Batocera scripts

If you are looking for how scripts behaved in **v37** and below, refer to the legacy scripts page.

You can opt to have scripts launched at various points in Batocera. This will change the location you save the script to and how it opens.

Scripts during boot/shutdown

When booting and shutting down, Batocera checks for scripts three times at different stages in the boot/shutdown process:

- /boot/boot-custom.sh, a special case if you need scripts executed very early in the boot process
- /boot/postshare.sh, to launch scripts right after userdata is mounted
- /userdata/system/custom.sh, after EmulationStation has loaded, the preferred method for most use-cases

When these scripts are executed, Batocera sends the first argument to the script as `start` when Batocera is booting and `stop` when Batocera is shutting down. Putting code into cases or other checks for these argument values will cause that code to only be executed when booting or shutting down, otherwise code will be executed on both boot and shutdown.

boot-custom: First during startup, last after shutdown

In Batocera **v32** and higher, scripts can be launched early in the boot process, before much of Batocera has even begun loading. This is done by storing the script in the boot partition at `/boot/boot-custom.sh`. Keep in mind that this script will be limited to more basic commands/modules, as most things have not loaded yet. The filename must be `boot-custom.sh` otherwise it will not be checked.

Precisely, this is the first thing executed once *init.d* has begun. This is before the *network/share* population/IR remote daemon have even loaded so capabilities are limited. Check `/etc/init.d` to see the exact order of all the modules have been loaded in (`S00bootcustom` is when the `/boot/boot-custom.sh` is executed).

This can be used to effectively patch and/or override the modules loaded by Batocera. Here are some examples that are only possible with this method:

- Customize or disable `S33disablealtfn` to allow a certain text mode console on a particular TTY session.

- Tinker with hwclock to save local time instead of universal time to RTC.
- Run `fsck -a` or `e2fsck -p` to harden the file system against data corruption in the case of a power cut/disconnection.

postshare: First after mounting userdata mount, last before userdata gets unmounted

In Batocera **v35** and higher, scripts can also be launched after `/userdata` is mounted in the boot process, before the splash video plays (and before Emulationstation has begun loading). It has to be placed to `/boot/postshare.sh` and is executed directly by the bash interpreter (no executable flag required).

This is the only init.d scripts that doesn't execute in background. So if you use a sleep timer for example, the script will halt till the time from sleep command is up! Be aware of endless do-while loops! Get familiar with shell commands and the usage of the ampersand  **&**

custom: Last during startup, first after shutdown

While custom.sh still works in **v40**, it's deprecated since **v38** and and may be removed in a later version. (See changelog.) The recommended way to do custom scripting in Batocera now is the services method detailed below.

custom.sh (deprecated)

Sometimes you just want to fire up one script after successfully booting, for example when you want to start up a VPN or other after-boot tasks. In order to do this, create a script text file at `/userdata/system/custom.sh`. Be aware that the script will be executed independently of the executable (x) attribute being set to the script file or not!

This script is the very last one that will be invoked by Batocera at boot time after EmulationStation has launched (check `/etc/init.d/` to see all the modules that are loaded before then, `S99userservices` (previously `S99custom` in Batocera **v38** and below) is when the user script is launched). The filename must be `custom.sh` otherwise it will not be checked.

Make sure your script ends with Unix line terminators (LF), not Windows-style line terminators (CR/LF) otherwise the script will not launch. Use a real text editor to edit your scripts, especially if you edit them under Windows.

A simple custom script as an example

custom.sh

```
#!/bin/bash
# Code here will be executed on every boot and shutdown.

# Check if security is enabled and store that setting to a variable.
securityenabled="$(/usr/bin/batocera-settings-get
system.security.enabled)"

case "$1" in
    start)
        # Code in here will only be executed on boot.
        enabled="$(/usr/bin/batocera-settings-get
system.samba.enabled)"
        if [ "$enabled" = "0" ]; then
            echo "SMB services: disabled"
            exit 0
        fi
        ;;
    stop)
        # Code in here will only be executed on shutdown.
        echo -n "Shutting down SMB services: "
        kill -9 `pidof smbd`
        RETVAL=$?
        rm -f /var/run/samba/smbd.pid
        echo "done"
        ;;
    restart|reload)
        # Code in here will executed (when?).
        echo "SMB services: reloaded"
        ;;
    *)
        # Code in here will be executed in all other conditions.
        echo "Usage: $0 {start|stop|restart}"
        ;;
esac

exit $?
```

services

You should replace `/userdata/system/custom.sh` with `/userdata/system/services/myservice`.

Important notes:

- Make sure that the filename is without the `.sh` extension.
- To debug your services, use: `bash -x batocera-services list`

Additional benefits of this system over `custom.sh`:

- You can now have more than one service running.
- You can enable and disable them individually, from the command line (`batocera-services`) or through the UI in EmulationStation (System settings → Services).

Disabling services (with `batocera-services disable`) doesn't **stop** them. That needs to be done manually with `batocera-services stop`. Disabling them only means that they wil not be automatically started after the next system startup.

Enabling them (with `batocera-services enable`) works the same. To actually **start** a service, enable it first, then restart Batocera or use `batocera-services start`.

Enabling or disabling the service in the EmulationStation user interface will also start or stop the service immediately.

Sample service

myservice

```
#!/bin/bash

case "$1" in
    start)
        echo "I've started."
        ;;
    stop)
        echo "I've stopped."
        ;;
    status)
        echo "This is my status."
        ;;
esac
```

Watch for a game start/stop event

Batocera **5.23** and higher supports running a script right before a game launches and/or after a game is exited. Here are some examples:

- Automatic controller setup
- Automatic scraping for new games
- Change screen resolution (though it might be better to use switchres for this)
- Sync savestates to an external/network device
- ... give me your idea here

Place an executable script (can have any filename) with the correct setted shebang (first line!) or an executable binary into directory `/userdata/system/scripts/`.

Game start/stop event scripts must be marked as executable with the `chmod +x` command. For example:

```
# Open up a text editor to create your script:
nano /userdata/system/scripts/first_script.sh
# Write in your script, save and exit.
# Then to add the executable bit to it:
chmod +x /userdata/system/scripts/first_script.sh
```

This means that if your userdata partition is of the NTFS, exFAT or older file system that are missing the executable bit functionality, it will fail to execute.

You can add as many subfolders as you want, every script placed there will be executed. To distinguish between **START** or **STOP** condition the first argument parsed in the script have either of these flags:

- `gameStart` is passed to your scripts if you select a game from EmulationStation (Game Starts!)
- `gameStop` is passed to your scripts if you are going back from a game to EmulationStation (Game Stops!)

If you do not set cases for first argument, then the script is executed on every start and on every end of a game.

What is parsed

Table of parsed arguments in correct order and their functions:

Arg no.	Parsed Parameter	Usage
1	`gameStart` or `gameStop`	To distinguish between START or STOP condition
2	`systemName`	The system shortname as is in `es_system.cfg`, eg. **atari2600**
3	`system.config['emulator']`	The emulator settings, eg. **libretro**
4	`system.config['core']`	The emulator core you have chosen, eg. **stella**
5	`args.rom`	The full rom path, eg. **/userdata/roms/atari2600/Mysterious Thief, A (USA).zip**

Simple game start/stop script as an example

At first create the directory where the scripts need to be set up. Connect through SSH and type `mkdir /userdata/system/scripts`. After this we can set up our first script by typing `nano /userdata/system/scripts/first_script.sh`. The filename can be anything readable by

bash.

Here's the template script:

first_script.sh

```
#!/bin/bash
# This is an example file of how gameStart and gameStop events can be
used.

# It's good practice to set all constants before anything else.
logfile=/tmp/scriptlog.txt

# Case selection for first parameter parsed, our event.
case $1 in
    gameStart)
        # Commands in here will be executed on the start of any game.
        echo "START" > $logfile
        echo "$@" >> $logfile
    ;;
    gameStop)
        # Commands here will be executed on the stop of every game.
        echo "END" >> $logfile
    ;;
esac
```

Now you can see a log file created `/tmp/scriptlog.txt`, that parsed all arguments in this file. This is just a small test of course. You can check out the SSH page and the usage of batocera-settings page for general and Batocera-specific commands.

EmulationStation scripting

In case Batocera's provided scripting functionality is not sufficient, ES itself also triggers scripts of its own volition. Extra scripts can be created at `/userdata/system/configs/emulationstation/scripts/<event name>/<script>.sh`; unlike regular Batocera scripts the event named will trigger all scripts in the respective event folder. For instance, to have a script execute at every `game-start` event, it must be placed in `/userdata/system/configs/emulationstation/scripts/game-start/`.

All EmulationStation scripts must be marked as executable with the `chmod +x` command. For example:

```
# Open up a text editor to create your script:
nano /userdata/system/configs/emulationstation/scripts/game-
start/first_script.sh
# Write in your script, save and exit.
# Then to add the executable bit to it:
```

```
chmod +x /userdata/system/configs/emulationstation/scripts/game-start/first_script.sh
```

This means that if your userdata partition is of the NTFS, exFAT or older file system that are missing the executable bit functionality, it will fail to execute.

Event name	Arguments	Notes
`game-start`	ROM name `rom`, ROM path `basename` (arguments described below)	When a game starts. Nearly identical to Batocera's `gameStart`, however this doesn't include as much information and only triggers when it's ES itself that starts it.
`game-end`	N/A	When a game ends. Nearly identical to Batocera's `gameStop`, however this only triggers when it's ES itself that ends it.
`game-selected`	`getSourceFileData()→getSystem()→getName()`, `getPath()`, `getName()`	New to Batocera **v33**. Whichever game is currently being hovered over. Includes games shown during the screensaver.
`system-selected`	System `getSelected()→getName()`	New to Batocera **v33**. Whichever system is currently being hovered over in the system list.
`theme-changed`	Theme being switched to `theme_set→getSelected()`, Previous theme `oldTheme`	When a different theme is selected. Mostly used for theme/element reloading.
`settings-changed`	N/A	When a system setting is saved.
`controls-changed`	N/A	When a controller mapping is saved from the **CONTROLLER MAPPING** menu.
`config-changed`	N/A	Whenever any configuration, be it system settings or controller mappings, is changed.

Event name	Arguments	Notes
`quit`	N/A	When the system is told to do a regular shutdown.
`reboot`	N/A	When the system is told to do a reboot.
`shutdown`	N/A	When the system is told to do a fast shutdown.
`sleep`	N/A	When the system is told to sleep.
`wake`	N/A	When the system is told to wake from sleep.
`achievements`	with arguments described below	When a RetroAchievement is unlocked - new to Batocera **v38**
`screensaver-start`	N/A	When the screensaver starts
`screensaver-stop`	N/A	When the screensaver stops (waken up by user)

EmulationStation sends different arguments to these scripts based on the event type. For game-related events:

Arg no.	Parsed parameter	Usage
1	`system`	The current system of the game.
2	`rom`	The filename of the current game.
3	`romname`	The name of the game as specified in its metadata.

All events need to be added here.

Simple EmulationStation script example

This script is an example of an EmulationStation `game-selected` script for controlling the Pixelcade display. More robust results with the Pixelcade may be achieved with the Pixelcade Batocera scripts.

pixelcade.sh

```
#!/bin/bash

# Save the arguments into variables.
system="${1}"
rom="${2}"
romname="${3}"
```

```
# Convert an argument into another value.
if [[ "${system}" == "fbneo" ]]; then
        system="mame"
fi

# Switch case for certain systems.
case ${system} in
  fbneo)
    system="mame"
    ;;
  scummvm)
    rom="${rom%.*}"
    ;;
esac

# Execute this part every time this event triggers.
curl -G \
        --data-urlencode "t=${romname}" \
        http://127.0.0.1:8080/arcade/stream/${system}/`basename ${rom}`
```

Real use cases

Friendly reminder that older examples have now been moved to the legacy scripts page.

Batocera after boot scripts

- Shutdown Batocera at a given time.

custom.sh

```
#!/bin/bash
while true; do
    currenthour=$(date +%k%M)
    if [ $currenthour -eq 2355 ]; then
        shutdown -h now
    fi
    sleep 30
done
```

- Disable a specific network interface (e.g. `eth0`, `eth1`, etc.):

custom.sh

```
#!/bin/bash
ifconfig eth1 down
```

Shut down Batocera after 1 hour of idleness

In that example, we want the Batocera system to shutdown one hour after the screensaver is launched (so that the system shuts down itself automatically after being idle).

To do so, you will need to add two scripts:

- `/userdata/system/configs/emulationstation/scripts/screensaver-start/shutdown-trigger.sh` that is launched when the screensaver starts, and will start a 1 hour timer (see in the script the 3rd line where you can tune that with options possible through the `date` command).
- `/userdata/system/configs/emulationstation/scripts/screensaver-stop/shutdown-cancel.sh` that cancels the automatic shutdown, triggered when you touch the controller.

The script to put in the directory `/userdata/system/configs/emulationstation/scripts/screensaver-start/` is:

shutdown-trigger.sh

```
#!/bin/sh
TRG=/tmp/shutdown.screensaver
date -d "+1 hour" +"%s" > "$TRG"
while true; do
    now=$(date +"%s")
    [ -f "$TRG" ] || break
    trg=$(cat "$TRG")
    if [ "$now" -ge "$trg" ]; then
        shutdown -h now
    rm "$TRG"
    fi
    sleep 5
done
```

And the script to put in the directory `/userdata/system/configs/emulationstation/scripts/screensaver-stop/` is:

shutdown-cancel.sh

```
#!/bin/sh
TRG=/tmp/shutdown.screensaver
rm "$TRG"
```

Batocera event watcher scripts

- Output text to a file

outputsomething.sh

```
#!/bin/bash
# by cyperghost for batocera

testfile="/userdata/system/scripts/output.txt"
echo "END Script!" >> $testfile
echo "Parameter: $@" >> $testfile
echo "Systemname: $1" >> $testfile
echo "Emulatorcore: $2" >> $testfile
echo "ROM: $(basename "$3")" >> $testfile
[[ "$4" == "libretro" ]] && echo "You are free!" >> $testfile || echo
"No hotkey is pure hell eh??" >> $testfile
```

- Change LED colors while running games, and shutdown on button press

shutdownsimple.sh

```
#!/bin/bash
LED1=22
LED2=27
SHUTDOWN=3

case "$1" in
start)
    # LED init
    echo "$LED1" > /sys/class/gpio/export
    echo "$LED2" > /sys/class/gpio/export
    echo out > /sys/class/gpio/gpio$LED1/direction
    echo 0 > /sys/class/gpio/gpio$LED1/value
    echo out > /sys/class/gpio/gpio$LED2/direction
    echo 1 > /sys/class/gpio/gpio$LED2/value

    #Button init
    echo "$SHUTDOWN" > /sys/class/gpio/export
    echo "in" > /sys/class/gpio/gpio$SHUTDOWN/direction
    #This loop continuously checks if the shutdown button was pressed
    #It sleeps as long as that has not happened.
    buttonstate1=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
    shutdownSignal=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
    while [ $shutdownSignal = $buttonstate1 ]; do
        shutdownSignal=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
        sleep 0.5
    done
    shutdown -h now
```

```
;;
stop)
      #unexport all GPIOs
       #echo "$SHUTDOWN" > /sys/class/gpio/unexport
       echo "$LED1" > /sys/class/gpio/unexport
       echo "$LED2" > /sys/class/gpio/unexport
;;
esac
exit $?
```

- Load latest created savestate automatically

loadlatestsave.sh

```
#!/bin/bash

# disable auto load/save state inside ES
# add 'global.retroarch.savestate_auto_load=true'
# to your batocera.conf
# by cyperghost aka lala for BATOCERA
#
# download this script to '/userdata/system/scripts' and set executable
bit
#

rom_no_ext="$(basename "${5%.*}")"
sav_path="/userdata/saves/$2"

[[ "$(batocera-settings get global.autosave)" -eq 1 ]] && exit
[[ "$(batocera-settings get global.retroarch.savestate_auto_load)" ==
"true" ]] || exit

if [[ $1 == "gameStart" ]]; then
    file="$(/bin/ls "$sav_path/$rom_no_ext."* -turR1A | tail -1)"
    [[ -n "$file" ]] || exit
    [[ "${file##*.}" == "png" ]] && file="${file%.*}"
    [[ -f "$file" ]] || exit
    cp -f "$file" "$sav_path/$rom_no_ext.state.auto"
fi

if [[ $1 == "gameStop" ]]; then
    rm -f  "$sav_path/$rom_no_ext.state.auto"
fi
```

- Faff about with audio settings.

audio-shenanigans.sh

```
case "$1" in
```

```
    start)
        # Get the current audio profile.
        defaultprofile="$(batocera-audio get-profile)"
        # Get the current audio device.
        defaultaudio="$(batocera-audio get)"
        # Set the audio device to HDMI-2.
        batocera-audio set HDMI-2
        # Set the default "auto" audio profile.
        batocera-audio set-profile auto

        # Unmute the audio device.
        batocera-audio setSystemVolume unmute
        # Set audio level to 69.
        batocera-audio setSystemVolume 69
        # Test the audio device with Mallet.wav
        batocera-audio test
    ;;
    stop)
        # Toggle the audio mute.
        batocera-audio setSystemVolume mute-toggle
```

* Watch for controller inactivity and do X when inactive

controller-inactivity-checker.sh

```
#!/bin/bash

LOCK="/var/run/controller-inactivity-checker.lock"
if [ -f "$LOCK" ]; then
    exit 1
fi

trap 'rm -f "$LOCK"; exit 0' SIGTERM EXIT
touch "$LOCK"

STATE="active"
JS_DEVICES=()

TIMER="$(/usr/bin/batocera-settings-get
system.controllerinactivitytimer)"
if [[ -z "$TIMER" || ! "$TIMER" =~ ^[0-9]+$ || "$TIMER" -le 60 ]]; then
    TIMER="900" # default in seconds
    /usr/bin/batocera-settings-set system.controllerinactivitytimer
"$TIMER"
fi

js_update() {
    JS_DEVICES=()
    for js_device in /dev/input/js*; do
```

```
        if [ -e "$js_device" ]; then
            JS_DEVICES+=("$js_device")
        fi
    done
}

do_inactivity() {
    STATE="inactive"
    # your code here
}

do_activity() {
    STATE="active"
    # your code here
}

monitor_controllers() {
    local JS_REFRESH_INTERVAL=10 # Number of loops before controller
refresh (by default 1 loop per second)
    LOOP_COUNT=0 # This should always be 0

    while true; do
        if (( LOOP_COUNT % JS_REFRESH_INTERVAL == 0 )); then
            js_update
        fi

        for i in "${!JS_DEVICES[@]}"; do
            js="${JS_DEVICES[$i]}"

        if [ -e "$js" ]; then
            if timeout 1 jstest --event "$js" | tail -n +25 | grep -q
"Event"; then
                LOOP_COUNT=0
                JS_DEVICES=("$js" "${JS_DEVICES[@]:0:$i}"
"${JS_DEVICES[@]:$((i + 1))}")
                if [ "$STATE" = "inactive" ]; then
                    do_activity
                fi
                break
            fi
        fi
        done

        ((LOOP_COUNT++))
        if (( LOOP_COUNT >= TIMER )); then
            if [ "$STATE" = "active" ]; then
                do_inactivity
            fi
        fi
    done
}
```

```
js_update
monitor_controllers

exit 0
```

Batocera boot scripts

Delay Syncthing until after everything else

boot-custom.sh

```
#!/bin/bash
# Delay Syncthing until other boot stuff has completed.

if [[ "$1" != "start" ]] && exit 0; then
  if mv /etc/init.d/S27syncthing /etc/init.d/S99syncthing; then
    echo "Successfully delayed Syncthing."
  elif ls /etc/init.d/S99syncthing; then
    echo "Command has already run!"
    exit 0
  else
    echo "Syncthing failed to be delayed."
    touch /userdata/check-boot-custom-sh
    shutdown -h now
    exit 1
  fi
fi

exit $?
```

EmulationStation scripts

Play a video on a second screen

Original forum post with a deeper explanation. Artwork needs to be sourced and placed in the appropriate `Marquee` and `roms/Marquee` folders first.

Place `game.sh` into `system/configs/emulationstation/scripts/game-selected`

game.sh

```
#!/bin/bash
System=$1 #system name
```

```
Romname=${2%.*} #romname
rom=${Romname##*/}
/userdata/marquee.sh Gameselected $System "$rom"
```

Place `system.sh` into `system/configs/emulationstation/scripts/system-selected`

system.sh

```
#!/bin/bash
System=$1 #System name
/userdata/marquee.sh Systemselected $System &
```

Place `marquee.sh` in `/userdata`

marquee.sh

```
#!/bin/bash

case $1 in
Start)
Romname=$3
Gamepath=$2
marqueeimage=$Gamepath/images/$romname-marquee.png
if [ -f "/userdata/roms/Marquee/videos/$Romname.mp4" ]
then
ffmpeg -i /userdata/roms/Marquee/videos/$Romname.mp4 -vf scale=1280:720
-sws_flags bilinear -pix_fmt rgb565le -f fbdev /dev/fb0

fi

if [ -f "/userdata/roms/Marquee/hires/$Romname.jpg" ]
then
fbv /userdata/roms/Marquee/hires/$Romname.jpg -fer
elif [ -f "$marqueeimage" ]
then
fbv $marqueeimage -fer
else
fbv /userdata/roms/mame/images/mame.png -fer
fi

;;
Gameselected)
System=$2 #system name
Romname=$3 #romname

if [ -f "/userdata/roms/Marquee/$Romname.png" ]
then
fbv /userdata/roms/Marquee/$Romname.png -fer
elif [ -f "/userdata/roms/$System/images/$Romname-marquee.png" ]
```

```
then
fbv "/userdata/roms/$System/images/$Romname-marquee.png" -fer
else
fbv /userdata/roms/Marquee/mame.png -fer
fi

;;

Systemselected)
imagepath="/userdata/roms/sysimages/$2"
if [ -f "$imagepath.png" ]
then
fbv "$imagepath.png" -fer
else
fbv /userdata/roms/mame/images/mame.png -fer
fi

;;

esac
```

Place `script.sh` in `system/scripts`

script.sh

```
#!/bin/bash

case $1 in
gameStart)

gamepath=${5%/*}
romname=${5##*/}
/userdata/marquee.sh Start $gamepath ${romname%.*} &
;;

gameStop)
killall ffmpeg
;;
esac
```