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/`

Each of the services may be enabled or disabled in the EmulationStation frontend: **MAIN MENU → SYSTEM SETTINGS → SERVICES**

Usage

How to write a proper script?

1. Use shell-script language like bash or sh
2. Avoid Windows line endings CRLF, use UNIX encodings instead
3. Make the script executable (`chmod +x your_script`, optional)
    - The executeable bit is good practise but fails on FAT-file systems
    - If executable bit is not set then bash interpreter is used directly, therefore it is optional
4. Store your script to `/userdata/system/services`
5. Reboot your device to make them selectable in EmulationStation.

Disabling services (with `batocera-services disable` or through the UI) 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` or through the UI) 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 specfic rules! This is caused as every service-file is exported as systemvariable. So only charaters from A-Z (upper- and lowercase, mixed), the underscore and digits (Please avoid as first character) are allowed.

Spaces, dots, bracketes and regional characters like ß,œ or я are not allowed in general and (as said again because of bad suprise!) 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

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 an 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 explainations:
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 proberly. I used two trick here:

1. A remark `# my remark` to avoid command breaks
2. Setted a bash envirionment 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.