Usage of: batocera-settings

Introduction

batocera-settings is a commandline tool that can work with regular config files and read/write/change it's content. There are keys and values stored in a file like `/userdata/system/batocera.conf`.

- It's recommended to use **## This is a text comment** for text comments
- It's recommended to use **#enable.godmode=hallelujah** for commenting values
- It's recommended to descripe content and functions in text form
- It's recommended to add content in sections
- It's recommended to use the form **key.description=value**

Down here is a small excerpt of a config file

```
# ------------ B - Network ------------ #
## Set system hostname
system.hostname=BATOCERA
## Activate wifi (0,1)
wifi.enabled=0
## Wifi SSID (string)
#wifi.ssid=new ssid
## Wifi KEY (string)
## after rebooting the batocera.linux, the "new key" is replace by a hidden
value "enc:xxxxx"
## you can edit the "enc:xxxxx" value to replace by a clear value, it will
be updated again at the following reboot
## Escape your special chars (# ; $) with a backslash : $ => \$
#wifi.key=new key
```

Recommended commands and expressions

batocera-setting is utilized by parameters parsed. These parameters can be used in the long and in the short format. It's a relict of RecalBox times thus the expression `python batoceraSetting.py -command load -key user.key` is used till today but arguments are now parsed to batocera-settings.

As batocera-settings is more modern and supports read/write/change of value let's see how the command line works:

- BATOCERA Basic usage
  - `batocera-settings -r [key]` read key `from batocera.conf`
  - `batocera-settings -w [key] -v [value]` write value `key=value` to batocera.conf
  - `batocera-settings -f [file] -r[key]` read value key from `file`
- BATCOERA Extended usage
  - `batocera-settings -e -r [key] -s [system]` iterate keys `system.key` and if

not available use `global.key`
- `batocera-settings -e -r [key] -s [system] -g [game]` iterate keys like above, but start with `system["game"].key`

Basic usage: *batocera-settings -f [file] -r [key] -w [key] -v [value]*
Extended usage: *batocera-settings -e -g [game] -s [system] -r [key]*

-f – Loads any config file, default '/userdata/system/batocera.conf'
-r – Read 'key' and returns value from config file
-w – Write 'key' to config file, mandatory parameter -v
-v – Set value to selected 'key', any alphanumeric value
-e – Activate extended mode, needed for parsing game/system specific keys
-g – Any alphanumeric string for game, set quotes to avoid globbing
-s – Any alphanumeric string for system

If `-e` is not set the parameters `-g` and `-s` are ignored!
Value `-w` needs an additional valid entry for `-v`!
Use `-w` together with `-e` to create system/game keys!
Use the errorlevels for scripts!
Always use quotes if you use arguments containing blanks!

Error code handling

Whenever `batocera-settings` was called you will receive an exit code number. This will help to identify errors - morover for debugging you can use the **status** command to held usefull output.

File and Key/Values error		
Error Code	**Error code explaination**	**Troubleshooting**
EC 0	No Error, value found	You made it!
EC 1	General error, e.g. command line error	Check your command line for correct parameters
EC 2	File error, e.g. the config file is not available	Check file path and r/w access to it
EC 10	Value error, key found but value is empty	Unusual setup but no error at all
EC 11	Value error, key found but it is commented out	Activate the key entry by `uncomment` command

EC 12	Key not found	Add the key by manual or check your command line for typos

Handling in scripts

I present here some short scripts, to show you how to make batocera-settings work in your script. As I'm more confident in shell scripting I give you just some small exampels in shell script.

1. bash: Obtain value
2. bash: Activate UART in `/boot/config.txt`
3. bash: Set a new key
4. python: Obtain a value

USE AT YOUR OWN RISK

1. bash: Obtain a value

obtain_value.sh

```
#!/bin/bash
#This is an example file how batocera-settings can be utilized
#to read a value out from /userdata/system/batocera.conf

value="$(batocera-settings -r power.switch.device)"
ret=$?
if [[ $ret -eq 0 ]]; then
    echo "Power Switch detected: '$value'"
else
    echo "No Power Switch detected!"
fi
```

2. bash: Activate UART in ''/boot/config.txt''

activate_uart.sh

```
#!/bin/bash
#This is an example file how batocera-settings can be utilized
#to activate UART in /boot/config.txt

batocera-settings /boot/config.txt -w enable_uart -v 1
ret=$?
if [[ $ret -eq 0 ]]; then
    echo "UART activated, uncommented enable_uart"
elif [[ $ret -eq 2 ]]; then
```

```
    echo "File is write protected!"
    echo "I make boot-partition writeable"
    mount -o remount, rw /boot
    echo "Please restart script"
else
    echo "Key: enable_uart not found"
    echo "Not a Raspberry System?"
fi
```

3. bash: Set a new key

activate_uart.sh

```
#!/bin/bash
#This is an example file how batocera-settings can be utilized
#to set a new key in /userdata/system/batocera.conf

value=$(batocera-settings -w core.PS4.emulator -v SONY4EVER
ret=$?
if [[ $ret -eq 0 ]]; then
    echo "PS4 core enabled!"
else
    echo "Annother error occoured!"
fi
```

4. python: Obtain a key

🔧 Fix Me!

obtain_value.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#This is an example file how batocera-settings can be utilized
#to read a value out from /userdata/system/batocera.conf with python

import subprocess

value = (subprocess.check_output(['batocera-settings', '-r',
'power.switch.device']))
if value:
    print "Power Switch Detected: ", value
else:
```

```
    print "No power switch detected!"
```