

Batocera services

Services are scripts that run at system startup and shutdown. Services are supported in Batocera **v38** and higher, and will replace the deprecated [custom.sh user script](#).

Two types of services are supported:

1. The **system services** are preinstalled in `/usr/share/batocera/services/`
2. The **user services** can be created and modified by the user in `/userdata/system/services/`

Usage

How to write a proper user service script:

1. Use shell script language like bash or sh
2. Use UNIX LF line termination, not Windows CRLF termination
3. Make the script executable (`chmod +x your_script`, optional)
 - Setting executable permissions is good practice but requires a filesystem with support for those permissions
 - If the executable bit is not set then bash interpreter is used directly, therefore it is optional
4. Store your script in `/userdata/system/services/`
5. Reboot your device to make the services selectable in EmulationStation.

Each of the services may be enabled or disabled in the EmulationStation frontend: **MAIN MENU** → **SYSTEM SETTINGS** → **SERVICES**. Toggling the service state in this menu will also immediately start or stop the service, in addition to controlling whether it will start at next system boot.

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 will 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`.

Filename conventions

For filenames there are some specific rules! This is caused as every service-file is exported as systemvariable. So only characters from A-Z (upper- and lowercase, mixed), the underscore and digits (Please avoid as first character) are allowed.

Spaces, dots, brackets and regional characters like ß,œ or я are not allowed in general and (as said again because of bad surprise!) do not use digits as first sign.

Some file examples:

- Hello -> okay

- Hello_5 -> okay
- Hello.sh -> not allowed
- 5_Hello -> not allowed
- Hello-5 -> not allowed
- Hallöle -> not allowed

You can test your script-names by typing `batocera-services list user`, then you will receive a result-list. Every invalid script name is not used and therefore can not be activated in ES frontend.

```
[root@BATOCERA /userdata/system/services]# batocera-services list user
Hello      -
Hello_5    -
WARNING: Invalid service script name: Hello.sh
WARNING: Invalid service script name: 5_Hello
WARNING: Invalid service script name: Hello-5
WARNING: Invalid service script name: Hallöle
```

Conditions

All these scripts are initiated through `/etc/init.d/S99userservices` so there is a **start** and a **stop** condition that can be used inside the scripts. S99 will wait for all scripts to be finished in stop condition, so be aware of your scripts using sleep timers and infinite do-while loops.

Script Examples

Just some examples to show how powerful these kind of scripts can be.

This script example is able to spawn "sub-services", these spawned scripts have just one single task: Set a single value to `batocera.conf`. The script is furthermore able to detect a setted and active value and to show this in your ES-Service-Section. The usage is simple. Place the script in `/userdata/system/services/` and start the service. Then leave the menu and select the Service-Section again. You'll now see that there are 2 new services are in place: `PI_RETROFLAG` and `PI_RETROFLAG_ADV`. If you select one of these, then the value **RETROFLAG** or **RETROFLAG_ADV** will be written to Batoceras config file. If you activate all the spawned services then the value is used.

By **stopping** the mainservice, all sub-services will be removed from system. I used the `system.power.switch` value here that is reasonable for all Raspberry Pie platforms. By **starting** the mainservice again, all sub-services will be created again. Usually a system reboot is needed to make the powerswitches work.

PI_POWERSWITCHES

```
#!/bin/bash
# Spawn Service Script by cyperghost (second life)
# This is a crude example, but shows the ability of the new services

switch="RETROFLAG RETROFLAG_ADV" #Add here more values
path="/userdata/system/services"
```

```

# First argument for condition: start=service switched on
case ${1} in
start)
    # Create single scripts per echo command and place them to
    /services
    # The test start inside the sub-services is needed to set the
    system.power.switch parameter only if service is ticked
    for i in ${switch};
    do
        echo "[ \${1} == start ] && batocera-settings-set
system.power.switch \${i}" > ${path}/PI_${i}
    done

    # Check and set indicator to on if a proper value is found in
    batocera.conf, if there is no value then indicator is setted to off
    active_service=$(batocera-settings-get system.power.switch)
    if [ -n ${active_service} ];
    then
        batocera-services enable PI_${active_service}
    fi
;;

stop)
    #Remove all entries from filesystem and in batocera.conf (done
    through batocera-services)
    for i in ${switch};
    do
        rm -f ${path}/PI_${i}
        batocera-services disable PI_${i}
    done
;;

esac

```

This script will check for proper filenames and automatically alter them and even make backups. The idea was born by an interesting thread on github, where a user mentioned that downloaded scripts from this wiki got Windows linebreaks. This was just caused by browser download. The question is:

How to fix Windows linebreaks without manually Code editing?

Of course it can be done.... - Run a small Sanatizer script (tools like DOS2UNIX do their best job here)

SANATIZE_SERVICE

```

#!/usr/bin/env -S bash #PROTECTED
# Sanatize Service by crcerror (second life) - or: How to fix Windows
linebreaks without manually code edits?
grep -rlq '\r' "$0" && dos2unix -k -q "$0" && exit 0 # Selfrepair for
called script

```

```
#only on start condition
[[ $1 == stop ]] && exit 0

# Sanatize Windows-CRLF to unix-style
# Sanatize filenames: Use underscore for non allowed characters

pushd /userdata/system/services > /dev/null

find -type f -printf '%f\n' |
while read USER_SERVICE
do
    SANATIZE="${USER_SERVICE//[^\0-9A-Za-z_]/_}"
    SANATIZE="$(echo "$SANATIZE" | sed 's/^[[:digit:]]*//')"
    if [[ "${USER_SERVICE}" != "${SANATIZE}" ]]
    then
        mv -b --suffix=_bak "${USER_SERVICE}" "${SANATIZE}"
        USER_SERVICE="-back2life-"
    fi

    [[ "${USER_SERVICE}" == "-back2life-" ]] &&
USER_SERVICE="${SANATIZE}"
    grep -rlq $\r "${USER_SERVICE}" && dos2unix -k -q
"${USER_SERVICE}"

done

popd > /dev/null
```

Some deeper explanations:

If you download this script in a Windows-Browser and then you copy the file to your Batocera-device, then it is likely that these files are in wrong format and every codeline got a ^M at it's end. Therefore every script-command and codelogic will fail (even the #!/bin/env for calling the command interpreter without setted proper seperator and environment).

As the first real code line is a logic-link-chain all commands will be executed properly. I used two trick here:

1. A remark # my remark to avoid command breaks
2. Setted a bash environiont with proper seperators

Usually the command for the first codeline (line 3) would be `exit 0` but **exitcode 0** is not **exitcode 0^M** and therefore you would see an error like `numeric argument required: exit: 0`. Nevertheless the script is able to "repair" itself and will work properly if you start it up a second time or reboot Batocera twice after script activation through EmulationStation.

From:

<https://www.wiki.batocera.org/> - **Batocera.linux - Wiki**

Permanent link:

https://www.wiki.batocera.org/scripting_services_rules_examples?rev=1729183764

Last update: **2024/10/17 16:49**

