Amazfit Bip WF Editor by Ilgruppotester

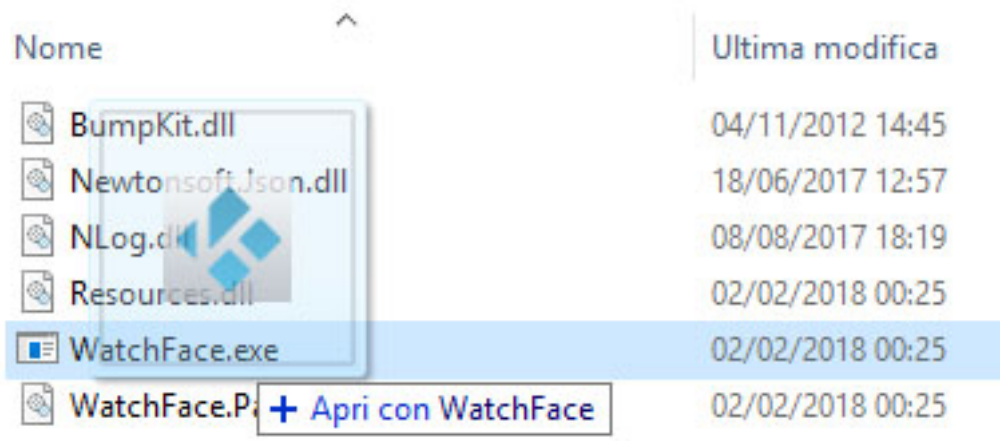
User's Guide

Prerequisites

- VCRedist 2008 Libraries
- watchface.exe 1.0.2.8 version to extract bin files and / or create them when you have completed editing process
https://bitbucket.org/valeronm/amazfitbiptools/downloads/

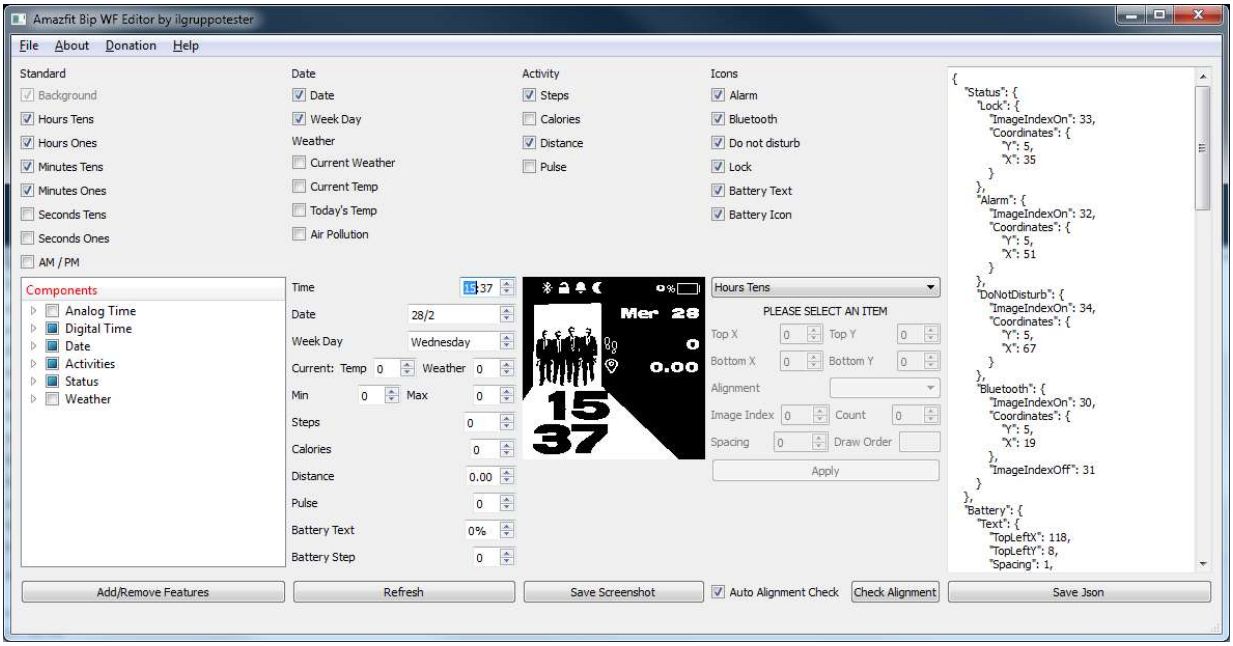
Before starting

- If you want to edit an existing Watchface, you must first extract the contents of the bin file. Once unpacked Watchface, simply drag the bin file over the icon of the executable file

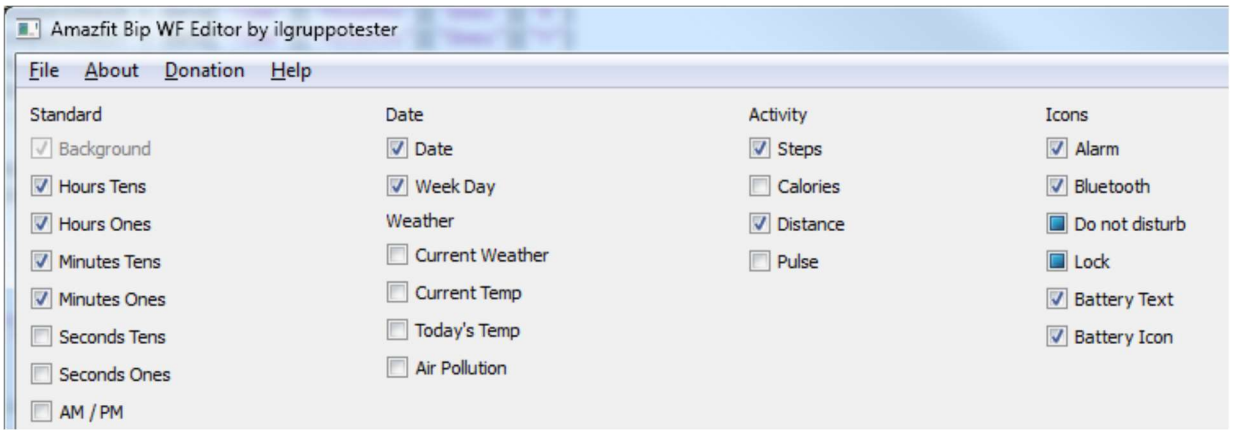


- Instead, if you want to create a new Watchface, no special operations are needed

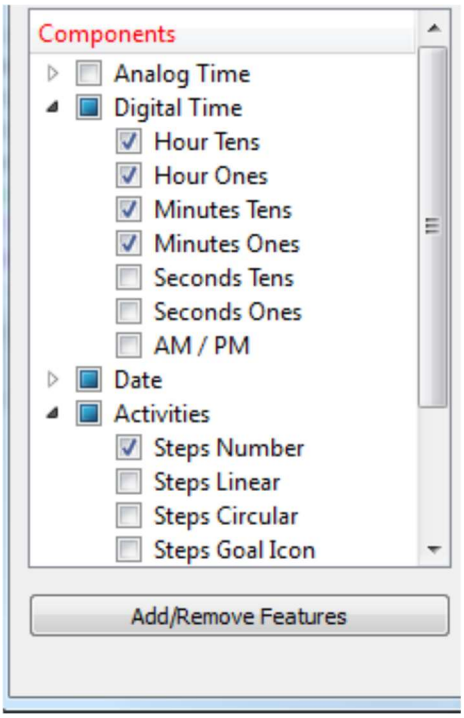
User interface



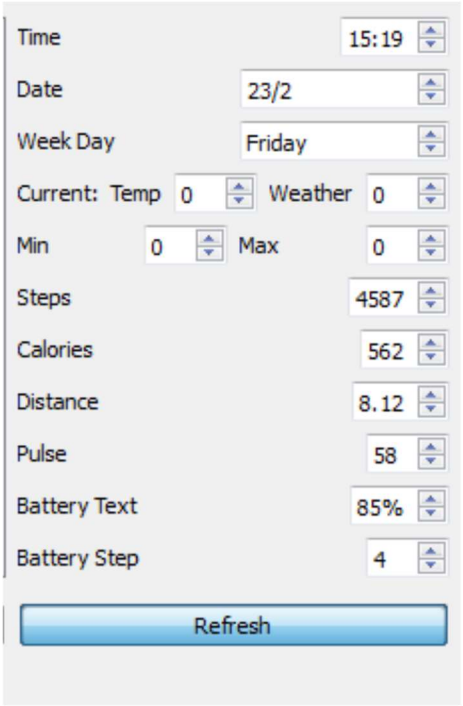
Overview



Checkbox Area *to enable / disable the display of an object. In the case of properties with icons On / Off, the checkbox has three states: object not displayed, OFF status icon and ON status icon*

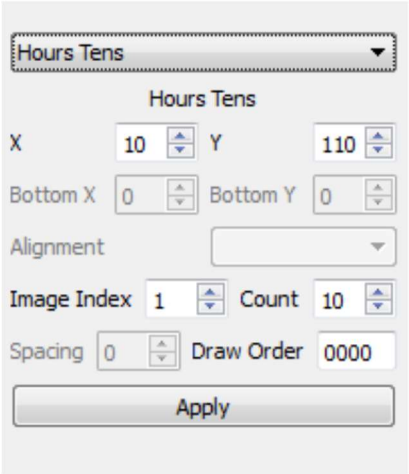


***Components tree**: Enables adding / removing objects from json. To change them, you must check / uncheck the required objects and click the Add / Remove Features button*



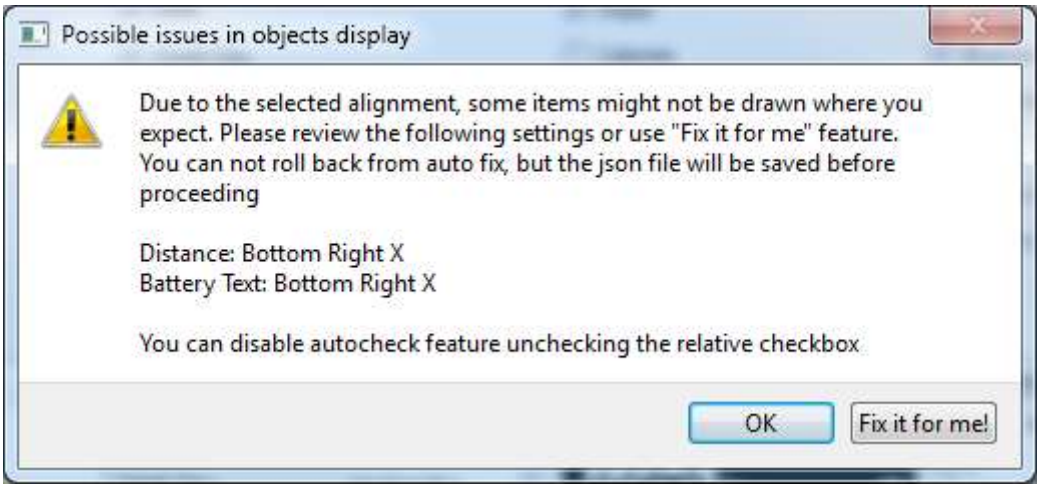
***Preview Settings**: Area where the values displayed in the preview can be changed. Note: some objects are linked to the set value, such as Step progress: until the number of inserted steps does not allow to reach a level such that it can be drawn, there will be no visual feedback. Progresses are calculated on the standard objective of 8,000 steps. Pressing the Refresh button, the new preview is generated*

***Preview**: Here you have a preview of the result. Clicking the Save Screenshot button, a preview in jpg format will be saved file in the bin same folder*

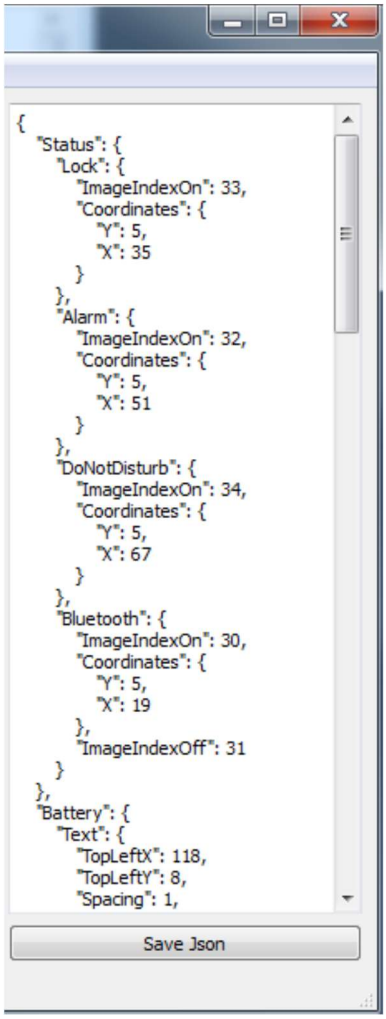


***Edit Area**: Selecting an item from the dropbox, you can change the available properties. By clicking on the Apply Button, the new values are entered into the json and a preview Updated will be generated.*

***Alignment Check Area:** When Auto Alignment Check checkbox is checked, at each refresh will run an alignment check is performed. If disabled, you can force an "Alignment Check" by clicking on the button.*



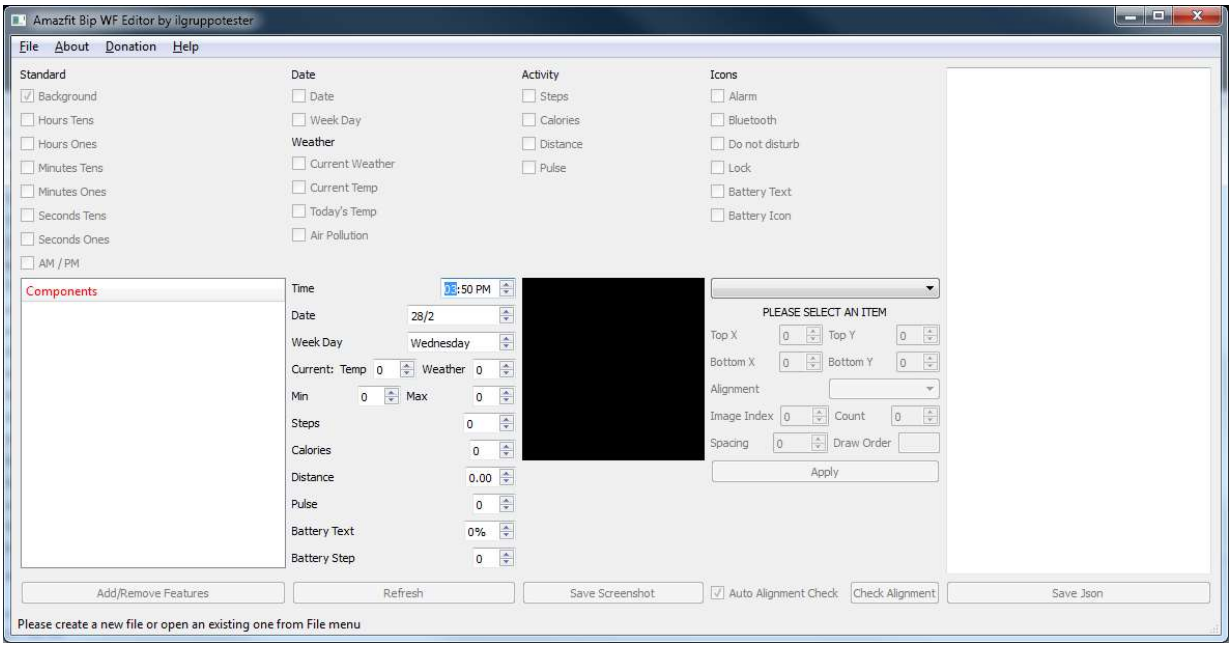
Alignment Error *If alignment errors are detected, a message is shown, reporting the objects for which there may be a problem. You can automatically fix them by clicking the "Fix it for me!" or click OK to ignore the message. For more details, see alignment in "Useful information"*



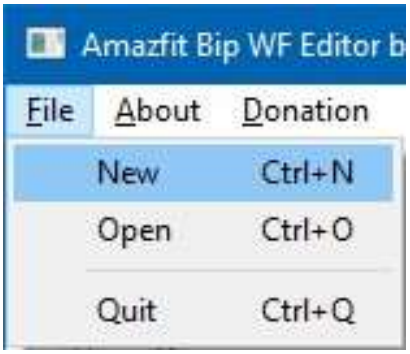
Json Editor: *You can also manually edit the JSON file (do not recommended if you do not know well what you are doing). The Save button is used to save the progress of the changes. There is no auto-save feature, so you will then need to save the file manually.*

Instructions

Unzip the zip file editor in any folder and launch the executable "Amazfit_Bip_WF_Editor_ilgruppotester_vxxx.exe"

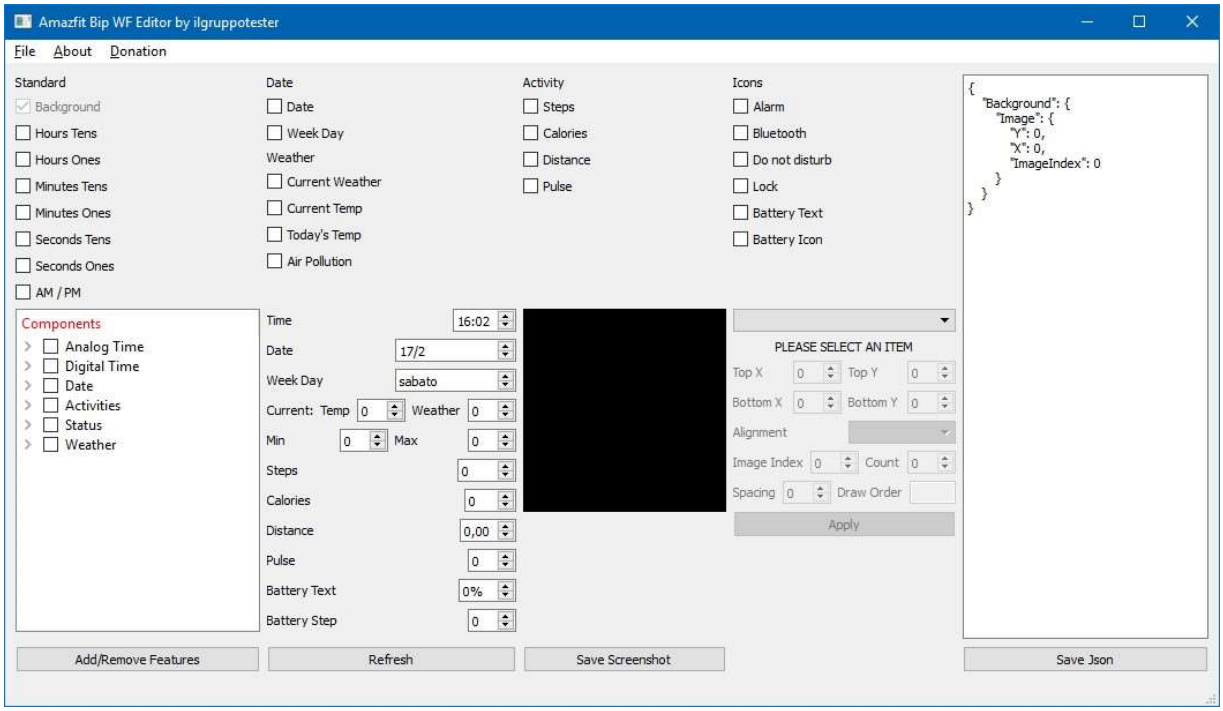


When you start, it will get a clean interface from which you will have to choose whether to create a new Watchface or edit an existing one, via the "File \ New" menu or "File \ Open"

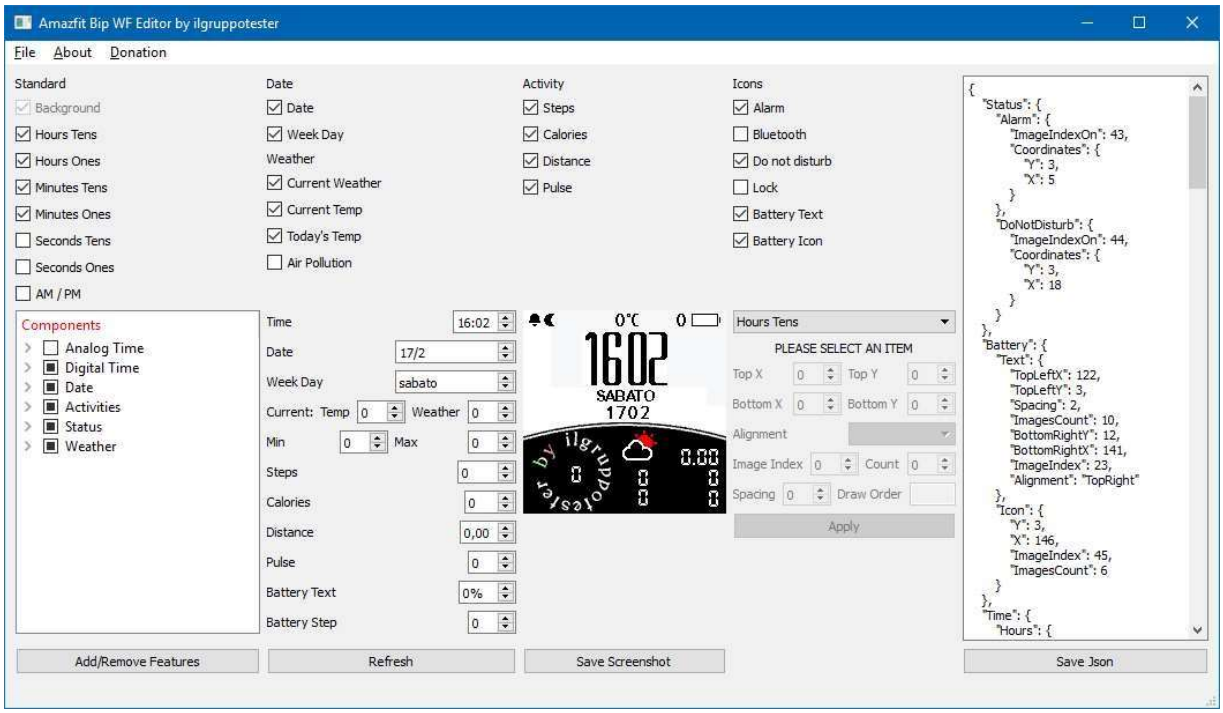


In the case of new wf, you are prompted to select the folder where to create the json file, in case of opening an existing one, however, will need to select the desired file json.

If a new wf is created, a background file will be copied to the folder and its part of the json will be populated.

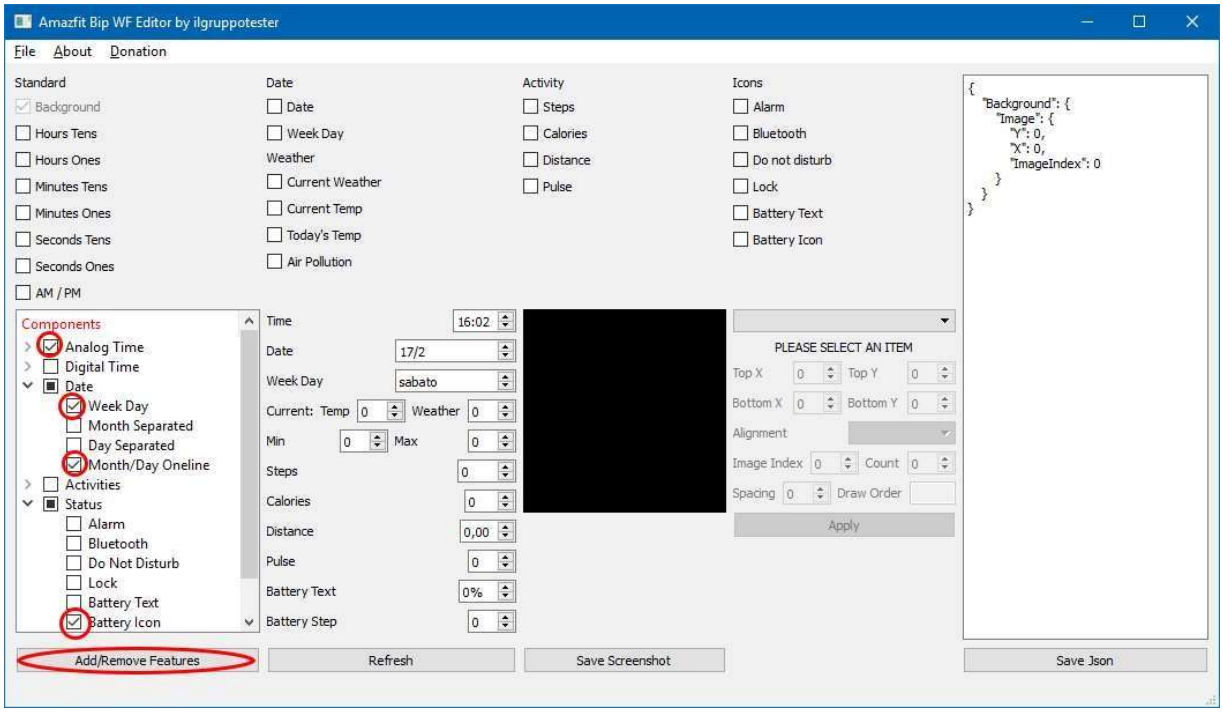


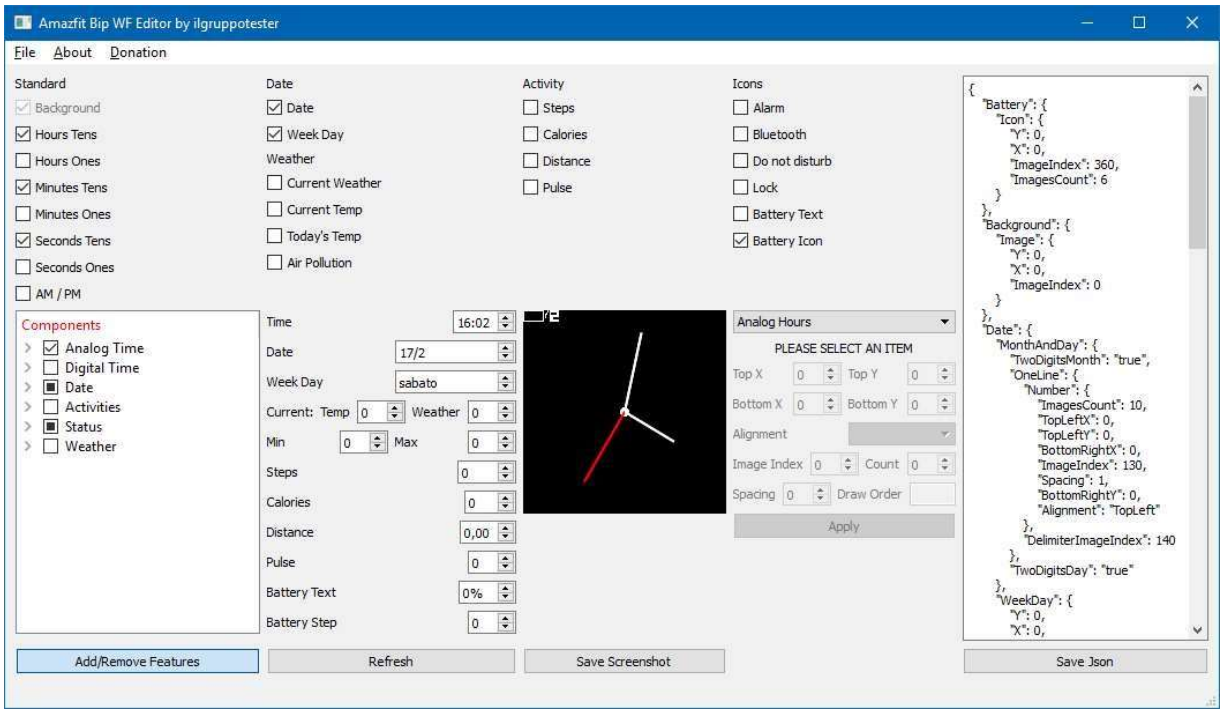
If, however, an existing json is selected, all the elements present in the wf will be loaded and displayed, also the section of the json will be populated with the entire contents.



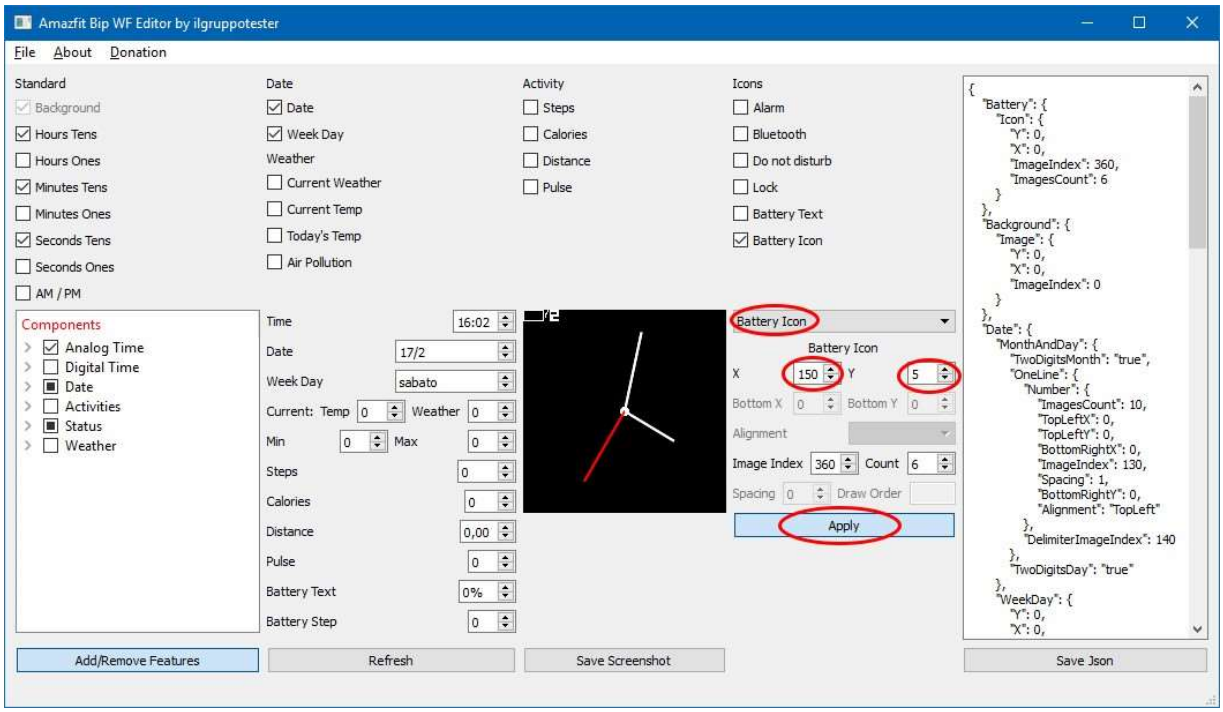
From this point on, we proceed the same way for both cases:

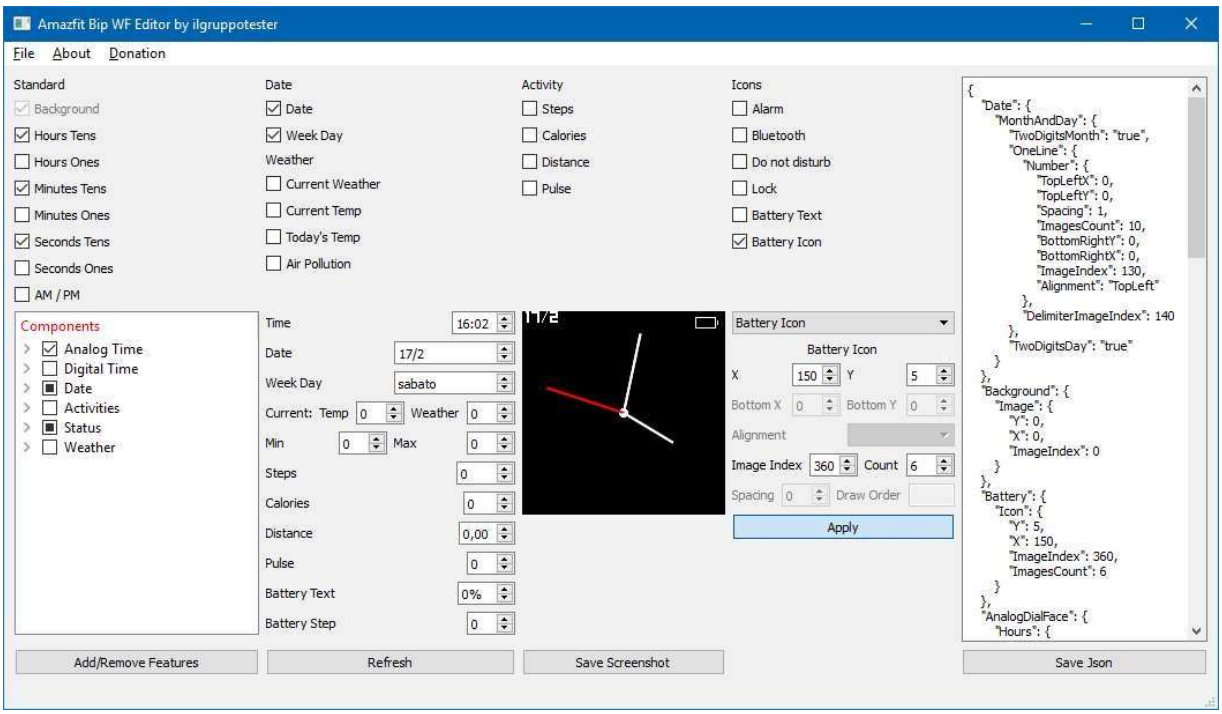
you can add and remove objects simply by checking or unchecking them from the tree view "Components" and clicking on the "add / remove features". In case of addition, default images will be copied into the json folder, in order to have undergone a visual feedback.



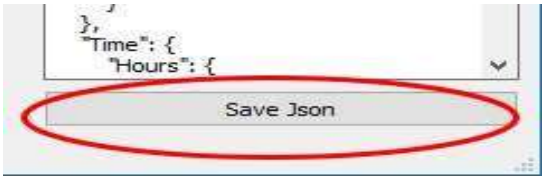


Instead selecting an object from the dropdown list, will display its properties, and once set, click on the "Apply" button, the data will be included in the json and it will show the preview date.

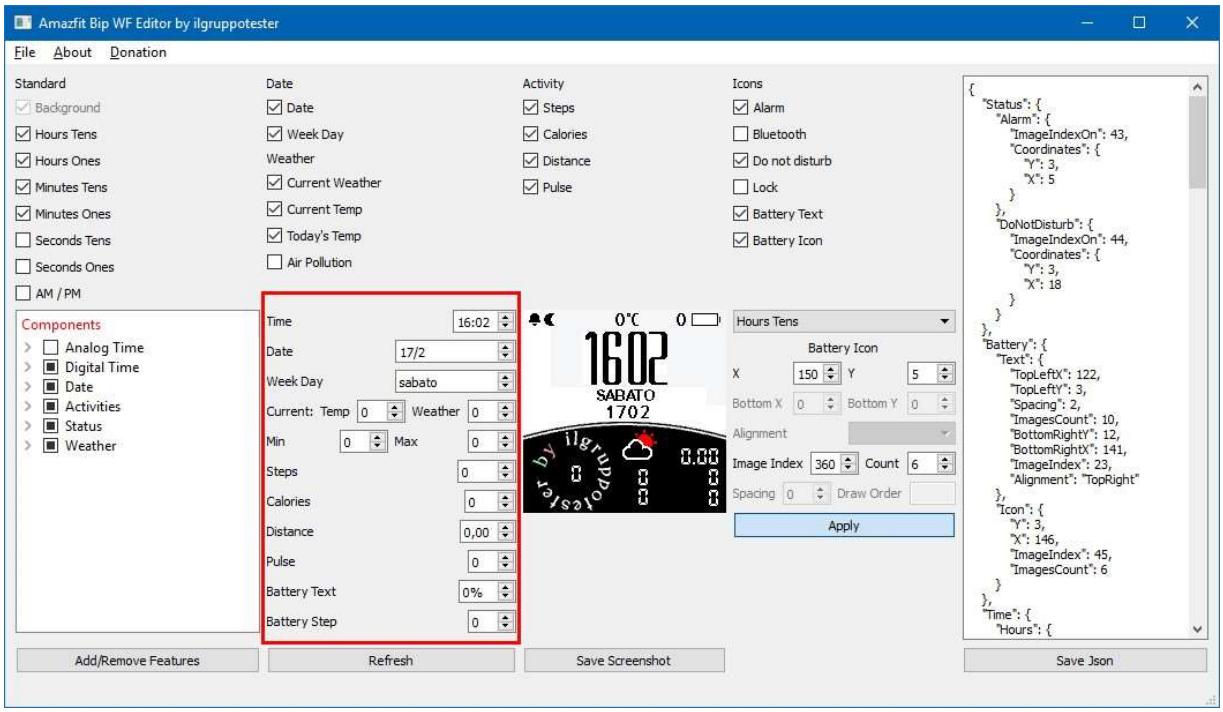


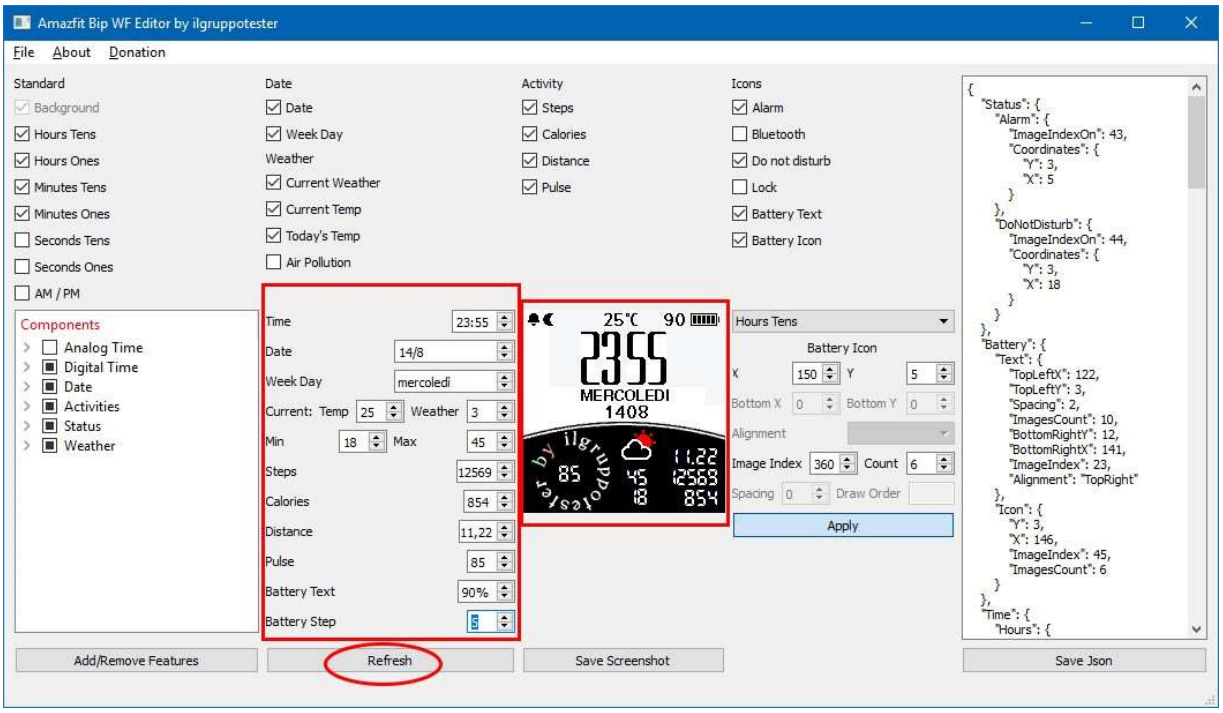


Remember, the json is not automatically saved, but you will need to click on the "Save Json" button.

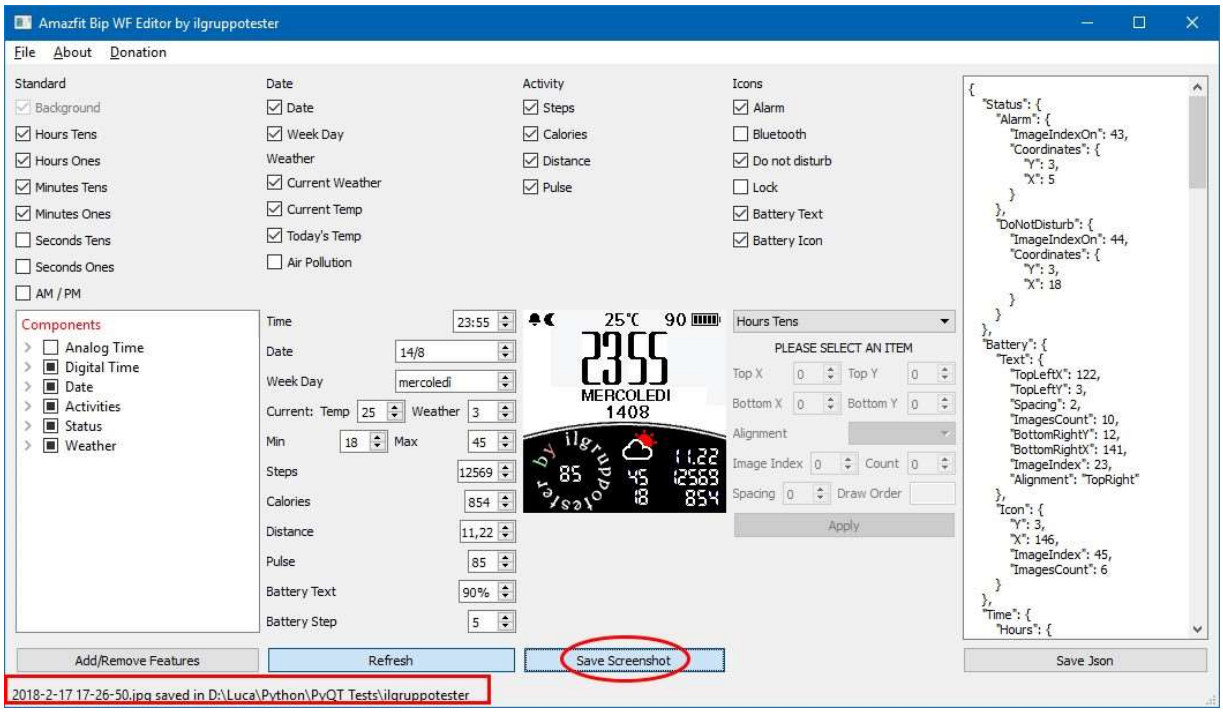


To have a detailed preview of the result, you can change the values displayed through the appropriate area, in such a way as not to have disappointments when the values change on the smartwatch.

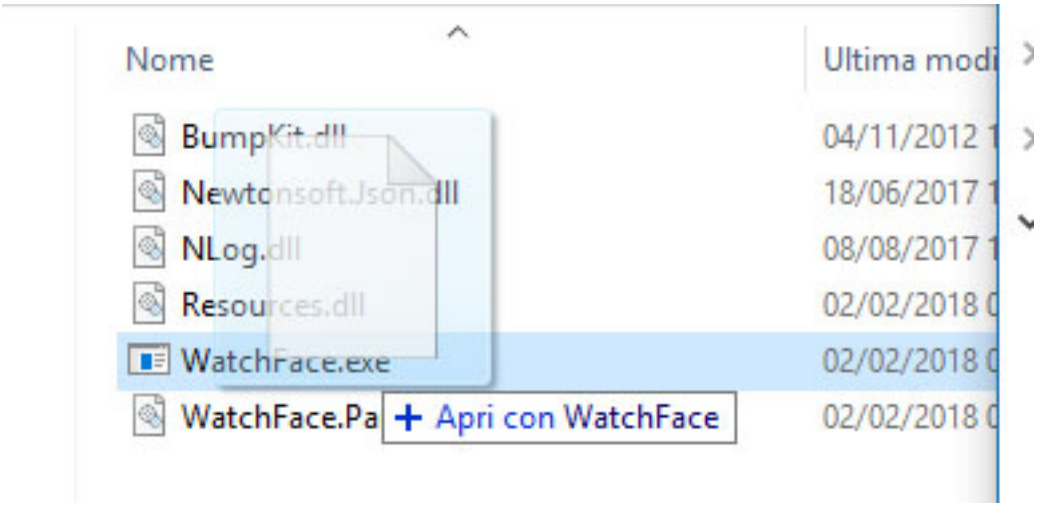




E 'can also save a screenshot of Watchface clicking on the "Save Screenshot" will create a jpg file in the same folder json.



To create the bin file to upload to the beeps, just drag the file executable to json Watchface.



useful guidance

Do not want to be an exhaustive guide, but a mere smattering of how the most used items in watchfaces.
The net is full of comprehensive guides, such as a rather interesting (in English) can 'be found here: http://amazfitcentral.com/2018/02/05/amazfit-bip-watchface-reference-including-some-json-docs/

General

Colors used

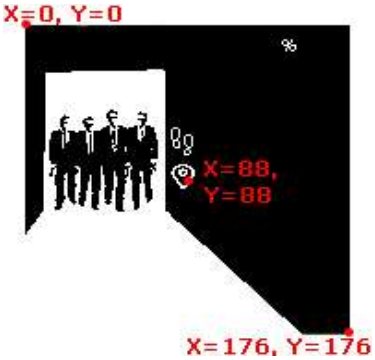
The color palette used on the beep is as follows, if images are present in other colors or other hues, watchface.exe will use the colors supported to return the visual effect of those not supported (dithering) in the phase of the bin creation.

Images

And 'possible, and recommended, to edit all images provided as a sample or extracted from existing bin, or create new ones, through an image editor such as GIMP or Photoshop. Also for this process are many guides on the net, and the advice is to try, as the satisfaction of having a Watchface with fonts that best suit your taste is priceless.

Background

The background image must be a png size 176x176 pixels



JSON File

The JSON file contains information about all the items on the watch face, to use the image alignment, and are different for each item. Those most important are:

X, Y

```
"X": 0,          # X coordinate of the upper-left of the image
"Y": 0,          # Y coordinates of the angle at the top left picture
```

In the case of individual images, such as the status icons, the hour digits ... the X, Y coordinates indicate the point on the screen from which the object will be drawn

ImageIndex and ImagesCount

```
"ImageIndex": 1,       # Name the file of the first digit, in this case 001.png
"ImagesCount": 10      # Number of images to be used starting at the (001.png - 010.png)
```

Used for example the hour and the day of the week, indicates the index of the first image and how many images are to be used starting with the

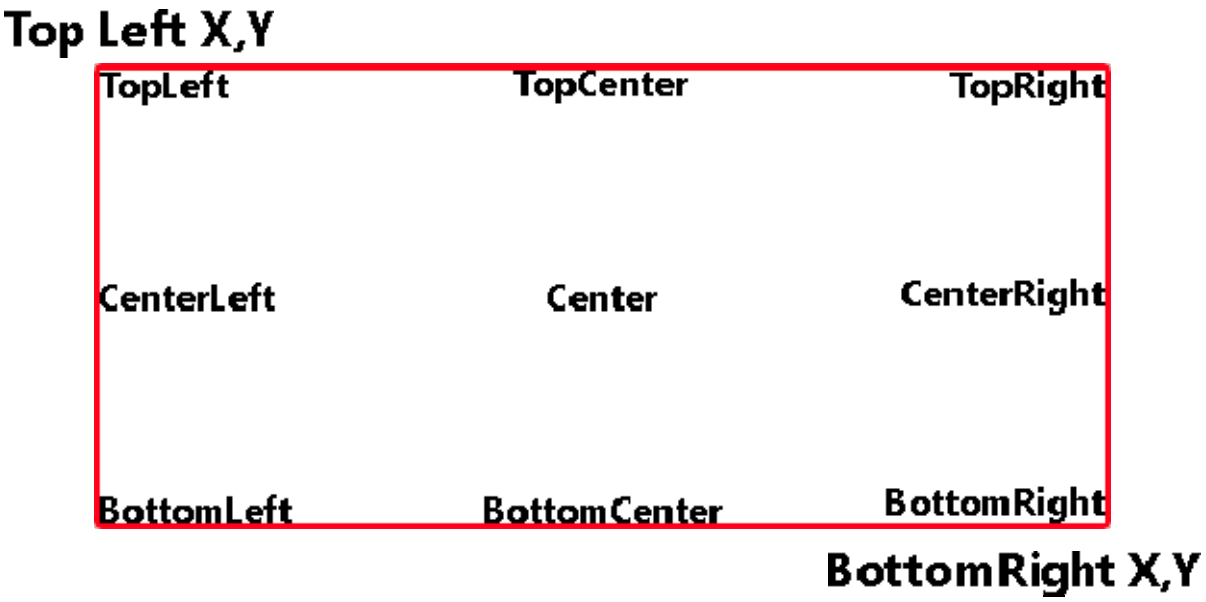
TopLeft, BottomRight X, Y

```
"TopLeftX": 102,              #Coordinate angle X in the upper left
"TopLeftY": 60,               #Coordinate angle Y in the upper left
"BottomRightX": 170,          #Coordinate angle X in the lower right
"BottomRightY": 70,           #Coordinate angle Y in the lower right
"Alignment": "BottomRight"    #Alignment, See next point
"Spacing": 2                  # Space in pixels between one digit and the other
```

Used in compound elements more than one digit, such as a percentage of the battery, number of steps ... remember: BottomRight X must be greater than TopLeftX, same thing for BottomRightY and TopLeftY

Alignment

For compound objects more than one digit, such as the battery is a numerical percentage, the steps, the distance ... you can choose how to align the text within a polygon having as its dimension coordinates ranging from TopLeftX, TopLeftY to BottomRightX, BottomRightY. In case of different alignments from TopLeft, if the text does not fit in the box, the beep the text will be automatically aligned TopLeft. For this reason, a warning message will be displayed when the information is incorrect



Let's look at a Watchface created during the tests on the editor:

Background

```
"Background": {
        "Image": {
                "X": 0,          # X coordinate of the upper-left
                "Y": 0,          # Y coordinates of the angle at the top left
                "ImageIndex": 0        # Name the file, in this case 000.png
        }
}
```

Now

```
"Time": {
   "Hours": {
     "Tens": {           # Figure indicating the tens of hours
       "X": 10,   #Coordinate angle X in the upper left
       "Y": 110,  # Y coordinates of the angle at the top left
       "ImageIndex": 1, # Name of the first digit of the file (0), in this case 001.png
       "ImagesCount": 10        # Number of images to be used starting at the (001.png - 010.png)
     },
     "Ones": {           # Digit indicating the units of hours
       "X": 47,
       "Y": 110,
       "ImageIndex": 1,
       "ImagesCount": 10
     }
   },
   "Minutes": {
```

```
    "Tens": {           # Figure indicating the tens of minutes
      "X": 10,
      "Y": 142,
      "ImageIndex": 1,
      "ImagesCount": 10
    },
    "Ones": {           # Digit indicating the minute units
      "X": 47,
      "Y": 142,
      "ImageIndex": 1,
      "ImagesCount": 10
    }
  }
},
```

Activities

Steps

```
"Steps": {
    "TopLeftX": 102,    #Coordinate angle X in the upper left
    "TopLeftY": 60,     #Coordinate angle Y in the upper left
    "BottomRightX": 170,        #Coordinate angle X in the lower right
    "BottomRightY": 70,         #Coordinate angle Y in the lower right
    "Alignment": "BottomRight"          #Allineamento, see note at the beginning
    "Spacing": 2                # Space between one digit and the other
    "ImageIndex": 11, # Name of the first digit file
    "ImagesCount": 10 # Number of images to be used starting at the
        }
```

Distance

```
"Distance": {
     "Number": {
       "TopLeftX": 111,
       "TopLeftY": 82,
       "BottomRightX": 172,
       "BottomRightY": 96,
       "Alignment", "topright"
       "Spacing": 2
       "ImageIndex": 11,
       "ImagesCount": 10
     },
     "SuffixImageIndex": 21,  # Image Index for the unit of measurement "Km"
     "DecimalPointImageIndex": 22     # Index of the image to separate units of the distance from
                                        the decimalGenerally. "" Or ","
```

Date

Day of the week

```
"Weekday": {
    "X": 92
    "Y": 30,
    "ImageIndex": 23,
    "ImagesCount": 7
  }
```

Day and / or month

```
"At your place": {
   "MonthAndDay": {
    "Separate": {
     "Day": {
       "TopLeftX": 130,
       "TopLeftY": 30,
       "BottomRightX": 170,
       "BottomRightY": 40,
       "Alignment", "topright"
       "Spacing": 2
       "ImageIndex": 11,
       "ImagesCount": 10
     }
    },
    "TwoDigitsMonth": false,  # If true, the month is always shown with 2 digits eg. January = 01
    "TwoDigitsDay": true      # If true, the day is always displayed with 2 digits eg. 05
```

State

For objects of state category, we have two values that indicate the icon to display when an object is active, and the display when it is not

```
"Bluetooth": {
     "Coordinates": {
       "X": 19,
       "Y": 5
     },
     "ImageIndexOn": 30,              # Filename to Status Icon ON
     "ImageIndexOff": 31              # Filename to Status Icon OFF
```

Battery

As for "Battery Text", the behavior is the same as the relative activity objects

```
"Battery": {
   "Text": {
     "TopLeftX": 118,
     "TopLeftY": 8
     "BottomRightX": 137,
     "BottomRightY": 16,
     "Alignment", "topright"
     "Spacing": 1,
     "ImageIndex": 35,
     "ImagesCount": 10
   },

"Battery": {
   "Icon": {
     "X": 149,
     "Y": 5,
     "ImageIndex": 45, # Starting Image (Battery Empty)
     "ImagesCount": 5 # Number of images relating to various states of the battery
   }
```