Batocera Services

This feature was introduces in **Batocera v38 Blue Moon** and is a powerful replacement for the **custom.sh**. We can devide two setups. First the preinstalled services which should not be altered and in second instance the **Userservices** that are stored to directory: `/userdata/system/services`. In both cases the activation state can altered through EmulationStation Frontend: Main Menu –> System Settings –> Services

Even if **custom.sh** is still working (10/2024) it is recommended to use the service section for future proof setups.

Usage

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 rules! This is caused by the handling for this kind of usecase. Every file is exported as systemvariable so only charaters from A-Z and digits 0-9 (not as first letter!) are allowed. Spaces, dots, bracketes and regional chars like ß,œ or я are not allowed.

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` you will receive a result-list of your services.

```
[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, so be aware of your scripts using sleep timers and infinite do-while loops.

As a bonus: These scripts can also be used on FAT-file systems and are started through bash-interpreter, so you don't need them executable through `chmod +x your_service_file` command (even if it would be best practise).

Script Examples

This script will check for proper filenames and automatically alter them and even make backups from your script. There was also an interesting thread, where a user asked if downloaded scripts from this wiki have Windows newlines just forced by browser download and how to avoid them. Of course it an be done.... We can run a small Sanatizer script. With some changes it will also Sanatize the boot scripts for example - imho tools like DOS2UNIX do their best job here.

SANATIZE_SERVICE

```
#!/bin/bash
# Sanatize Service by crcerror (second life)
# Selfrepair first
grep -rlq $'\r' "$0" && dos2unix -k -q "$0" && exit 0

#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
    SANITIZE=${SERVICE//[^0-9A-Za-z_]/}
    SANITIZE=$(echo "$SANITIZE" | sed 's/^[[:digit:]]*//')
    if [[ "${USER_SERVICE}" =! "${SANITIZE}" ]]
    then
        mv -b --suffix=_backup "${USER_SERVICE}" "${SANITIZE}"
        grep -rlq $'\r' "${SANITIZE}" && dos2unix -kq "${SANITIZE}"
    fi

        grep -rlq $'\r' "${USER_SERVICE}" && dos2unix -kq
"${USER_SERVICE}"

done
```

```
popd > /dev/null
```