||| and then to split again by the key value separator, unless you are not using key-values separator.

EXAMPLE

We have some options for a battery in home screen

<GrxPreferenceCategory android:title="@string/battery_home" …. />

<GrxSwitchPreference android:key="grxbath_enabled" grx:groupedValueKey="grxcirbathome" grx:groupedValueMyAlias="be" grx:groupedValueSystemType="system" grx:groupedValueBroadCast="com.grx.tests" />

<GrxSeekBar android:key="grxbath_size"… grx:groupedValueKey="grxcirbathome" grx:groupedValueMyAlias="size" />

<GrxSeekBar … android:key="grxbath_stroke" grx:groupedValueKey="grxcirbathome" grx:groupedValueMyAlias="stro"/>

And same options with different android:key names for lockscreen…

<GrxPreferenceCategory android:title="@string/battery_lockscreen" …. />

<GrxSwitchPreference android:key="grxbatls_enabled" grx:groupedValueKey=" grxcirbatls" grx:groupedValueMyAlias="be" grx:groupedValueSystemType="system" grx:groupedValueBroadCast="com.grx.tests" />

<GrxSeekBar android:key="grxbatl_size"… grx:groupedValueKey=" grxcirbatls " grx:groupedValueMyAlias="size" />

<GrxSeekBar … android:key="grxbatl_stroke" grx:groupedValueKey=" grxcirbatls " grx:groupedValueMyAlias="stro"/>

Android Switch key for home is grxbath_enabled and switch key for lockscreen is grxbatls_enabled. Grouped key value for switch home is grxcirbathome and it´s the same attribute defined for the seekbars for home screen. In Lock screen the grouped value key is grxcirbatls, the same that are using the 2 seekbars for the lockscreen battery. However the standar key names of the seekbars are different.

All the preferences for the battery options in homescreen are using the same group value key and have defined an alias. All the preferences for the battery options in lockscreen are using the same group value key and have definde an alias. **The alias** used for the corresponding preferences for each battery are the same. The alias of the switch for home and lock screen are the same. Same happens with the seekbars. So, in the battery mod we could use the same code to decode our individual values. How can then to know in my mod code which of the 2 grouped values keys to use in order to configure by code my observer etc?. Well, there are many ways, for example tagging differently the views in the different layouts you use the view. For example, let´s tag the batteries of the example with the grouped value key name. The logic would be… "if my tag is grxcirbathome" then observe grxcirbathome…If my tag is grxcirbatls then observe grxcirbatls grouped value.." Once the observer is warned that a changed ocurred we decode the right grouped values saved to settings or received via broadcast easily. So, I do not need to think and to write a very long code as we do in the traditional way (…this is "… if I am grxcirbathome" then the key name for main switch is grxbath_enabled.. and the seekbar is … ".) We know that the main switch, for example, will appear in our grouped values using the key "be" ..

You can define other strategies when using these grouped values keys. There are many posibilities and it will depend on how you design your mod.

I highly recommend to convert the grouped value string to a map<key,Value> and to add support to get an integer, boolean or string value from the map in order to be used in your mods. But this is another story and I will be providing methods.

Finally, let´s imagin we want **to update our mod asynchronously by using a GrxButtonPreference** to update just the lockscreen battery. Adding this button

<GrxButtonPreference …. grx:groupedValueKey=" grxcirbatls " />