Cheat Sheet - Wichtigste Befehle (Übersicht)

Auf dieser Seite finden sich wichtige Befehle im Zusammenhang mit der Benutzung von Difluoroborane.

bffh Daemon (bffhd) Befehlsübersicht (`-h`, `--help`)

```
/usr/bin/bffhd --help
```

```
difluoroborane 0.4.4



USAGE:
    bffhd [OPTIONS]

OPTIONS:
    -c, --config <config>            Path to the DHALL config file to use
        --check                      Check DHALL config for validity
        --dump-db <FILE>             Dump all internal databases (states and users) to the
given
                                     file as TOML
        --dump-users <FILE>          Dump the users db to the given file as TOML
        --force                      Force owerwriting existing files
    -h, --help                       Print help information
        --load-db <FILE>             Load values from TOML into the internal databases
        --load-users <FILE>          Load users from TOML into the internal databases
        --log-format <log format>    Use an alternative log formatter. Available: Full,
Compact,
                                     Pretty [possible values: Full, Compact, Pretty]
        --log-level <log level>      Set the desired log levels. [possible values: info, warn,
                                     error, debug, trace]
        --print-default              Print a default DHALL config to stdout instead of running
        --quiet                      Decrease logging verbosity
        --tls-key-log [<PATH>...]    Log TLS keys into PATH. If no path is specified the value
of
                                     the envvar SSLKEYLOGFILE is used.
    -v, --verbose                    Increase logging verbosity. Stackable from -v up to -vvv
    -V, --version                    Print version information
```

Logging-Konfiguration (`--log-level`, `--log-format`, `--quiet`, `-v`, `--verbose`, `--tls-key-log`)

Log-spezifische Parameter inklusive Audit Log sind zusammenfassend in Server Logs konfigurieren genauer erklärt.

Alle internen Datenbanken exportieren (`--dump-db <bffh-db.toml file>`)

BFFH speichert zwei Datenbanken intern ab und trennt sie auf in Benutzer (users) und Zustände (states).

> Die BFFH Datenbank muss bei jedem Versionsupgrade exportiert und sauber importiert werden.

Dazu muss `--config <Pfad zu bffh.dhall>` angegeben werden, damit bffhd weiß, welche Datenbank angefragt werden soll.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --dump-db <bffh-db.toml file>
```

Soll eine bereits existierende Datei überschrieben werden, nutzen wir zusätzlich den optionalen Parameter `--force`.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --dump-db <bffh-db.toml file> --force
```

> Wird kein Dateiname hinter `--dump-db` angegeben, wird die Datei automatisch `bffh-db.toml` benannt

Datenbeispiel eines Dumps

Dump-Beispiel anzeigen

Ein Datenbank-Dump sieht wie folgt aus und enthält sowohl die Benutzer, als auch die States.

```
[users."Raum 1 Manager"]
roles = ["zam_raum1_manager"]
```

```
passwd =
"$argon2i$v=19$m=4096,t=3,p=1$aE7DYpmOPy+ZAB305S26iQ$G+cx4wEQzaVsB4Vq05+mvvxBgqXlYnejbzpLc
K24SPg"

[users.Werkstattleiter]
roles = ["_manager_schichtleitung"]
passwd =
"$argon2i$v=19$m=4096,t=3,p=1$nqY/EsDGzlwLzRgtZQUBzA$a55mDPB20CxYixvafyYGRIZH/EsPBguzhTBm7
O3D3QA"

[users.Admin]
roles = ["zam_raum1_ecke1_user", "zam_raum1_ecke2_user", "zam_raum1_ecke3_user",
"zam_raum1_ecke4_user", "zam_raum1_ecke5_user", "zam_raum1_ecke6_user",
"zam_raum1_ecke7_user", "zam_raum1_ecke8_user", "zam_raum1_ecke9_user",
"_manager_schichtleitung", "Admin", "zam_raum1_manager"]
passwd =
"$argon2i$v=19$m=4096,t=3,p=1$Ykyx7xGXwWKPMP7Q5FysBA$lbMnVRwZZheYt5u2kEZYuwkWW8DwaHF/JNgqH
791WdQ"
[state.zam-raum1-ecke7-random3."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke2-arrow."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke5-random1."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke6-random2."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke9-shutdown."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke8-macgyver."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke10-restartbffh."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"

[state.zam-raum1-ecke10-restartbffh."1.3.6.1.4.1.48398.612.2.4".previous]
id = "Admin"
[state.zam-raum1-ecke3-fan."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
[state.zam-raum1-ecke1-lamp."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
```

```
[state.zam-raum1-ecke4-mesh."1.3.6.1.4.1.48398.612.2.4"]
state = "Free"
```

Alle internen Datenbanken importieren (`--load-db <bffh-db.toml file>`)

Dieses Kommando lädt die angegebene `*.toml` Datei in die internen Datenbanken innerhalb `bffh.db` hinein.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --load-db <db.toml file>
```

Benutzerdatenbank exportieren (`--dump-users <users.toml file>`)

Einmal importiere Nutzerdaten können genauso wieder aus der bffh Datenbank exportiert werden. Dazu muss `--config <Pfad zu bffh.dhall>` (oder in kurz `-c`) angegeben werden, damit bffhd weiß, welche Datenbank (`db_path`) angefragt werden soll. Außerdem muss auch der Ausgabepfad für `--dump-users` angegeben werden.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --dump-users /etc/bffh/users.toml
```

Soll eine bereits existierende Datei überschrieben werden, nutzen wir zusätzlich den optionalen Parameter `--force`.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --dump-users /etc/bffh/users.toml --force
```

> Wird kein Dateiname hinter `--dump-users` angegeben, wird die Datei automatisch `users.toml` benannt

Benutzerdatenbank importieren (`--load-users <users.toml file>`)

Dieses Kommando lädt die angegebene `*.toml` Datei in die Benutzerdatenbank innerhalb `bffh.db` hinein.

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --load-users <users.toml file>
```

Es kann auch ein Alias `--load` verwendet werden, der das gleiche bewirkt:

```
/usr/bin/bffhd -c /etc/bffh/bffh.dhall --load <users.toml file>
```

> Praxistipps zum smarten Umgang mit der `users.toml` findest du unter Nutzerdatenbank laden / hashen / prüfen.

> Das Laden einer `users.toml` Datei tauscht alle Nutzer in der internen BFFH-Datenbank aus. Etwaige, nicht gesicherte Nutzer, die per App angelegt wurden, werden somit unwiederbringlich verworfen! Hier ist ein ggf. ein Diff zwischen der dann vorher zu sichernden `users.toml` (kann per `--dump-users` erzeugt werden) und der neuen `users.toml` lohnenswert. Ein geeignetes Diff-Werkzeug ist beispielweise Meld.

Konfigurationsdatei prüfen (`--check`)

Dieser Parameter prüft die angegebene Konfigurationsdatei auf fehlende oder fehlerhafte Angaben und überprüft generell, ob die *.dhall Datei geparsed werden kann. Sobald eine z.B. eine eckige oder geschweifte Klammer, ein Hochkomma oder normales Komma fehlt, gibt es in der Regel Probleme. Für das Prüfen muss ebenso `--config <Pfad zu bffh.dhall>` angegeben werden.

Die Ausgabe ist granuliert (Fehler werden mit Zeile und Spalte angezeigt) und sieht zum Beispiel so aus:

```
failed to parse config:  --> 2:2
  |
2 | □let mqtt_password =
  | □^---
  |
  = expected any_label_or_some or empty_record_literal
```

Konfigurationsstandard ausgeben (`--print-default`)

```
/usr/bin/bffhd --print-default
```

Dieser Befehl gibt eine minimale Beispielkonfiguration im Dhall-Format aus, die wir direkt in eine `bffh.dhall` Datei pipen können. Die ausführliche Erläuterung der bffh-Konfiguration findest du hier.

```
{ actor_connections = [{ actor = "actor_123", machine = "resource_a" }], actors = { actor_123 = { module = "Shelly", params = {=} } }, auditlog_path = "/var/log/bffh/audit.json", certfile = "/etc/bffh/certs/bffh.crt", db_path = "/var/lib/bffh/bffh.db", init_connections = [{ initiator = "initiator_123", machine = "resource_a" }], initiators = { initiator_123 = { module = "Process", params = { args = "", cmd = "echo" } } }, instanceurl = "https://fabaccess.sample.space", keyfile = "/etc/bffh/certs/bffh.key", listens = [{ address = "127.0.0.1" }], machines = { resource_a = { category = Some "A category", description = Some "A description", disclose = "lab.some.disclose", manage = "lab.some.manage", name = "Resource A", prodable = True, read = "lab.some.read", wiki = Some "https://some.wiki.url", write = "lab.some.write" }, resource_b = { category = Some "A category", description = Some "A description", disclose = "lab.some.disclose", manage = "lab.some.manage", name = "Resource B", read = "lab.some.read", wiki = Some "https://some.wiki.url", write = "lab.some.write" } }, mqtt_url = "mqtt://127.0.0.1:1883", roles = { admin = { permissions = ["bffh.users.info", "bffh.users.manage", "bffh.users.admin"] }, member = { permissions = ["lab.some.disclose", "lab.some.read", "lab.some.write", "lab.some.manage"] } }, spacename = "fabaccess.sample.space" }
```

Sofern das Paket `dhall` installiert wurde, kann auch die Ausgabe sauber formatiert werden:

```
/usr/bin/bffhd --print-default | dhall format > bffh.dhall
```

```
{ actor_connections = [ { actor = "actor_123", machine = "resource_a" } ]
, actors.actor_123 = { module = "Shelly", params = {=} }
, auditlog_path = "/var/log/bffh/audit.json"
, certfile = "/etc/bffh/certs/bffh.crt"
, db_path = "/var/lib/bffh/bffh.db"
, init_connections = [ { initiator = "initiator_123", machine = "resource_a" } ]
, initiators.initiator_123 =
  { module = "Process", params = { args = "", cmd = "echo" } }
, instanceurl = "https://fabaccess.sample.space"
, keyfile = "/etc/bffh/certs/bffh.key"
, listens = [ { address = "127.0.0.1" } ]
, machines =
  { resource_a =
```

```
    { category = Some "A category"
    , description = Some "A description"
    , disclose = "lab.some.disclose"
    , manage = "lab.some.manage"
    , name = "Resource A"
    , prodable = True
    , read = "lab.some.read"
    , wiki = Some "https://some.wiki.url"
    , write = "lab.some.write"
    }
  , resource_b =
    { category = Some "A category"
    , description = Some "A description"
    , disclose = "lab.some.disclose"
    , manage = "lab.some.manage"
    , name = "Resource B"
    , read = "lab.some.read"
    , wiki = Some "https://some.wiki.url"
    , write = "lab.some.write"
    }
  }
, mqtt_url = "mqtt://127.0.0.1:1883"
, roles =
  { admin.permissions =
    [ "bffh.users.info", "bffh.users.manage", "bffh.users.admin" ]
  , member.permissions =
    [ "lab.some.disclose"
    , "lab.some.read"
    , "lab.some.write"
    , "lab.some.manage"
    ]
  }
, spacename = "fabaccess.sample.space"
}
```

Version anzeigen (`--version`, `-V`)

Folgendes Kommando kann genutzt werden, um die von dir derzeitig verwendete Version von BFFH auszulesen:

```
/usr/bin/bffhd -V
```

```
diflouroborane 0.4.3
```

Wer BFFH als Debian-Paket installiert hat, kann es auch wie folgt herausfinden:

```
dpkg -l | grep fabaccess-bffh
```

```
ii  fabaccess-bffh    0.4.3    amd64    FabAccess Diflouroborane Server (bffh)
```

Wer den genauen Feature Branch auslesen will, von dem kompiliert wurde:

```
VERSION_DETAIL=$(BFFH_LOG=trace timeout 1s /usr/bin/bffhd | grep "branch:") && echo -e
$VERSION_DETAIL | tail -n +2
```

```
pkg_version:0.4.4
branch:feature/cardloginwithtoken
commit_hash:2d127d5c
build_time:2025-03-11 01:30:34 +01:00
build_env:rustc 1.84.1 (e71f9a9a9 2025-01-27),stable-x86_64-unknown-linux-gnu"
```

FabFire Provisioning Tool

Die Kommandos für dieses kleine Tools sind separat in FabFire Tools beschrieben.

Helfer-Skripte

Diverse Helfer-Scripts, die verschiedene Optionen/Parameter automatisieren (z.B. Benutzerdankbank zuverlässig sichern) finden sich in der Script-Sammlung.

Ein paar Tipps für Git

Für BFFH nutzen wir diverse Git-Repositories, die teilweise externe Abhängigkeiten laden. Hier finden sich ein paar Tipps für den Alltag.

Git Repository ausgecheckt, aber Untermodule vergessen auszuchecken?

Wer z.B. das Repo von BFFH auscheckt, muss die externen Module laden, um das Projekt erfolgreich zu kompilieren. Wenn das vergessen wurden, ist das kein Problem, das geht auch nachträglich:

```
git submodule update --init --recursive
```

Und reguläre Updates vom Projekt inklusive Untermodulen dann mit:

```
git pull --recurse-submodules
```

Untermodule updaten

```
git submodule update --remote --merge
```

Version #47
Erstellt: 2024-11-06 00:14:16 CET von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-03-17 16:33:21 CET von Mario Voigt (Stadtfabrikanten e.V.)