Customize ES Systems

EmulationStation displays systems based on a file called `es_systems.cfg`. In other distributions, this is typically the only file. However, in Batocera Fix Me! and higher, the `/userdata/system/configs/emulationstation/es_systems_<custom_name>.cfg` can be used as an overlay to the original `/usr/share/emulationstation/es_systems.cfg` file.

If preferred, the entire file can be overridden by instead using `/userdata/system/configs/emulationstation/es_systems.cfg`. Be careful if doing this, as the entire notion of maintaining this file as Batocera upgrades its internals (namely Python) thus rely on you. This is really only suitable if you never intend on updating Batocera again.

Whenever Batocera is upgraded, its version of Python used may also change. This means if you are using an `es_systems.cfg` which manually defines the <command> (for example: <command>/usr/lib/python2.7/site-packages/configgen/emulatorlauncher.py</command> which is calling an older Python version), it will need to be updated accordingly.

In general, it is recommended to avoid using a custom <command> and to just use the default one.

Modify an existing system with a new system-specific "es_systems_<custom_name>.cfg" file

You can create a file named `es_systems_<custom_name>.cfg` where <custom_name> is a name you wish to use for it. This file needs to follow the same conventions as the original `es_systems.cfg`, but does not need to include the entirety of the file. For example:

es_systems_3do.cfg

```
<?xml version="1.0"?>
<systemList>
  <!-- This line is a comment, not necessary for the system. These will explain things, and can be safely removed. -->
  <system>
        <!-- The full name of the system, the one that will appear in menus and such. -->
        <fullname>3DO Interactive Multiplayer</fullname>
        <!-- The short name, the one used for the path to the system's
```

```
ROMs and other internal uses. -->
        <name>3do</name>
        <!-- Metadata. The company/companies that made the system, also
known as "Brand". Metadata tags like this aren't truly necessary, but
can make organization easier. -->
        <manufacturer>Panasonic - Sanyo - Goldstar</manufacturer>
        <!-- Metadata. Release date. -->
        <release>1993</release>
        <!-- The type of hardware. Batocera doesn't particularly use
this for anything, but it might be useful to specify here for future
changes. -->
        <hardware>console</hardware>
        <!-- Extremely important, the path to look for this system's
ROMs in. This should always start with /userdata/roms/. -->
        <path>/userdata/roms/3do</path>
        <!-- The file extensions of ROMs that should be scanned into
ES's gamelist.xml when the user rescans their gamelist. Delimited by
whitespaces. You must include the period mark (.) and proper
capitalization. -->
        <extension>.iso .chd .cue</extension>
        <!-- Ordinarily this is the command line used to directly run
the program, but Batocera relies on config generators to do so. This
simply calls that generator, along with some other information. -->
        <command>python /usr/lib/python3.9/site-
packages/configgen/emulatorlauncher.py %CONTROLLERSCONFIG% -system
%SYSTEM% -rom %ROM%</command>
        <!-- The platform to use when scraping for metadata for this
system's games. The full list of available platforms are in
https://github.com/batocera-linux/batocera-emulationstation/blob/master
/es-app/src/PlatformId.cpp -->
        <platform>3do</platform>
        <!-- The theme to load from the current theme-set, if
supported. Usually, this is identical to the shortname. Check your
theme's readme for more info. -->
        <theme>3do</theme>
        <emulators>
        <!-- These are the emulators used for the system. This is
handled entirely by Batocera, and is not necessary for a custom es
systems, but is shown here for reference. -->
            <emulator name="libretro">
                <cores>
                    <core default="true">opera</core>
                </cores>
            </emulator>
        </emulators>
  </system>
</systemList>
```

would only affect the 3DO system on your Batocera, with the rest of the systems still referring to `/usr/share/emulationstation/es_systems.cfg` for their system CFG. When both files contain

the same <name>, `es_systems_<custom_name>.cfg` will take priority.

In case you'd like to have complete control over ES systems CFG, you can still use the two previous methods to completely override it (ie. using overlays or copying the entirety of `es_systems.cfg` to `/userdata/system/configs/emulationstation/es_systems.cfg` without appending a custom system name to its filename).

Batocera is smart enough to *only* apply the changes you make. Such as, if the only thing you need to change for a system is its ROM path, you can use the following in your `/userdata/system/configs/emulationstation/es_systems_pico8.cfg`:

es_systems_pico8.cfg

```
<?xml version="1.0" encoding="UTF-8"?>
<systemList>
  <system>
        <name>pico8</name>
        <path>/userdata/roms/pico8real</path>
  </system>
</systemList>
```

Here, it'll use the folder `/userdata/roms/pico8real/` to look for Pico-8 games instead of the regular one, and use the rest of the configuration from the `/usr/share/emulationstation/es_systems.cfg` file.

Create a new system

The method above can also be applied to make a brand new system in place of editing an already existing one. For instance, you might want to create a new “megadrivehacks” system intended for Megadrive/Genesis ROMhacks.

Create a `/userdata/system/configs/emulationstation/es_systems_megadrivehacks.cfg` file that contains the following:

es_systems_megadrivehacks.cfg

```
<?xml version="1.0"?>
<systemList>
  <system>
        <!-- The custom name of this system. -->
        <fullname>Megadrive hacks</fullname>
        <!-- Keep the name the same for the system you are basing it
off it, saving you work later. -->
        <name>megadrive</name>
        <manufacturer>Sega</manufacturer>
        <release>1988</release>
        <hardware>console</hardware>
```

```
        <!-- Specify your new path. -->
        <path>/userdata/roms/megadrivehacks</path>
        <extension>.bin .gen .md .sg .smd .zip .7z</extension>
        <command>python /usr/lib/python3.9/site-
packages/configgen/emulatorlauncher.py %CONTROLLERSCONFIG% -system
%SYSTEM% -rom %ROM%</command>
        <!-- Since this is used only for scraping, it's safe to leave
it as default. You'll probably have to be providing your own metadata
for ROMhacks anyway. -->
        <platform>megadrive</platform>
        <!-- Change the theme to use the custom assets instead. -->
        <theme>megadrivehacks</theme>
        <!-- Leave the emulators the same. -->
        <emulators>
            <emulator name="libretro">
                <cores>
                    <core>blastem</core>
                    <core default="true">genesisplusgx</core>
                    <core>genesisplusgx-wide</core>
                    <core>picodrive</core>
                </cores>
            </emulator>
        </emulators>
  </system>
</systemList>
```

The <name> and <platform> paths in this example are kept the same such that they can still use the defaults as specified by Batocera. If you really want to use a unique <name>, you'll have to manually specify your default emulator in `batocera.conf`. For instance if you were defining a new system with <name>cps1</name> then the following would be added to `batocera.conf`:

```
cps1.emulator=libretro
cps1.core=fbneo
```

These default emulators can also be defined in the `/usr/share/batocera/configgen/configgen-defaults.yml` and/or `/usr/share/batocera/configgen/configgen-defaults-arch.yml` system defaults file.

You must use a unique <name> field when adding a new system if you wish to keep the old one in addition. However, if using duplicate shortnames is a necessity and you still want the new and old system to exist simultaneously, you can also put both systems in a single CFG file.

Despite <core default="true"> being specified in the `es_systems_megadrivehacks.cfg` file, Batocera doesn't actually use this as the

default when launching an emulator. It instead looks for its own list of defaults for its configgen.

In case you're making a brand new system with brand new emulators/parameters, refer to the developer documentation instead.

Older Batocera versions

Click to reveal

EmulationStation displays systems based on a file called `es_systems.cfg`, located at `/usr/share/emulationstation`. In the past, you had to edit this file and use the command `batocera-save-overlay`. This had issues when updating as all overlays have to be removed, meaning you'd have to redo these steps every update.

Batocera Fix Me! introduced the ability to copy the file to `/userdata/system/configs/emulationstation/es_system.cfg` and edit it there, however this meant you would have to compare and copy the file every update, as well as appending your changes and working out if something had broken (as Batocera adds new systems nearly every major release and changes the launch methods for other systems too).