Mode Opératoire

Contrôleur UniFi en Docker bornes Wi-Fi U7 Pro Wall

Code :	MO-NET-006
Version :	1.3
Date :	20 septembre 2026
Auteur :	Cédric LEGRAND
Classification :	USAGE INTERNE — Équipe BTS SIO

Historique des révisions

Version	Date	Modifications
1.0	01/09/2026	Création initiale, validée par le déploiement effectif du contrôleur (version 10.6.101) et l'exposition de l'interface via Traefik.
1.1	04/09/2026	Création du WLAN `BTS SIO Etudiants` (WPA2/WPA3, PSK en coffre) et tentative d'adoption : les bornes, gérées par une autre console, restent en *Connection Interrupted* — adoption en attente de leur réinitialisation (voir phase 7).
1.2	10/09/2026	Adoption effective des trois bornes (i111 le 07/09, i109 et i110 de retour en ligne le 10/09) et mise en production du Wi-Fi pédagogique : WLAN `BTS SIO Etudiants` basculé en WPA2/WPA3-Enterprise (802.1X) sur le VLAN 240, création du SSID `BTS SIO Profs` (VLAN 241) et du profil RADIUS `NPS-DC1` (authentification NPS, accounting RSSO `10.0.112.20:1813`). Architecture complète : **MO-NET-008**.
1.3	20/09/2026	UI basculée derrière la porte Authentik (forwardAuth, login local conservé) ; accès direct `10.0.112.228:8443` restreint par règles flottantes OPNsense ; login du compte local passé à `admin@bts.sio` (CLI MongoDB).

1 Objet

Ce mode opératoire décrit le déploiement d'un **contrôleur UniFi auto-hébergé** sur l'infrastructure BTS SIO, nécessaire à l'administration des bornes Wi-Fi **Ubiquiti UniFi U7 Pro Wall** installées dans les salles S109, S110 et S111. Il couvre la chaîne complète : repérage des bornes sur le réseau, création de la stack Docker (application + base MongoDB) sur le serveur de conteneurs `docker-srv`, exposition de l'interface web via le reverse proxy Traefik, initialisation du contrôleur, mise en coffre du compte administrateur et adoption des bornes.

Une borne UniFi ne s'administre pas seule : elle exige une application de contrôle (*UniFi Network*) joignable en permanence [3]. Le présent déploiement retient la voie **conteneurisée** demandée par l'équipe pédagogique, au moyen de l'image communautaire de référence `linuxserver/unifi-network-application` [1]. Le choix d'architecture et ses alternatives sont justifiés en section 3.

2 Champ d'application

Public concerné	Administrateurs de l'infrastructure BTS SIO
Équipements gérés	3 bornes Ubiquiti UniFi U7 Pro Wall (WiFi 7, PoE+, uplink 2,5 GbE) [5]
Bornes déployées	`BTSSIOI109` (`10.0.230.2`), `BTSSIOI110` (`10.0.230.3`), `BTSSIOI111` (`10.0.230.5`) — salles S109, S110, S111
Plateforme cible	CT LXC 200 `docker-srv` (Debian 13, Docker 29) sur l'hyperviseur ProxMox `10.0.112.200`
Adresse du contrôleur	`10.0.112.228` (IP dédiée macvlan), UI `https://unifi.docker.bts.sio`
Accès requis	SSH vers ProxMox (alias `bts-proxmox`), coffre Vaultwarden de l'équipe
Durée estimée	45 à 60 minutes (hors téléchargement des images, ≈ 2,3 Go)
Référence appliquée	Déploiement validé le 01/09/2026, UniFi Network Application **10.6.101**

3 Choix d'architecture

3.1 Pourquoi pas UniFi OS Server ?

Ubiquiti propose depuis 2025 **UniFi OS Server**, présenté comme le nouveau standard de l'auto-hébergement [4]. C'est la voie suggérée par l'enseignant ayant installé les bornes. Elle est écartée ici pour une raison bloquante : la documentation officielle précise qu'UniFi OS Server **ne peut pas s'exécuter dans un conteneur Docker** — *ń This is a complete solution that requires certain services to be run on the host ż* [2]. L'installateur natif exige un Debian 13/Ubuntu 24.04 dédié, pilote lui-même des charges podman et monopolise plusieurs ports de l'hôte.

Deux voies étaient donc possibles :

	A. Docker (retenue)	**B. UniFi OS Server natif**
Logiciel	UniFi Network Application (branche classique 10.x)	UniFi OS Server 5.x (officiel)
Conteneur	Oui (image linuxserver + MongoDB)	Non (installateur hôte, podman piloté)
Emplacement	CT 200 `docker-srv` existant	Nouveau CT/VM Debian 13 dédié
Fonctions UniFi OS	Non (Organizations, Site Magic absentes)	Oui
Pérennité	Branche ń legacy ż encore maintenue	Voie stratégique d'Ubiquiti

Pour trois bornes en environnement pédagogique, les fonctions exclusives d'UniFi OS sont sans usage et la voie A s'intègre à l'outillage existant (Portainer, Traefik, sauvegardes vzdump du CT). La voie B reste documentée comme alternative [2].

3.2 Adresse IP dédiée en macvlan

Le contrôleur écoute sur les ports 8443 (UI), 8080 (*inform*), 3478/udp (STUN) et 10001/udp (découverte) [1]. Or le port 8080 de `docker-srv` est déjà occupé par Netbox, et la découverte de couche 2 traverse mal la translation de ports. La stack reçoit donc sa propre adresse IP `10.0.112.228` sur un réseau **macvlan** adossé à `eth0` du CT : tous les ports du conteneur sont joignables directement, sans conflit ni NAT, et les bornes (même segment L2, réseau `10.0.0.0/16` plat) peuvent contacter l'*inform*.

Limite connue du macvlan

Par conception, l'hôte (`docker-srv`) ne peut pas joindre l'IP macvlan de ses propres conteneurs : les tests depuis le CT doivent cibler le réseau interne de la stack ou être faits depuis un autre poste du LAN. Tous les autres équipements du réseau joignent `10.0.112.228` normalement.

3.3 Exposition web via Traefik

Conformément à la convention de la plateforme, l'interface d'administration est publiée par le reverse proxy Traefik sous `https://unifi.docker.bts.sio` (certificat wildcard auto-signé `*.docker.bts.sio`, enregistrement DNS wildcard existant). Le backend UniFi n'exposant l'UI qu'en **HTTPS auto-signé** sur 8443, un transport `insecureSkipVerify` est ajouté à la configuration dynamique de Traefik. Le flux *inform* des bornes, lui, continue de viser directement `http://10.0.112.228:8080/inform` sans passer par le proxy.

3.4 Protection de l'accès (20/09/2026)

Depuis le 20/09/2026, l'accès à l'UI est protégé à deux niveaux (chantier « UniFi derrière la porte Authentik », spec `docs/superpowers/specs/2026-09-20-unifi-porte-authentik-design.md`) :

— **Porte Authentik sur le FQDN** : le routeur Traefik `unifi` porte le middleware `authentik@docker` (forwardAuth, provider proxy `unifi` de l'outpost embarqué, policy « membre GG-AdminsPlateforme » — voir **MO-PLT-025**). L'accès à `https://unifi.docker.bts.si` exige donc d'abord le compte AD, **puis** le login local du contrôleur (double saisie, limite produit Ubiquiti : pas de délégation d'authentification d'administration).

— **Filtrage de l'accès direct IP** : l'IP macvlan `10.0.112.228:8443` contourne par construction le reverse proxy (Traefik route par en-tête `Host`). Deux règles flottantes OPNsense la restreignent aux sources d'administration : PASS TCP 8443 depuis l'alias `UniFi_Console_Admin_Sources` (VLAN admin `10.0.112.0/24` + VPN collègues `10.10.30.0/24`), puis BLOCK journalisé pour toute autre source. Les ports `8080` (inform) et `3478/udp` (STUN) restent ouverts — les bornes en dépendent.

Le compte local unique du contrôleur utilise le login `admin@bts.sio` (modifié le 20/09/2026 en CLI MongoDB, nom d'affichage `Administrateur` et mot de passe inchangés, item coffre à jour).

4 Prérequis

Prérequis

— **Bornes raccordées et alimentées en PoE+ (voyants allumés), adresses DHCP distribuées par les contrôleurs de domaine**

— **Accès SSH à l'hyperviseur (alias `bts-proxmox`) et droits `pct exec` sur le CT 200**

— **Une adresse IP libre dans le plan d'adressage (vérifiée par ping et table ARP) — `10.0.112.228` retenue, à référencer dans Netbox (MO-PLT-017)**

— **Le coffre Vaultwarden de l'équipe (MO-PLT-009) pour y enregistrer le compte administrateur**

— **Poste d'administration avec `curl`, `nmap`, `jq` et le client Bitwarden `bw`**

Secrets

Aucun secret ne doit figurer dans le fichier `docker-compose.yml` : les mots de passe MongoDB sont générés aléatoirement et stockés dans `/opt/docker/unifi/.env` (`chmod 600`), sur le serveur uniquement. Le compte administrateur du contrôleur est, lui, conservé dans Vaultwarden.

5 Procédure

5.1 Phase 1 — Repérage des bornes sur le réseau

1 Balayer les plages DHCP et relever les adresses MAC

Depuis un poste connecté au réseau BTS (RJ45), balayer les plages DHCP pédagogiques puis relever la table de voisinage :

```
nmap -sn -T4 10.0.230.0/23 -oG - | grep 'Status: Up'
ip neigh show dev enp45s0 | grep -v -E 'INCOMPLETE|FAILED'
```

Identifier les équipements Ubiquiti par leur OUI constructeur. Ici : trois adresses MAC en `58:d6:1f:xx` (OUI `58:D6:1F` = Ubiquiti), quasi séquentielles — signature d'un même lot de bornes neuves.

2 Confirmer l'identité des bornes

Vérifier le DNS interne (les baux DHCP des bornes sont enregistrés dynamiquement dans l'AD) et la bannière SSH :

```
nmap -sV --version-light -p 22 10.0.230.2,3,5
```

Attendu : noms `BTSSIOI109.bts.sio`, `BTSSIOI110.bts.sio`, `BTSSIOI111.bts.sio` (une borne par salle) et service `Dropbear sshd` — le démon SSH embarqué des AP UniFi. Des machines

Dell récentes (OUI `E8:CF:83`) présentes sur la même plage ne doivent pas être confondues avec les bornes.

Vérification

- ☐ **Trois bornes identifiées :** `10.0.230.2`, `10.0.230.3`, `10.0.230.5`
- ☐ **Résolution DNS inverse correcte (BTSSIOI109/110/111)**
- ☐ **Les bornes répondent au ping et présentent Dropbear sur le port 22**

5.2 Phase 2 — Création de la stack Docker sur docker-srv

3 Créer l'arborescence et le fichier de secrets

Sur l'hyperviseur :

```
ssh bts-proxmox
pct exec 200 -- mkdir -p /opt/docker/unifi
```

Créer `/opt/docker/unifi/.env` avec des mots de passe générés (`openssl rand -hex 20`), droits `600` :

```
MONGO_ROOT_USER=root
MONGO_ROOT_PASSWORD=<mot de passe genere>
MONGO_USER=unifi
MONGO_PASS=<mot de passe genere>
MONGO_DBNAME=unifi
MONGO_AUTHSOURCE=admin
```

4 Déposer le script d'initialisation MongoDB

L'application UniFi exige depuis la version 8.x une base MongoDB avec authentification RBAC [1]. Déposer `/opt/docker/unifi/init-mongo.sh` (contenu intégral en an-

nexe B) monté dans `/docker-entrypoint-initdb.d/` : il crée l'utilisateur applicatif avec les rôles `clusterMonitor` et `dbOwner` sur les bases `unifi`, `unifi_stat`, `unifi_audit` et `unifi_restore`.

5 Déposer le fichier docker-compose.yml

Déposer `/opt/docker/unifi/docker-compose.yml` (contenu intégral en annexe A). Points structurants :

- réseau `unifi-macvlan` (parent `eth0`, sous-réseau `10.0.0.0/16`) : le conteneur `unifi` y reçoit l'IP fixe `10.0.112.228` ;
- réseau `unifi-internal` (bridge interne) : seul lien entre `unifi` et `mongo`, jamais exposé ;
- image MongoDB figée (`mongo:7.0`) : MongoDB ne supporte pas les montées de version majeure automatiques [1] ;
- `restart: unless-stopped` sur les deux services, cohérent avec les autres stacks du serveur.

5.3 Phase 3 — Démarrage et validation de la stack

6 Démarrer la stack

```
pct exec 200 -- docker compose -f /opt/docker/unifi/docker-compose.yml
    up -d
```

Le premier tirage télécharge ≈ 2,3 Go d'images (compter 5 à 15 minutes selon le lien). Le premier démarrage de MongoDB exécute `init-mongo.sh` (création du compte root puis de l'utilisateur applicatif) ; le premier démarrage d'UniFi génère le certificat

auto-signé et initialise la base (2 à 4 minutes).

7 Contrôler l'état des services

```
pct exec 200 -- docker ps --filter name=unifi --format '{{.Names}}
    {{.Status}}'
pct exec 200 -- docker logs unifi 2>&1 | tail -5
```

Attendu : `unifi` et `unifi-mongo` *Up*, journal se terminant par `[ls.io-init] done.` sans trace d'exception Java.

Erreur typique au premier déploiement

Si le journal affiche `IllegalArgumentException: No username is provided in the connection string`, l'authentification MongoDB n'est pas en place (ancienne convention sans mot de passe). Reprendre les étapes 4 et 5 puis repartir d'un volume vierge : `docker compose down`, suppression de `mongo-data/` et `config/`, puis `up -d`. Le script d'initialisation ne s'exécute qu'au premier démarrage de la base.

Vérification

- ☐ `curl -kI https://10.0.112.228:8443/` **renvoie** `302` **(redirection vers** `/manage` **ou** `/setup`**)**
- ☐ `curl http://10.0.112.228:8080/inform` **renvoie** `400` **(endpoint *inform* actif : le code 400 est la réponse normale à un GET)**
- ☐ **Depuis un poste du LAN,** `https://10.0.112.228:8443` **affiche l'assistant UniFi**

5.4 Phase 4 — Exposition de l'interface via Traefik

8 Ajouter un transport HTTPS permissif à Traefik

L'UI UniFi n'existe qu'en HTTPS auto-signé : sans réglage, Traefik répond `502 Bad Gateway`. Ajouter un *serversTransport* dans la configuration dynamique : créer `/opt/docker/traefik/dynamic.yml` :

```
http:
  serversTransports:
    insecure:
      insecureSkipVerify: true
```

Déclarer le provider fichier dans `traefik.yml` (bloc `providers:`) et monter `dynamic.yml` dans le conteneur :

```
providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    exposedByDefault: false
    network: traefik-public
  file:
    filename: /etc/traefik/dynamic.yml
```

```
volumes:
  - ./traefik.yml:/etc/traefik/traefik.yml:ro
  - ./dynamic.yml:/etc/traefik/dynamic.yml:ro
```

Composant partagé

Traefik porte tous les services web de la plateforme. Sauvegarder `traefik.yml` et `docker-compose.yml` avant modification (`cp fichier fichier.bak-AAAAMMJJ`) et vérifier la non-régression des autres services après rechargement (étape 13). La coupure du proxy pendant le rechargement est

de quelques secondes.

9 Publier le contrôleur sous unifi.docker.bts.sio

Compléter le service `unifi` de la stack (annexe A) : rattachement au réseau externe `traefik-public` et labels Traefik calqués sur les autres services, avec deux spécificités — `server.scheme=https` (backend TLS) et le transport permissif :

```
labels:
  - "traefik.enable=true"
  - "traefik.docker.network=traefik-public"
  - "traefik.http.routers.unifi.rule=Host(`unifi.docker.bts.sio`)"
  - "traefik.http.routers.unifi.entrypoints=websecure"
  - "traefik.http.routers.unifi.tls=true"
  - "traefik.http.services.unifi.loadbalancer.server.scheme=https"
  - "traefik.http.services.unifi.loadbalancer.server.port=8443"
  - "traefik.http.services.unifi.loadbalancer.serverstransport=insecure@file"
```

Aucun enregistrement DNS à créer : le wildcard `*.docker.bts.sio` (Pi-hole et DNS AD) résout déjà le nom vers Traefik.

10 Recharger Traefik et recréer le conteneur UniFi

```
pct exec 200 -- docker compose -f /opt/docker/traefik/docker-compose.yml
    up -d
pct exec 200 -- docker compose -f /opt/docker/unifi/docker-compose.yml
    up -d
```

Vérification

- ☐ `curl -k https://unifi.docker.bts.sio/` renvoie 302 vers `/setup/` ou `/manage`
- ☐ **Non-régression** : `grafana`, `netbox`, `portainer`, `kuma.docker.bts.sio` ré-

pondent normalement

☐ **L'accès direct `https://10.0.112.228:8443` reste fonctionnel (chemin *inform* des bornes)**

5.5 Phase 5 — Initialisation du contrôleur via l'API

L'assistant graphique de première exécution (*setup wizard*) peut être déroulé au navigateur, mais la phase est ici industrialisée par appels API (reproductible, sans poste graphique). Les endpoints ci-dessous ont été relevés dans le frontend de l'assistant (version 10.6.101).

11 Créer le compte super-administrateur local

```
PW="$(openssl rand -hex 20)"
curl -k -X POST https://10.0.112.228:8443/api/cmd/sitemgr \
  -H 'Content-Type: application/json' \
  -d "{\"cmd\":\"add-default-admin\",\"name\":\"Administrateur\",\
\"email\":\"admin@bts.sio\",\"x_password\":\"$PW\"}"
```

Attendu : `{"meta":{"rc":"ok"}}`. Conserver `$PW` pour les étapes suivantes ; il sera versé en coffre en phase 6.

12 Ouvrir une session et fixer le pays et le fuseau horaire

```
curl -k -c /tmp/unifi.cookies -X POST \
  https://10.0.112.228:8443/api/login -H 'Content-Type:
    application/json' \
  -d "{\"username\":\"admin@bts.sio\",\
\"password\":\"$PW\",\"remember\":false}"
CSRF=$(awk '$6=="csrf_token"{print $7}' /tmp/unifi.cookies)
curl -k -b /tmp/unifi.cookies -X POST \
  https://10.0.112.228:8443/api/set/setting/country \
  -H "X-CSRF-Token: $CSRF" -H 'Content-Type: application/json' \
  -d '{"code":"250"}'
```

```
curl -k -b /tmp/unifi.cookies -X POST \
  https://10.0.112.228:8443/api/set/setting/locale \
  -H "X-CSRF-Token: $CSRF" -H 'Content-Type: application/json' \
  -d '{"timezone":"Europe/Paris"}'
```

Le code pays 250 (France, table `/api/stat/ccode`) fixe le domaine réglementaire radio des bornes ; le fuseau `Europe/Paris` aligne journaux et planifications.

13 Clôturer l'assistant (étape obligatoire)

```
curl -k -b /tmp/unifi.cookies -X POST \
  https://10.0.112.228:8443/api/cmd/system \
  -H "X-CSRF-Token: $CSRF" -H 'Content-Type: application/json' \
  -d '{"cmd":"set-installed"}'
```

Piège constaté en déploiement réel

Tant que `set-installed` n'a pas été appelé, l'URL `/manage/account/login` redirige vers l'assistant (`/setup/`), dont l'écran ń *Sign In to Your UI Account* ż attend un compte cloud Ubiquiti (ui.com) — les identifiants locaux y sont rejetés (ń *Invalid username or password* ż) alors qu'ils sont valides. Après clôture de l'assistant, la vraie page de connexion accepte le compte local. L'étape de liaison cloud est optionnelle et volontairement ignorée ici (contrôleur autonome, sans dépendance externe).

Vérification

- ☐ `GET /api/self` **(avec session) retourne le compte,** `is_super: true`**,** `is_owner: true`
- ☐ `curl -k https://unifi.docker.bts.sio/manage/account/login` **renvoie 200 (plus de redirection vers `/setup/`)**
- ☐ **Connexion réussie au navigateur sur** `https://unifi.docker.bts.sio` **(in-**

> **terface en français pour un navigateur français)**

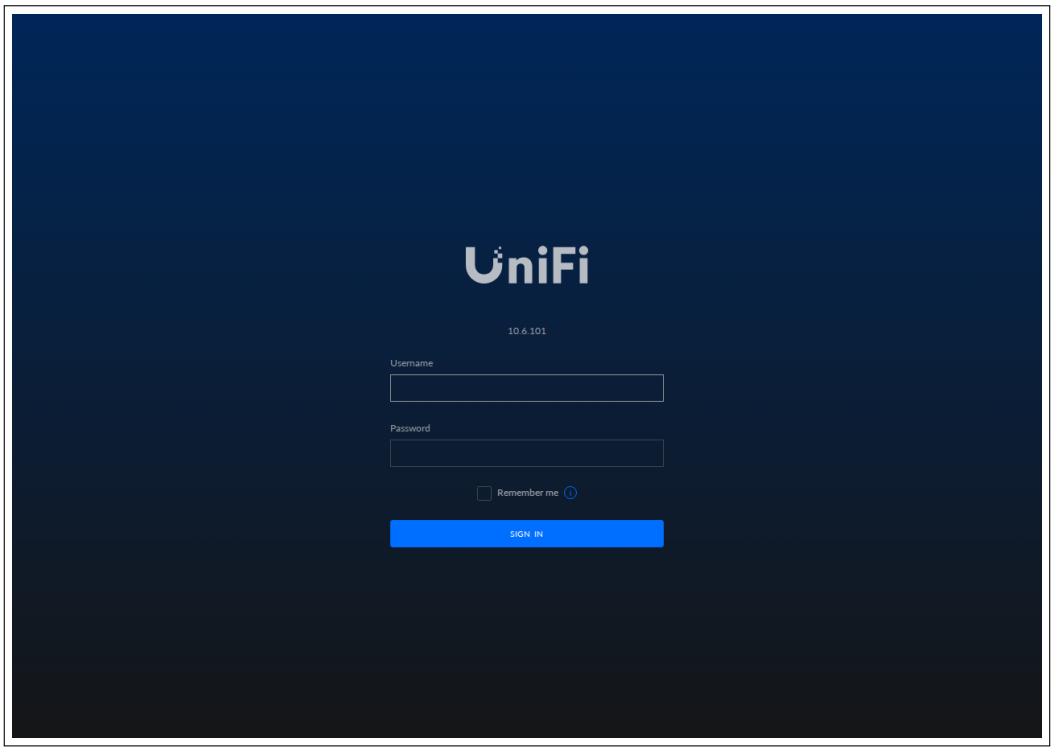

Figure 1 : Page de connexion du contrôleur après clôture de l'assistant (version 10.6.101) : le compte **local** créé en phase 5 y est accepté, contrairement à l'écran ń *UI Account* ż de l'assistant qui exige un compte cloud Ubiquiti.

5.6 Phase 6 — Mise en coffre du compte administrateur

14

Enregistrer l'identifiant dans Vaultwarden

Créer une entrée dans le coffre de l'équipe (MO-PLT-005), par exemple avec le client `bw` :

```
bw get template item | jq --arg pw "$PW" '
  .type=1
  | .name="UniFi Network Controller (docker-srv)"
  | .login.username="admin@bts.sio"
```

```
  | .login.password=$pw
  | .login.uris=[{"uri":"https://unifi.docker.bts.sio"}]
' | bw encode | bw create item
```

Renseigner en note : version de l'application, emplacement de la stack (`/opt/docker/unifi`, CT 200) et IP macvlan. Verrouiller le coffre après usage (`bw lock`) et supprimer tout fichier temporaire contenant le mot de passe.

5.7 Phase 7 — Adoption des bornes

15 Vérifier le paramètre Inform Host

Dans l'UI : Settings → System → Advanced, renseigner *Inform Host* = `10.0.112.228` et cocher *Override* [1]. Ce réglage garantit que le contrôleur annonce aux équipements une adresse joignable (et non une IP interne au conteneur).

16 Adopter depuis l'inventaire

Le contrôleur étant joignable en couche 2 par les bornes (même réseau `10.0.0.0/16`), celles-ci apparaissent en attente dans UniFi Devices. Cliquer Adopt sur chacune (l'adoption prend 1 à 3 minutes, la borne redémarre).

Bornes ń gérées par une autre console ż

Si une borne a déjà été adoptée par une autre console (console d'un collègue, application mobile, ancien contrôleur), l'adoption échoue silencieusement : la borne passe en *Connection Interrupted* (`two_phase_adopt`) car le contrôleur ne peut pas pousser l'URL *inform* sans les identifiants SSH posés par l'autre

console. Trois issues, par ordre de préférence : (1) supprimer (*Forget*) la borne depuis l'autre console, ce qui la réinitialise à distance ; (2) saisir les identifiants SSH actuels de la borne lorsque l'UI les demande ; (3) appuyer une dizaine de secondes sur le bouton *reset* de la borne (retour usine, identifiants `ubnt/ubnt`) puis refaire l'adoption — la borne récupère aussitôt le WLAN configuré. Constaté le 04/09/2026 sur les trois bornes : adoption en attente de réinitialisation.

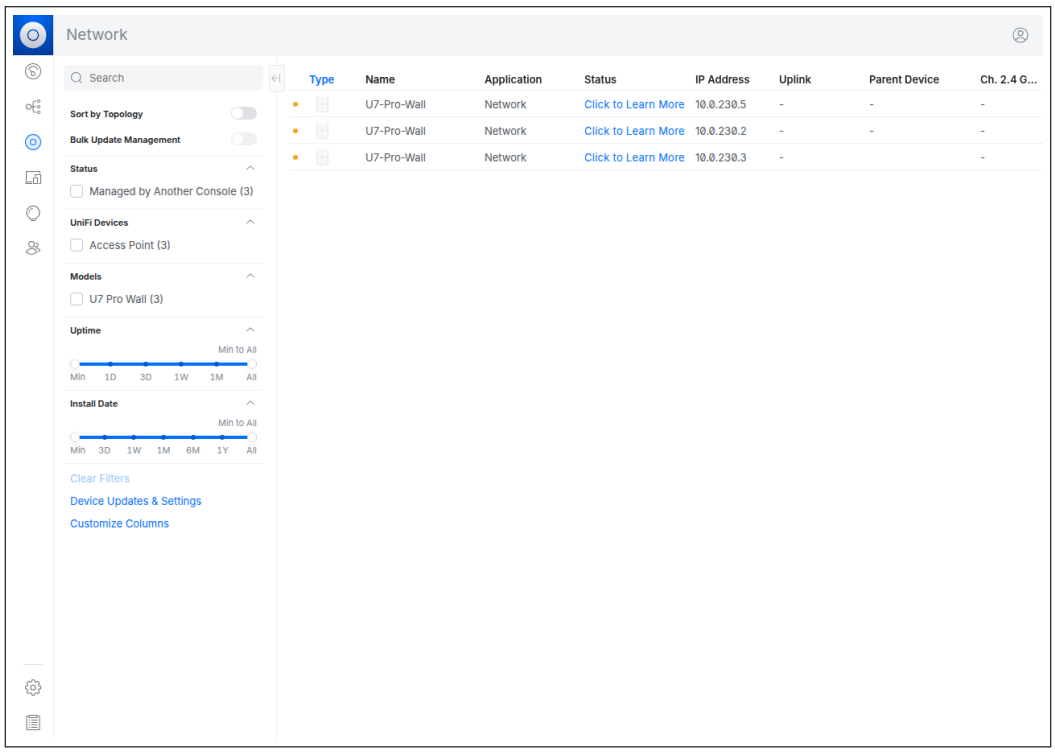

Figure 2 : Les trois U7 Pro Wall découvertes automatiquement par le contrôleur (pastilles orange, en attente d'adoption) : la diffusion de découverte de couche 2 aboutit grâce à l'IP macvlan, sans aucune intervention sur les bornes.

17 Forcer la prise de contact si une borne n'apparaît pas

En SSH sur la borne (identifiants d'usine `ubnt/ubnt` tant qu'elle n'est pas adoptée) :

```
ssh ubnt@10.0.230.2
```

```
set-inform http://10.0.112.228:8080/inform
```

La borne contacte le contrôleur et apparaît en attente d'adoption dans la minute. Après adoption, ses identifiants SSH sont remplacés par ceux définis dans Settings → System → *Device SSH Authentication*.

5.8 Phase 8 — Vérification finale

Vérification

- ☐ **UI accessible : `https://unifi.docker.bts.sio` (porte Authentik puis login local, voir « Protection de l'accès ») et `https://10.0.112.228:8443` (direct, restreint à l'alias `UniFi_Console_Admin_Sources` depuis le 20/09/2026)**
- ☐ **Les 3 bornes figurent dans UniFi Devices, état *Online*, bonnes salles**
- ☐ **Compte administrateur présent dans Vaultwarden, mot de passe testé depuis le coffre**
- ☐ **Redémarrage simulé : `docker restart unifi unifi-mongo`, services *Up* et UI de nouveau accessible**
- ☐ **IP `10.0.112.228` référencée dans Netbox (MO-PLT-017) et dans `ACCES_INFRASTRUCTURE.md`**

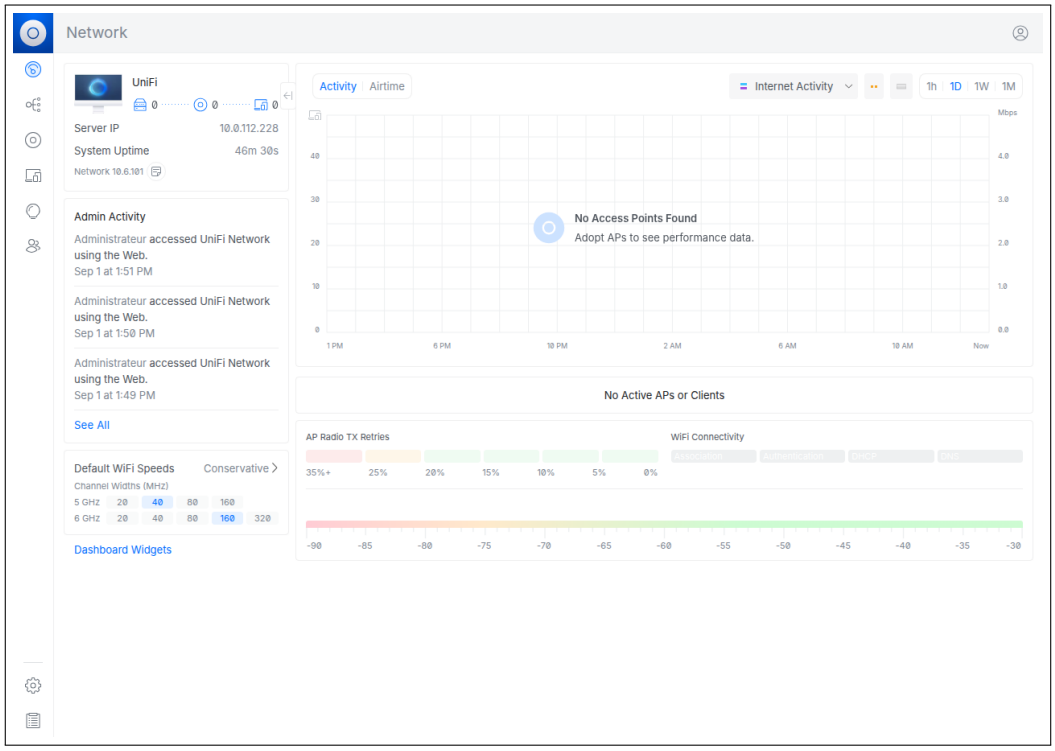

Figure 3 : Tableau de bord du contrôleur en service : le cartouche serveur confirme l'adresse `10.0.112.228` et la version 10.6.101. Les widgets de performance restent vides tant que les bornes ne sont pas adoptées (phase 7).

6 Exploitation courante

6.1 Configuration Wi-Fi en production (état au 10/09/2026)

Le contrôleur sert la production du Wi-Fi pédagogique. Les objets actifs sont :

Objet	État en production
WLAN `BTS SIO Etudiants`	WPA2/WPA3-Enterprise (802.1X, transition), mappé sur le réseau `wifi-etudiants` (*VLAN only* 240)
WLAN `BTS SIO Profs`	WPA2/WPA3-Enterprise, mappé sur `wifi-profs` (*VLAN only* 241)
Réseaux `wifi-etudiants` / `wifi-profs`	*VLAN only*, tags 240 / 241 (le réseau *Default* reste disponible pour retour au plat)
Profil RADIUS `NPS-DC1`	authentification `10.0.112.2:1812` (NPS sur DC1), accounting `10.0.112.20:1813` (collecteur RSSO), interim 300 s
Bornes	3 U7 Pro Wall adoptées et en ligne (`BTSSIOI109/I110/I111`), firmware 8.7.11, les deux SSID émis sur *All APs*

L'authentification des utilisateurs (802.1X PEAP-MSCHAPv2 contre NPS/AD), la segmentation VLAN, l'adressage DHCP, le portail captif et l'ouverture de session automatique (SSO) sont documentés dans le **MO-NET-008** (architecture complète, exploitation, dépannage). Le présent MO reste la référence pour la plateforme du contrôleur (stack, adoption, sauvegardes).

Accès API

L'ouverture de session API se fait par `POST /api/login` (voir annexe C). Sur cette version auto-hébergée, `POST /api/auth/login` renvoie `401 api.err.LoginRequired` même avec des identifiants valides — piège des exemples de documentation UniFi OS.

6.2 Sauvegardes

Deux niveaux complémentaires :

— **Sauvegarde applicative** : l'UI UniFi génère des sauvegardes automatiques (**Settings** → **System** → *Backup*) stockées dans `/opt/docker/unifi/config/data/backup/`. À inclure dans une copie externe (NAS) si exigé par la politique de sauvegarde.

— **Sauvegarde infrastructure** : le CT 200 est couvert par les jobs `vzdump` de l'hyperviseur (MO-PLT-016) ; la stack complète (`/opt/docker/unifi`) est restaurable avec le conteneur.

6.3 Mises à jour

```
pct exec 200 -- docker compose -f /opt/docker/unifi/docker-compose.yml pull
pct exec 200 -- docker compose -f /opt/docker/unifi/docker-compose.yml up -d
```

L'image applicative suit l'étiquette `latest` (pratique cohérente avec les autres stacks du serveur) ; l'image MongoDB reste **figée** en `7.0` et ne doit pas être montée de version majeure sans procédure dédiée [1]. Avant toute mise à jour applicative, déclencher une sauvegarde depuis l'UI.

6.4 Restauration sur une stack neuve

Déployer la stack (phases 2 et 3), ouvrir l'UI et, dans l'assistant de première exécution, choisir *Restore from backup* en fournissant le fichier `.unf`. Les bornes se reconnectent d'elles-mêmes dès que l'adresse *inform* (`10.0.112.228:8080`) redevient joignable — l'IP macvlan doit donc être conservée.

7 Dépannage

Symptôme	Diagnostic et résolution
`unifi` redémarre en boucle, *ń No username is provided in the connection string ż*	Authentification MongoDB absente. Reprendre le script `init-mongo.sh` et le fichier `.env`, repartir d'un volume vierge (voir note de la phase 3).
`502 Bad Gateway` sur `unifi.docker.bts.sio`	Le backend est HTTPS auto-signé : vérifier le label `server.scheme=https` et la présence du transport `insecure@file` (`dynamic.yml` monté, provider `file` déclaré dans `traefik.yml`).
Login refusé sur l'écran ń *UI Account* ż	Cet écran attend un compte cloud Ubiquiti, pas le compte local. Si l'URL de connexion redirige vers `/setup/`, l'assistant n'est pas clôturé : appeler `set-installed` (phase 5).
Une borne n'apparaît pas dans l'inventaire	Vérifier *Inform Host* = `10.0.112.228` (case *Override*), puis `set-inform http://10.0.112.228:8080/inform` en SSH sur la borne.
Borne adoptée mais figée *Connection Interrupted*	La borne était gérée par une autre console : les identifiants SSH actuels sont inconnus et l'URL *inform* n'a pas pu être poussée (aucun paquet *inform* reçu). Voir l'avertissement de la phase 7 — réinitialisation de la borne requise.
Le CT n'arrive pas à tester `10.0.112.228`	Limite macvlan : l'hôte ne joint pas l'IP macvlan de ses conteneurs. Tester depuis un autre poste du LAN.
Interface lente, conteneur limité mémoire	La variable `MEM_LIMIT` (Mo) borne le tas Java (2048 par défaut). Surveiller la mémoire du CT 200 (8 Go partagés avec les autres stacks).

8 Retour arrière

1. Supprimer la stack : `docker compose -f /opt/docker/unifi/docker-compose.yml down` puis `rm -rf /opt/docker/unifi`.
2. Retirer le provider `file` de `traefik.yml` et le montage `dynamic.yml`, ou restaurer les sauvegardes `*.bak-AAAAMMJJ` de la phase 4, puis recharger Traefik.
3. Libérer l'IP `10.0.112.228` dans Netbox et retirer l'entrée du coffre si le contrôleur n'est pas redéployé.
4. Les bornes conservent leur dernière configuration et continuent de diffuser ; elles devront être réinitialisées (*reset* physique) pour être rattachées à un autre contrôleur.

A Annexe A — docker-compose.yml (intégral)

```
---
networks:
  unifi-macvlan:
    driver: macvlan
    driver_opts:
      parent: eth0
    ipam:
      config:
        - subnet: 10.0.0.0/16
          gateway: 10.0.112.1
  unifi-internal:
    driver: bridge
    internal: true
  traefik-public:
    external: true

services:
  mongo:
    image: mongo:7.0
    container_name: unifi-mongo
    restart: unless-stopped
    environment:
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_ROOT_USER}
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_ROOT_PASSWORD}
      MONGO_USER: ${MONGO_USER}
      MONGO_PASS: ${MONGO_PASS}
      MONGO_DBNAME: ${MONGO_DBNAME}
      MONGO_AUTHSOURCE: ${MONGO_AUTHSOURCE}
    volumes:
      - ./mongo-data:/data/db
      - ./init-mongo.sh:/docker-entrypoint-initdb.d/init-mongo.sh:ro
    networks:
      - unifi-internal
```

```
unifi:
  image: lscr.io/linuxserver/unifi-network-application:latest
  container_name: unifi
  restart: unless-stopped
  depends_on:
    - mongo
  environment:
    PUID: "1000"
    PGID: "1000"
    TZ: Europe/Paris
    MONGO_USER: ${MONGO_USER}
    MONGO_PASS: ${MONGO_PASS}
    MONGO_HOST: mongo
    MONGO_PORT: "27017"
    MONGO_DBNAME: ${MONGO_DBNAME}
    MONGO_AUTHSOURCE: ${MONGO_AUTHSOURCE}
    MEM_LIMIT: "2048"
  volumes:
    - ./config:/config
  labels:
    - "traefik.enable=true"
    - "traefik.docker.network=traefik-public"
    - "traefik.http.routers.unifi.rule=Host(`unifi.docker.bts.sio`)"
    - "traefik.http.routers.unifi.entrypoints=websecure"
    - "traefik.http.routers.unifi.tls=true"
    - "traefik.http.services.unifi.loadbalancer.server.scheme=https"
    - "traefik.http.services.unifi.loadbalancer.server.port=8443"
    -
 "traefik.http.services.unifi.loadbalancer.serverstransport=insecure@file"
  networks:
    unifi-internal: {}
    unifi-macvlan:
      ipv4_address: 10.0.112.228
    traefik-public: {}
```

B Annexe B — init-mongo.sh (intégral)

Script fourni par la documentation de l'image [1], exécuté par le conteneur MongoDB à son premier démarrage uniquement :

```
#!/bin/bash

if which mongosh > /dev/null 2>&1; then
  mongo_init_bin='mongosh'
else
  mongo_init_bin='mongo'
fi
"${mongo_init_bin}" <<EOF
use ${MONGO_AUTHSOURCE}
db.auth("${MONGO_INITDB_ROOT_USERNAME}", "${MONGO_INITDB_ROOT_PASSWORD}")
db.createUser({
  user: "${MONGO_USER}",
  pwd: "${MONGO_PASS}",
  roles: [
    "clusterMonitor",
    { db: "${MONGO_DBNAME}", role: "dbOwner" },
    { db: "${MONGO_DBNAME}_stat", role: "dbOwner" },
    { db: "${MONGO_DBNAME}_audit", role: "dbOwner" },
    { db: "${MONGO_DBNAME}_restore", role: "dbOwner" }
  ]
})
EOF
```

C Annexe C — Endpoints API utilisés

Relevés dans le frontend de l'assistant (version 10.6.101), utiles à l'automatisation :

Endpoint	Rôle
`GET /status`	État du serveur et version (non authentifié)
`POST /api/cmd/sitemgr` (`add-default-admin`)	Création du super-administrateur local
`POST /api/login`	Ouverture de session (cookies `unifises`, `csrf_token`)
`POST /api/set/setting/country`	Pays réglementaire (code `250` = France)
`POST /api/set/setting/locale`	Fuseau horaire (`Europe/Paris`)
`POST /api/cmd/system` (`set-installed`)	Clôture de l'assistant (débloque `/manage`)
`GET /api/self`	Compte courant (contrôle `is_super`)

Les requêtes d'écriture exigent l'en-tête `X-CSRF-Token` reprenant le cookie `csrf_token`.

D Références

[1] LinuxServer.io. *docker-unifi-network-application*. Image Docker communautaire de l'application UniFi Network (branche autohébergée classique) : variables d'environnement, initialisation MongoDB avec authentification RBAC, procédure d'adoption et paramètre Inform Host. URL : https://github.com/linuxserver/docker-unifi-network-application (visité le 01/09/2026).

[2] Ubiquiti Inc. *Self-Hosting UniFi*. Centre d'aide Ubiquiti. UniFi OS Server comme nouveau standard d'auto-hébergement ; impossibilité officielle d'exécution en conteneur (ń This is a complete solution that requires certain services to be run on the host ż). URL : https://help.ui.com/hc/en-us/articles/34210126298775-Self-Hosting-UniFi (visité le 01/09/2026).

[3] Ubiquiti Inc. *UniFi Device Adoption*. Centre d'aide Ubiquiti. Méthodes d'adoption des équipements UniFi : découverte de couche 2, URL inform et identifiants d'usine. URL : https://help.ui.com/hc/en-us/articles/204909754-UniFi-Device-Adoption (visité le 01/09/2026).

[4] Ubiquiti Inc. *UniFi OS Server — Releases*. Page de téléchargement officielle d'UniFi OS Server (installateur natif Linux x64 et Windows), alternative non conteneurisée au présent déploiement. URL : https://www.ui.com/download/releases/unifi-os-server (visité le 01/09/2026).

[5] Ubiquiti Inc. *UniFi U7 Pro Wall — Tech Specs*. Spécifications techniques de la borne : WiFi 7 (802.11be), 6 flux spatiaux, alimentation PoE+, liaison montante 2,5 GbE, montage mural. URL : https://techspecs.ui.com/unifi/wifi/u7-pro-wall (visité le 01/09/2026).