30.09.2021 // Config Docs

BFFH uses DHALL for Config-File structure BFFH uses RBAC for access control

General BFFH Config

General BFFH Config is in `bffh.dhall` file.

listens

Contains the Addresses BFFH is listen for Connection for the API Default Port for BFFH is `59661`

Example:

```
listens =
[
   { address = "127.0.0.1", port = Some 59661 }
]
```

mqtt_url

Contains the Address for the MQTT Server BFFH connects to **Example:**

```
mqtt_url = "tcp://localhost:1883"
```

db_path

Contains the Path for the internal Database BFFH uses. BFFH will create two files: `<db_path>` and `<db_path>-lock`. Make sure that BFFH has write access in the relevant directory **Example:**

```
db_path = "/tmp/bffh"
```

Permissions

BFFH uses a Path-style string as permission format, separated by ".". So for example `this.is.a.permission` consists of the parts `this`, `is`, `a` and `permission`. When requireing permissions, such as in machines you always need to give an exact permission, so for example `test.write`. When granting permissions, such as in roles you can either give an

exact permission or you can use the two wildcards `*` and `+`. These wildcards behave similar to regex or bash wildcards:

- `*` grants all permissions in that subtree. So, `perms.read.*` will match for any of:
  - `perms.read`
  - `perms.read.machineA`
  - `perms.read.machineB`
  - `perms.read.machineC.manage`
- `+` grants all permissions below that one. So, `perms.read.+` will match for any of:
  - `perms.read.machineA`
  - `perms.read.machineB`
  - `perms.read.machineC.manage`
  - **but not** `perms.read`

Wildcards are probably most useful if you group you machines around them, e.g. your 3D-printers and your one bandsaw require:

1. Write permissions
   - `machines.printers.write.prusa.sl1`
   - `machines.printers.write.prusa.i3`
   - `machines.printers.write.anycubic`
   - `machines.bandsaws.write.bandsaw1`
2. Manage permissions
   - `machines.printers.manage.prusa.sl1`
   - `machines.printers.manage.prusa.i3`
   - `machines.printers.manage.anycubic`
   - `machines.bandsaws.manage.bandsaw1`
3. Admin permissions
   - `machines.printers`
     - For all printers
   - `machines.bandsaws`
     - For all bandsaws

And you then give roles permissions like so:

- Use any 3D printer:
  - `machines.printers.write.+`
- Only allow use of the "cheap" printers
  - `machines.printers.write.anycubic.*`
  - `machines.printers.write.prusa.i3`
- Allow managing of printers:
  - `machines.printers.+`

- Allow administrating printers:
  - `machines.printers.*`

This way if you buy a different anycubic and split the permissions to e.g.

- `machines.printers.write.anycubic.i3`
- `machines.printers.write.anycubic.megax`

It still works out.

Machine Config

Machine Config is in `machine.dhall` file.

`machines`

Contains list of machines

Machines have different perission levels to interact with:

- disclose: User can see the machine in machine list
- read: User can read information about the machine and there state
- write: User can use the machine
- manage: User can interact with the machine as Manager (Check, ForceFree, ForceTransfer)

Example:

```
machines =
{
  Testmachine =
  {
    name = "Testmachine",
    description = Some "A test machine",
    disclose = "lab.test.read",
    read = "lab.test.read",
    write = "lab.test.write",
    manage = "lab.test.admin"
  }
}
```

Roles Config

Roles Config is in `roles.dhall` file.

roles

Contains list of roles

Roles have a list of permission and can be inherited. Permission can be wildcard in permission list.

Example:

```
roles =
{
   testrole =
   {
      permissions = [ "lab.test.*" ]
   },
   somerole =
   {
      parents = ["testparent"],
      permissions = [ "lab.some.admin" ]
   },
   testparent =
   {
      permissions =
      [
         "lab.some.write",
         "lab.some.read",
         "lab.some.disclose"
      ]
   }
}
```

Actors Config

Actors Config is in `actors.dhall` file.

actors

Contains list of actors Actors are defined by a module and one or more paramters

Currenty supported actors: **Shelly** Parameters: id = ID of the Shelly

Process Parameters: cmd = Path of executable args = Arguments for executable

Example:

```
actors =
{
   Shelly_1234 = { module = "Shelly", params =
   {
      id = "12345"
   }},
   Bash = { module = "Process", params =
   {
      cmd = "./examples/actor.sh",
      args = "your ad could be here"
   }}
}
```

actor_connections

Connects the actor with a machine A machine can have multiple actors **Example:**

```
actor_connections =
[
   { _1 = "Testmachine", _2 = "Shelly_1234" },
   { _1 = "Another", _2 = "Bash" },
   { _1 = "Yetmore", _2 = "Bash2" }
]
```

Version #2
Erstellt: 15 Oktober 2024 10:21:42 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 14 Dezember 2024 18:23:07 von Mario Voigt (Stadtfabrikanten e.V.)