Guide API EPORDO Connect (Français)

Élément	Contenu
Version du document	**1.1.5**
Date de publication	2026-11-11
Lecteurs	Développeurs intégrateurs, revendeurs / intégrateurs, équipes de livraison projet
Produit	**EPORDO Connect** (plateforme cloud d'accès pour équipements tiers)
Forme de livraison	Ce manuscript Markdown + **PDF** téléchargeable depuis le portail (Aide → Télécharger le guide API)

> Ce document décrit l'**API REST côté locataire** (préfixe de chemin `/connect/v1`). Les communications entre l'équipement et le cloud sont gérées par Connect ; l'intégrateur **n'a pas besoin** d'analyser ni d'implémenter le protocole binaire / texte brut de l'équipement.

> Les exemples utilisent par défaut une Base URL de style production ; pour les essais, remplacez-la par l'adresse fournie pour votre environnement.

> **Convention :** les chemins ci-dessous sont relatifs à la Base URL. Les noms de champs JSON font foi dans ce document (par ex. le corps métier d'un événement est `payload_json`, à parser avec `JSON.parse`).

Historique des révisions

Version	Date	Description
1.1.5	2026-11-11	§5.3.1 : `door_mode` et `door_opened` dans les cinq langues ; `photo_status` = `absent`, `stored`, `omitted_too_large` ou `omitted_invalid`
1.1.4	2026-11-10	§5.3 : types courants `door.remote.recorded` et `device.presence` ; fenêtre en ligne de 10 minutes, scan hors ligne ~5 minutes, anciens noms limités au front correspondant
1.1.3	2026-11-08	Peaufinage : en-tête §5.3 = `event_type` et ajout de `door.open.recorded` ; annexe D : `GET /commands/{id}` + `POST .../cancel`
1.1.2	2026-11-08	Corrections : le champ d'événement est `event_type` (pas `type`) ; l'API photo renvoie du JSON/`data_base64` (pas un flux image direct) ; pas de `GET /webhooks/{id}` ; pas de `DELETE /device-groups/{id}` (utiliser `status=DISABLED`) ; conservation alignée sur 3–14 jours
1.1.1	2026-11-08	Vérification contre la production : ajout API Key / chemins de groupes et exemples ; vérification de signature Webhook et outbox ; exemples `payload_json` d'événements et note de conservation ; exemple d'envoi de personnel en fan-out ; table des matières
1.1.0	2026-11-08	Ajout de la vue d'ensemble système et des relations de communication ; exemples cURL et scénarios par chapitre ; ajout du détachement d'équipement ; historique des révisions ; livraison PDF
1.0.0	2026-11-07	Première version publique : revendication, événements, Webhook/MQTT, commandes standard, fan-out, résumé des codes d'erreur

Les corrections de pure rédaction n'ajoutent qu'une version patch à l'historique ; un changement de comportement d'API augmente la version mineure et est noté dans le chapitre concerné.

Table des matières

1. Vue d'ensemble du système
2. Environnement et authentification (gestion des API Key incluse)
3. Format d'erreur unifié
4. Équipements
5. Événements
6. Webhook / MQTT
7. Commandes standard
8. Fan-out par liste de numéros de série
9. Groupes d'équipements et Scoped Key
10. Limitation de débit et quotas journaliers
11. Flux d'intégration recommandé

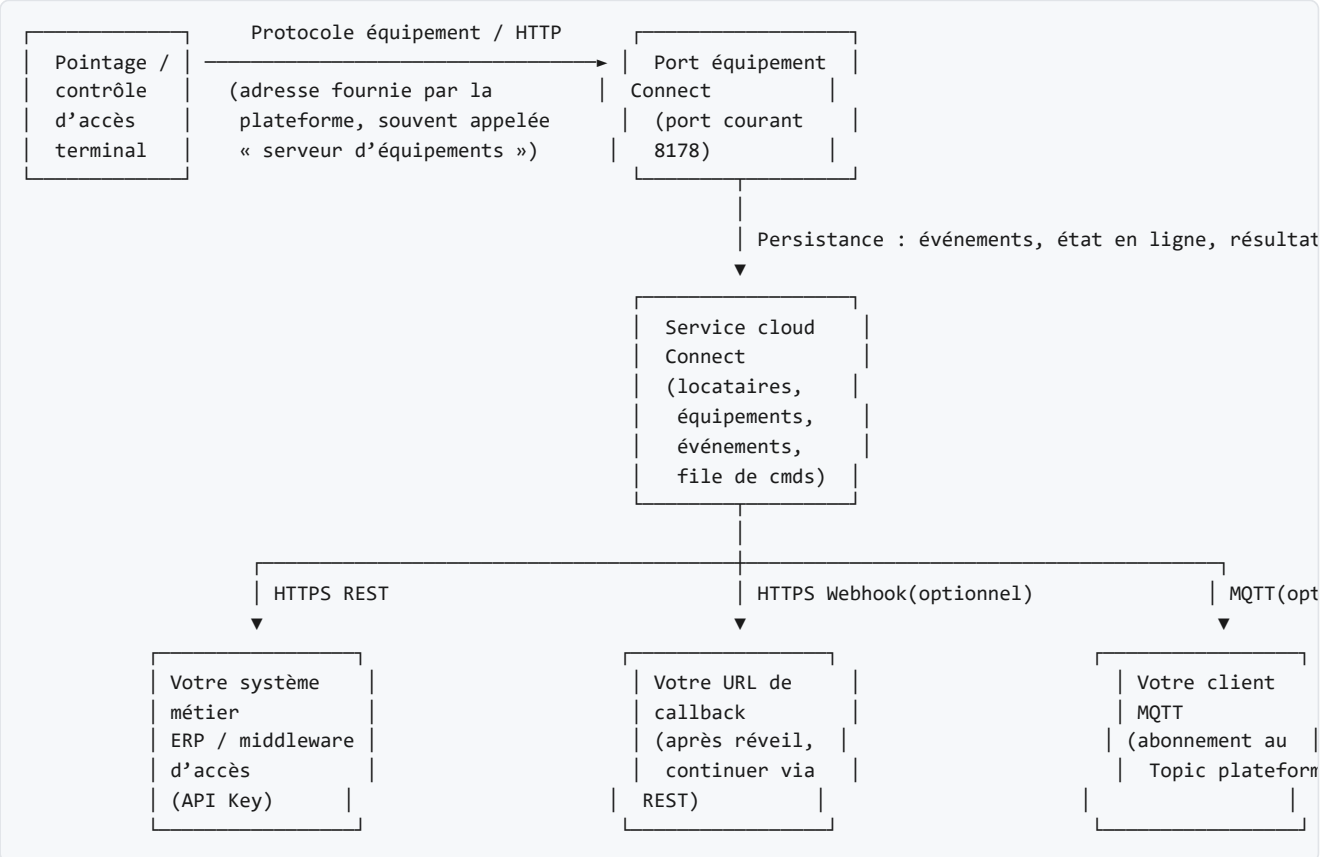
- Annexe A Codes d'erreur · Annexe B kind des credentials · Annexe C Support · Annexe D Référence rapide

---

1. Vue d'ensemble du système (à lire en premier)

1.1 Place de Connect dans l'architecture globale

On peut le voir comme une « couche d'adaptation cloud entre les équipements et votre système métier » :



Ne confondez pas les deux lignes :

Ligne	Qui se connecte à qui	Ce que vous devez faire
Ligne équipement	Terminal → serveur d'équipements Connect (exemple : `http://api.bnetlink.com:8178/`)	Renseigner l'adresse fournie par la plateforme dans le menu de l'équipement ; **ne pas** assembler les trames d'équipement dans votre code
Ligne métier	Votre serveur → REST Connect (exemple : `https://api.bnetlink.com/connect/v1`)	Appeler les interfaces de ce document avec un **Bearer API Key**

Le portail (navigateur) sert aux opérateurs pour revendiquer, consulter les événements et tester des commandes ; **pour l'intégration formelle, utilisez votre propre backend + API Key**, et n'écrivez jamais la Key dans une page web publique.

1.2 Protocole de communication équipement (point de vue intégrateur)

Les terminaux communiquent avec le **serveur d'équipements** Connect via le protocole constructeur (ex. machines dynamiques BS, machines ordinaires FK, etc.). Connect assure :

1. La reconnaissance du numéro de série et l'appartenance au locataire ;
2. La normalisation des remontées (pointage / enrollment, etc.) en **JSON d'événement** unifié ;
3. La traduction des **commandes standard** que vous envoyez via REST en trames exécutables par l'équipement, la mise en file et le retour de résultat.

C'est pourquoi les documents d'intégration utilisent des noms standard, par ex. `set-user`, `delete-user`, et **non** les chaînes de codes de commande internes à l'équipement. Vous n'avez besoin que de :

- Revendiquer le numéro de série `sn` ;
- Récupérer les événements / recevoir Webhook ou MQTT ;
- Appeler les commandes standard et interroger l'état via `command_id`.

1.3 Cycle de vie typique (exemple)

Exemple : un magasin installe une nouvelle borne faciale reliée à votre middleware de pointage.

1. À la sortie d'usine, le constructeur a déjà inscrit ce numéro de série dans le « stock vendable », lié à votre locataire.
2. Vous (ou le portail) appelez `POST /devices`, `sn=ABC123` → **revendication réussie**.
3. Dans la configuration de l'équipement, renseignez l'adresse du serveur d'équipements et mettez en réseau → démarrage des heartbeats / remontées.
4. Un employé passe le visage → Connect produit un événement `punch.recorded` → vous le tirez avec `GET /events`, ou êtes d'abord réveillé par Webhook/MQTT puis tirez.
5. Depuis l'ERP vous envoyez l'employé « 1001 » : `POST /devices/ABC123/commands/set-user`.
6. Interrogez `GET /commands/{command_id}` jusqu'à `SUCCEEDED` ou `FAILED`.
7. Si le magasin retire la machine : `DELETE /devices/ABC123` **détache**, le numéro de série revient à non ouvert et pourra être revendiqué plus tard ; les événements de pointage historiques sont conservés.

1.4 Ce que Connect fait / ne fait pas

Fait :

- Revendiquer / désactiver / activer / détacher des équipements ;
- Recevoir les événements de pointage, changements d'enrollment, etc. ;
- Envoyer des commandes standard (mise à l'heure, ouverture de porte, personnel, plages d'accès, etc.);
- Webhook optionnel, MQTT optionnel.

Ne fait pas :

- Planning, paie, moteur de règles de présence (c'est le rôle du système métier ou d'autres produits);
- Conserver pour vous le curseur « jusqu'à quel événement j'ai lu » (stockez-le dans votre propre base);

- Conversion automatique de formats de modèles visage / empreinte entre firmwares.

1.5 Les trois blocs du quotidien de l'intégrateur

Capacité	Description
Équipements	Revendiquer `sn`, lister, désactiver / activer, détacher
Événements	Tirage ; Webhook et/ou MQTT optionnels comme réveil ou canal parallèle
Commandes	Mise à l'heure, statut, ouverture, personnel, plages ; batch possible par liste de SN (fan-out)

---

2. Environnement et authentification

2.1 Base URL

Environnement	Préfixe REST (exemple)
Production	Fourni par contrat / livraison, de la forme `https://api.bnetlink.com/connect/v1`
Essais locaux (si activés)	`http://127.0.0.1:8080/connect/v1`

Tous les chemins ci-dessous sont relatifs à ce préfixe. Ex. `GET /me` = `GET {base}/me`.

Exemple de serveur d'équipements (à renseigner sur le terminal): `http://api.bnetlink.com:8178/` (selon livraison).

2.2 Authentification

Sauf pour `GET /health`, l'en-tête de requête doit contenir :

```
Authorization: Bearer {API_KEY}
Content-Type: application/json
Accept: application/json
```

Notes :

- Une API Key correspond à un **locataire** et n'accède qu'aux données de ce locataire.
- La Key est créée dans la page « Clés » du portail, ou délivrée à l'ouverture de compte ; le **texte en clair n'apparaît qu'une fois**, conservez-le soigneusement.
- La **clé de signature Webhook** et le **mot de passe d'abonnement MQTT** ne sont **pas** la même chaîne que l'API Key.
- Le compte de connexion au portail (identifiant + mot de passe) sert uniquement aux opérations navigateur et **ne peut pas** remplacer l'API Key pour le REST.

2.3 Contrôle de santé (exemple)

```
curl -s "https://api.bnetlink.com/connect/v1/health"
```

Succès :

```
{ "status": "ok" }
```

2.4 Locataire courant (exemple)

```
curl -s "https://api.bnetlink.com/connect/v1/me" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

En cas de succès, la réponse contient `tenant_id`, l'offre, la portée de la Key, etc.(champs selon l'instance).

Exemple : si `scope_type` vaut une portée de groupes (`GROUPS`), certaines interfaces de niveau compte (revendication, détachement, gestion Webhook / clés / groupes) renvoient `403` `forbidden_scope`.

2.5 Gestion des API Key (REST)

La page « Clés » du portail s'appuie sur les interfaces ci-dessous. À la création, le texte en clair **n'est renvoyé qu'une fois**.

```
# Liste(sans texte en clair)
curl -s "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"

# Créer une Key de compte entier
curl -s -X POST "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"name\":\"erp-prod\",\"scope_type\":\"ACCOUNT\"}"

# Créer une Key limitée aux groupes(group_ids doit être des groupes ACTIVE de ce locataire)
curl -s -X POST "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"name\":\"store-a\",\"scope_type\":\"GROUPS\",\"group_ids\":[1,2]}"

# Désactiver / réactiver(status: REVOKED | ACTIVE)
curl -s -X PUT "https://api.bnetlink.com/connect/v1/credentials/2" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"status\":\"REVOKED\"}"
```

Exemple de succès à la création (enregistrez immédiatement `api_key`):

```
{
  "api_key": "epc_live_…",
  "key_prefix": "epc_live_…",
  "scope_type": "ACCOUNT",
  "warning": "Store api_key now; it will not be shown again."
}
```

Note : on ne peut pas désactiver la **dernière** Key `ACTIVE` du locataire (`409` `last_active_api_key`). Une Key de groupe **ne peut pas** se connecter au portail.

---

3. Format d'erreur unifié

```
{
  "error": {
    "code": "device_not_found",
    "message_key": "connect.error.device_not_found",
    "message": "device_not_found"
  }
}
```

HTTP	Signification (résumé)
400	Paramètres invalides
401	Key absente / Key invalide ou désactivée
403	Droits insuffisants (ex. Key de groupe sur une interface de niveau compte)
404	Ressource inexistante ou hors de ce locataire
409	Conflit (revendication en double, équipement désactivé, conflit d'idempotence, etc.)
429	Limitation par minute ou quota journalier de commandes atteint

Exemple : ouverture de porte sur un équipement déjà désactivé → HTTP `409`, `code` = `device_disabled`.

Les `code` courants figurent en Annexe A.

---

4. Équipements

4.1 Revendiquer un équipement

```
POST /devices
```

```
{ "sn": "TEST-SN-001", "alias": "Portique accueil" }
```

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sn\":\"TEST-SN-001\",\"alias\":\"Portique accueil\"}"
```

Exemple de succès :

```
{
  "sn": "TEST-SN-001",
  "connect_device_id": 12,
  "register_status": "ACTIVE",
  "protocol_track": "BS",
  "model_code": "P5",
  "firmware_version": "1.02"
}
```

Notes :

- On ne peut revendiquer que des numéros de série **présents dans le stock constructeur, déjà liés à ce locataire, et pas encore ouverts**.
- Protocole / modèle / firmware viennent du préremplissage constructeur ; le corps de requête **ne peut pas** les falsifier. Le modèle vendu est le nom court `P5` ou `P7`. Le `dev_model` du Header ne le change pas.
- Numéro inconnu → `404` `device_not_in_stock`.
- Déjà attribué à un autre locataire → `403` `device_not_for_tenant`.
- Déjà revendiqué par ce locataire → `409` `device_already_registered`.
- Le champ externe s'appelle toujours `sn`.

Exemple : l'étiquette de la boîte indique `B515…`, mais le constructeur n'a pas encore inscrit ce numéro dans votre stock → la revendication renvoie `device_not_in_stock` ; contactez le constructeur pour le préremplissage puis réessayez.

4.2 Liste / détail des équipements

```
curl -s "https://api.bnetlink.com/connect/v1/devices?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

La liste prend en charge un curseur : `next_cursor` est opaque ; pour la page suivante, passez-le tel quel dans `cursor=`.

4.3 Désactiver / activer

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/disable" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/enable" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

- **Désactiver** : reste rattaché à ce compte, mais plus aucune commande ne peut être envoyée (`409` `device_disabled`). Les événements remontés par l'équipement peuvent en général encore être enregistrés (selon le registry).
- **Activer** : rétablit l'envoi de commandes.
- Désactiver ≠ détacher.

4.4 Détacher un équipement

```
curl -s -X DELETE "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Exemple de succès :

```
{ "sn": "TEST-SN-001", "claim_status": "UNCLAIMED" }
```

Effets :

- L'équipement est retiré de ce compte ; le stock constructeur revient à « non ouvert », **revendication à nouveau possible** ;
- L'enrollment personnel côté plateforme pour cette machine est effacé ; les commandes en file sont annulées ;
- **Les événements de pointage historiques sont conservés** ;
- Une sous-clé de portée groupe **ne peut pas** appeler cette interface (`403`).

4.5 Personnel et credentials sur l'équipement (enrollment courant côté plateforme)

Méthode	Chemin	Description	
GET	`/devices/{sn}/users`	Liste du personnel ; optionnel `scope=device\`	compatible `,` q`
GET	`/devices/{sn}/users/{userId}`	Détail personnel et métadonnées des credentials	
GET	`/devices/{sn}/users/{userId}/credentials/{kind}/{slot}`	Corps du credential (`data_base64`)	

```
curl -s "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/users?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

`kind` autorisés uniquement : `photo`, `face`, `fingerprint`, `palm`, `card`, `password` (voir Annexe B).

---

5. Événements

Connect **ne conserve pas** votre curseur de lecture. Enregistrez dans votre propre base « jusqu'à quel `event_id` / heure j'ai traité ».

5.1 Récupérer les événements

```
curl -s "https://api.bnetlink.com/connect/v1/events?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Tirer le dernier jour selon l'heure d'ingestion (exemple ; remplacez par de vrais ISO-8601):

```
curl -s "https://api.bnetlink.com/connect/v1/events?received_from=2026-11-07T00:00:00Z&received_to=2026-11-08T00:
  -H "Authorization: Bearer YOUR_API_KEY"
```

Résumé des règles :

- Les paramètres temporels sont en ISO-8601 ; lorsque début et fin forment une paire, l'étendue ne dépasse pas **7 jours**.
- `from` / `to` = heure métier de survenue ; `received_from` / `received_to` = heure d'ingestion plateforme.
- Chaque élément de liste contient une **chaîne** `payload_json` (forme textuelle du JSON métier), **sans** la grande image de capture.
- La plateforme conserve les événements chauds selon la config locataire : environ **7 jours** par défaut, plage configurable d'environ **3–14 jours** (selon contrat / config plateforme); l'horloge suit `received_at`. Après expiration, `GET /events/{id}` peut devenir introuvable. Persistez à temps.
- La limite « chaque paire de fenêtre temporelle ≤ 7 jours » dans les requêtes de liste est une limite de **fenêtre de requête**, distincte de la durée de conservation sur disque.

Forme d'un élément de liste (indicatif, noms de champs alignés sur la production):

```
{
  "event_id": "evt_a1b2c3d4e5f6478990abcdef12345678",
  "event_type": "punch.recorded",
  "occurred_at": "2026-10-29T14:30:00",
  "received_at": "2026-10-29T14:30:01.123+00:00",
  "connect_device_id": 12,
  "sn": "TEST-SN-001",
  "payload_json": "{\"event_id\":\"evt_a1b2…\",\"sn\":\"TEST-SN-001\",\"user_id\":\"1001\",\"occurred_at\":176172
}
```

Effectuez un `JSON.parse` sur `payload_json`. Notez que le champ externe est **`event_type`**, pas `type`. Le corps HTTP poussé par Webhook / MQTT est le **JSON métier obtenu après parse de `payload_json`** (pas l'enveloppe de ligne de liste).

Exemple : un Webhook a échoué la nuit et 20 événements ont manqué. Au matin, tirez en pagination avec `received_from` couvrant la nuit, dédupliquez par `event_id` dans la base métier pour compléter.

5.2 Événement unitaire et capture photo

```
curl -s "https://api.bnetlink.com/connect/v1/events/{eventId}" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s "https://api.bnetlink.com/connect/v1/events/{eventId}/photo" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

L'interface photo n'est disponible que si le pointage a déjà une capture en stock (souvent `photo_status=stored`). La réponse est du **JSON**, pas un flux d'octets image direct, par exemple :

```
{
  "event_id": "evt_…",
  "content_type": "image/jpeg",
  "byte_length": 12345,
  "sha256": "…",
  "data_base64": "/9j/4AAQ…"
}
```

Décodez `data_base64` puis enregistrez sur disque. Pas de photo → `404` `photo_not_found` ; événement inexistant → `404` `event_not_found`. Liste / détail ne portent pas les octets d'image.

5.3 Types d'événements courants

event_type	Signification
`punch.recorded`	Pointage
`enroll.data.received`	Résumé d'enrollment / changement de credentials
`command.completed` / `command.failed`	État final de commande
`door.state.changed`	Changement d'état du contact de porte (remontée proactive de l'équipement)
`door.remote.recorded`	Action de porte à distance (`mode` : pulse/hold/close ; pas un pointage)
`device.presence`	Front en ligne / hors ligne (`online` : 1/0)

Les nouvelles lignes n'utilisent que les types ci-dessus. `online` de `GET /devices` vaut `1` si `last_seen_at` date de moins de **10 minutes**, sinon `0`. `device.presence` ne s'écrit qu'au front : une remontée écrit `online=1` ; un scan d'environ **5 minutes** écrit `online=0`, donc l'événement hors ligne arrive environ **10–15 minutes** après le dernier paquet. `occurred_at` est la dernière remontée ; `received_at` est l'instant où Connect a enregistré le front. Les anciens noms `device.online` / `device.offline` et `door.open.recorded` / `door.close.recorded` filtrent et s'abonnent encore, mais seulement au front correspondant.

5.3.1 Champs de `payload_json` pour le pointage (après parse)

Champ	Type	Description
`event_id`	string	Id d'événement, clé primaire de déduplication
`sn`	string	Numéro de série de l'équipement
`user_id`	string	Identifiant personnel
`occurred_at`	number	Heure métier de survenue, horodatage en **millisecondes**
`device_time`	string	Chaîne d'heure locale de l'équipement (souvent `yyyyMMddHHmmss`)
`verify_mode`	string	Description du mode de vérification (ex. Face / Card)

`door_mode`	string	Facultatif. Texte de porte de l'équipement (ex. `hand_open` / `open`)
`door_opened`	boolean	**P5 / BS seulement.** `true` si l'équipement a envoyé un `doorMode` non vide (serrure commandée) ; `false` s'il manque ou est vide. Résultat seul, sans motif de refus. `remote_open` / `remote_close` ne sont pas des pointages
`has_log_photo`	boolean	Déclare s'il y a une capture
`photo_status`	string	`absent`, `stored`, `omitted_too_large` ou `omitted_invalid`

5.4 Résumé des derniers événements (optionnel)

```
curl -s "https://api.bnetlink.com/connect/v1/sync/status" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Utile pour voir grossièrement s'il y a du nouveau ; **ce n'est pas** un curseur de niveau d'eau, et ne remplace pas votre propre point de reprise.

---

6. Webhook (optionnel)

Après enregistrement d'une URL HTTPS de callback, Connect POST vers vous à chaque événement (avec signature).

Recommandation : le Webhook sert uniquement de **réveil** ; la source d'autorité reste la reprise via `GET /events`.

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/webhooks" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"url\":\"https://example.com/hooks/connect\",\"secret\":\"at-least-16-chars!\",\"event_types\":[]}"
```

Autres : `GET /webhooks` (liste), `PUT /webhooks/{id}`, `POST /webhooks/{id}/test`.

Il n'existe pas de `GET /webhooks/{id}` isolé : prenez les champs de détail depuis le résultat de liste selon `id`.

`PUT` peut modifier `url` / `event_types` / `status` (`ACTIVE` | `DISABLED`) ; pour changer `secret`, fournissez la nouvelle valeur ; **l'omission conserve le secret d'origine**. `event_types` vide signifie abonnement à tous les types.

Points clés :

- En production, l'URL publique doit être en **HTTPS**, sans pointer vers un réseau privé / loopback.
- `secret` de 16–128 caractères, libre ; **n'utilisez pas** l'API Key comme secret Webhook.
- Liste / détail **ne renvoient plus** le secret en clair ; conservez-le à la création.
- Le corps HTTP du callback = JSON métier de l'événement (identique à `payload_json` parsé), **pas** la photo.

Vérification de signature (obligatoire)

Connect utilise HMAC-SHA256 :

```
chaîne signée = timestamp + "." + raw_body
X-Epordo-Signature = hex_lower( HMAC_SHA256(secret, chaîne signée) )
```

Les en-têtes incluent aussi : `X-Epordo-Timestamp`, `X-Epordo-Event-Id`, `X-Epordo-Event-Type`.

Le récepteur doit vérifier avec les **octets bruts du body** ; tout **2xx** signifie succès ; en cas d'échec, des retries ont lieu, pouvant aboutir à `DEAD`.

Exemple (pseudo-code):

```
expected = hmac_sha256_hex(secret, header["X-Epordo-Timestamp"] + "." + rawBody)
if expected != header["X-Epordo-Signature"]: reject
else: upsert_by_event_id(JSON.parse(rawBody)); return 200
```

File de livraison (diagnostic)

Méthode	Chemin	Description
GET	`/webhooks/outbox`	Lignes de file : `PENDING` / `FAILED` / `SENT` / `DEAD`
GET	`/webhooks/deliveries`	Audit de chaque tentative (code HTTP, durée, résumé d'erreur)
POST	`/webhooks/outbox/{id}/replay`	Rejouable uniquement si `FAILED` / `DEAD`

Exemple : à la réception d'un Webhook, notez seulement « il y a du nouveau », puis appelez aussitôt `GET /events` avec la fenêtre temporelle précédemment sauvegardée ; ainsi, même si un callback est perdu, les données métier ne le sont pas.

6.1 MQTT (optionnel, utilisable en parallèle du Webhook)

Connect peut publier le **même** JSON d'événement que le Webhook sur un Topic fixe. Peut tourner **en parallèle** du Webhook ; dédupliquez avec `event_id`.

Élément	Description
Topic	`epc/{tenant_id}/events/{event_type}` (fixé par la plateforme)
Charge utile	Même JSON métier que événement / Webhook
QoS	0 ou 1 (défaut 1)
Broker	Fourni par la plateforme ; **émettre un compte d'abonnement** dans la page MQTT du portail pour obtenir utilisateur / mot de passe
Connexion	Serveurs souvent en TCP ; navigateurs souvent en WSS (adresses exactes selon les indications du portail)

```
# Enregistrer les types d'événements à pousser(une entrée par locataire)
curl -s -X POST "https://api.bnetlink.com/connect/v1/mqtt" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"event_types\":[],\"qos\":1}"

# Émettre un compte d'abonnement(mot de passe affiché une seule fois)
curl -s -X POST "https://api.bnetlink.com/connect/v1/mqtt/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Notes : le compte d'abonnement **n'est pas** l'API Key, ni le secret Webhook. La communication équipement passe toujours par le serveur d'équipements et **ne devient pas** une connexion MQTT directe des terminaux.

---

7. Commandes standard

7.1 Création et interrogation

```
# Envoi(exemple : mise à l'heure)
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-time" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{}"

# Interroger l'état
curl -s "https://api.bnetlink.com/connect/v1/commands/{command_id}" \
  -H "Authorization: Bearer YOUR_API_KEY"

# Annulation possible uniquement si QUEUED
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/{command_id}/cancel" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Champs courants en cas de succès de création :

```
{
  "command_id": "cmd_a1b2c3d4e5f6",
  "sn": "TEST-SN-001",
  "command": "set-time",
  "state": "QUEUED"
}
```

Champ	Description
`command_id`	Ex. `cmd_` + hexadécimal
`state`	`QUEUED` → `SENT` → `SUCCEEDED` / `FAILED` / `CANCELLED`
`result`	Objet de résultat possible à l'état final (selon la commande)

- En-tête optionnel `X-Idempotency-Key` (≤128): même Key + même corps → renvoie la commande d'origine ; même Key + corps différent → `409` `idempotency_conflict`.
- Équipement en ligne : une commande ordinaire sans progrès pendant ~3 minutes échoue ; idle inter-paquets pour relecture fragmentée ~10 minutes. Au plus une `SENT` à la fois par machine.
- Équipement hors ligne : la commande reste en `QUEUED` et sera envoyée à la reconnexion (toujours sous contraintes de timeout et de quota).

Exemple : une instabilité réseau vous fait réessayer deux fois une ouverture. Avec le même `X-Idempotency-Key` et le même JSON, une seule entrée en file a lieu, évitant une double ouverture.

7.2 Liste des commandes

Nom standard	Suffixe de chemin	Rôle
`set-time`	`set-time`	Mise à l'heure
`query-status`	`query-status`	Interroger l'état de l'équipement
`unlock`	`unlock`	Ouverture à distance / maintien ouvert / fermeture
`set-user`	`set-user`	Envoyer l'enrollment d'une personne (modèles optionnels inclus)
`delete-user`	`delete-user`	Supprimer une ou plusieurs personnes
`list-user-ids`	`list-user-ids`	Relire la liste des identifiants personnels

`get-user`	`get-user`	Relire le personnel et le persister côté plateforme
`set-access-schedule`	`set-access-schedule`	Plages d'accès
`clear-managers`	`clear-managers`	Effacer les drapeaux administrateur (sans supprimer les personnes)
`pull-logs`	`pull-logs`	Recompléter les pointages sur une fenêtre temporelle

Actuellement **non fournis** : interface d'enregistrement distant, réglages de sonnerie, etc. Ne les intégrez pas.

7.3 `set-time`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-time" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"time\":\"20261108120000\"}"
```

`time` est optionnel, format `yyyyMMddHHmmss` ; s'il est omis, l'horloge locale du serveur au moment de la mise en file est utilisée.

7.4 `query-status`

Sans corps. En succès, `result.model` est le `dev_model` de cet échange (P5 et P7 envoient `P5`), pas le modèle vendu `model_code`. Firmware, capacités, etc. (les éléments non supportés par l'équipement peuvent être `null`).

7.5 `unlock`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/unlock" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"mode\":\"pulse\"}"
```

mode	Signification
`pulse` (défaut)	Ouvre une fois puis referme selon le délai de l'équipement
`hold`	Maintient la porte ouverte
`close`	Ferme

`SUCCEEDED` signifie seulement que l'équipement a confirmé l'exécution, **pas** que le contact de porte est déjà ouvert.

7.6 `set-user` (une personne · une machine)

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"user_id\":\"1001\",\"name\":\"Dupont\",\"role\":\"user\",\"card\":\"123456\"}"
```

Exemple de champs complets :

```
{
  "user_id": "1001",
  "name": "Dupont",
  "role": "user",
  "card": "123456",
  "password": "666",
  "valid_from": "20260101",
  "valid_to": "20261231",
  "replace": false,
  "face": "<Base64>",
  "photo": "<Base64>",
  "fingerprint": ["<Base64>"],
  "palm": ["<Base64>"]
}
```

- `user_id` obligatoire ; `fingerprint` / `palm` jusqu'à 10 emplacements ; **un tableau vide signifie vider cette modalité**.
- Connect **ne convertit pas** les formats de modèles.
- Pour les plages d'accès, utilisez `set-access-schedule` ; ne les empilez pas dans cette commande.

7.7 `delete-user` (plusieurs personnes possibles · une machine)

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/delete-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"user_ids\":[\"1001\",\"1002\"]}"
```

- Une commande peut supprimer plusieurs personnes (max 100); ne passez pas `user_id` et `user_ids` en même temps.
- **Pas** de « vider tout le personnel de la machine ».

7.8 `list-user-ids` / 7.9 `get-user`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/list-user-ids" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" -d "{}"

curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/get-user" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" \
  -d "{\"user_ids\":[\"1001\"]}"
```

- `get-user` accepte aussi `{"all":true}` (persistance silencieuse, pour éviter une tempête d'événements).
- Après persistance d'ids spécifiés, on obtient en général `enroll.data.received`.

7.10 `set-access-schedule`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-access-schedule" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"time_groups\":{\"mon\":[\"09001200\",\"14001800\"],\"tue\":[\"09001800\"]},\"user_ids\":[\"1001\",\"1002
```

Champ	Description
`time_groups`	Obligatoire. Clés `sun` … `sat` ; valeurs = tableaux de chaînes `HHmmHHmm`, **1–6 plages par jour**
`user_ids`	Optionnel ; omis / vide = tout le personnel sur l'équipement

Les plages n'existent que sur l'équipement. Pour vérifier, renvoyez `get-user` et consultez `access_schedules`.

Exemple : lundi 09:00–12:00 et 14:00–18:00 → `"mon": ["09001200", "14001800"]`.

7.11 `clear-managers` / `pull-logs`

- `clear-managers` : sans corps ; efface les drapeaux administrateur.
- `pull-logs` : `{"from":"…","to":"…"}` (8 ou 14 chiffres, étendue ≤31 jours).

---

8. Fan-out par liste de numéros de série

```
POST /commands/fan-out/set-user
POST /commands/fan-out/delete-user
POST /commands/fan-out/set-access-schedule
```

Champ commun : `sns` (tableau de chaînes non vides).

8.1 Principes

Sur une même machine, regroupez autant que possible la même opération dans **une** commande (selon les capacités de l'équipement), plutôt qu'« un HTTP par personne ».

Interface	Écriture recommandée	Comportement de file
Envoi de personnel	`users:[{user_id,…}]`	**Une commande par SN**
Suppression de personnel	`user_ids:[…]`	**Une commande par SN**
Plages d'accès	`time_groups` partagé + `user_ids` optionnel	**Une commande par SN**

8.2 Plafonds (défaut)

Élément	Défaut
Nombre de `sn` distincts	60 (peut descendre à 30 avec de grands modèles)
Nombre d'ids personnels	100
Nombre de commandes en file produites	200

Dépassement → `400` `fan_out_too_large`.

8.3 Envoi de personnel multi-machines (exemple)

Préférez `users[]` (chaque personne peut avoir nom / carte / modèles propres); **une seule commande en file par SN** :

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/fan-out/set-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sns\":[\"SN-A\",\"SN-B\"],\"users\":[{\"user_id\":\"1001\",\"name\":\"Dupont\",\"card\":\"123\"},{\"user
```

Les grands modèles (visage, etc.) peuvent dépasser le tampon d'une machine ; paginez ou réduisez les modèles avant d'envoyer.

8.4 Suppression multi-personnes multi-machines (exemple)

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/fan-out/delete-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sns\":[\"SN-A\",\"SN-B\"],\"user_ids\":[\"1001\",\"1002\"]}"
```

Forme de la réponse :

```
{
  "command": "delete-user",
  "queued": [
    { "sn": "SN-A", "user_ids": ["1001", "1002"], "command_id": "cmd_…" }
  ],
  "skipped": [
    { "sn": "SN-X", "reason": "device_disabled" }
  ]
}
```

Interrogez chaque `command_id` séparément. `skipped.reason` courants : `device_not_found`, `device_disabled`, liés au quota.

8.5 Même jeu de plages d'accès multi-machines (exemple)

```
{
  "sns": ["SN-001", "SN-002"],
  "time_groups": {
    "mon": ["09001200", "14001800"],
    "tue": ["09001800"]
  },
  "user_ids": ["1001", "1002", "1003"]
}
```

---

9. Groupes d'équipements et Scoped Key

Préfixe de chemin : `/device-groups`. Créer / renommer / activer-désactiver un groupe exige une Key de compte entier (ACCOUNT).

Il n'existe pas de `DELETE /device-groups/{id}` : pour désactiver, `PUT` avec `status` = `DISABLED`.

Méthode	Chemin	Description	
GET	`/device-groups`	Liste	
POST	`/device-groups`	Création : `{"code":"store-a","name":"Magasin A"}`	
GET	`/device-groups/{id}`	Détail + équipements membres	
PUT	`/device-groups/{id}`	Modifier `name` / `status` (`ACTIVE` \	`DISABLED` ; `code` non modifiable)
PUT	`/device-groups/{id}/devices/{sn}`	Ajouter un équipement déjà revendiqué	
DELETE	`/device-groups/{id}/devices/{sn}`	Retirer un membre	
POST	`/device-groups/{id}/commands/set-time`	Mise à l'heure en lot dans le groupe	

POST	`/device-groups/{id}/commands/query-status`	Interrogation de statut en lot dans le groupe
POST	`/device-groups/{id}/commands/unlock`	Ouverture en lot dans le groupe (`mode` comme en mono-machine)

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/device-groups" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" \
  -d "{\"code\":\"store-a\",\"name\":\"Magasin A\"}"

curl -s -X PUT "https://api.bnetlink.com/connect/v1/device-groups/1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s -X POST "https://api.bnetlink.com/connect/v1/device-groups/1/commands/query-status" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

La réponse des lots de groupe est toujours **200** (succès partiel autorisé), de forme similaire au fan-out : `queued[]` / `skipped[]` .

Notes :

- Personnel / suppression / plages d'accès **n'ont pas** de lot au niveau groupe ; utilisez le fan-out §8 (`sns` explicites).
- On peut émettre une API Key `scope_type=GROUPS` pour un groupe (voir §2.5).
- Une Key de groupe **ne peut pas** se connecter au portail, ni revendiquer / détacher / gérer Webhook, MQTT, clés, ni les groupes eux-mêmes.
- Si le statut du groupe est `DISABLED` , un envoi de niveau groupe peut renvoyer `409` `group_disabled` .

---

10. Limitation de débit et quotas journaliers (résumé)

- **Par minute** : débit de requêtes par Key ; plafond → `429` `rate_limited` (la réponse peut porter `Retry-After`).
- **Quota journalier (jour UTC)** : commandes « coûteuses » telles qu'ouverture, relecture complète, recomplétion, envoi / suppression de personnel, plages, etc. ; un usage anormal peut entraîner la désactivation automatique de la Key (`quota_exceeded_key_revoked`).
- Les rejeux idempotents **ne déduisent généralement plus** le quota. Les chiffres exacts dépendent du contrat / de la config plateforme.

---

11. Flux d'intégration recommandé

11.1 Boucle minimale (pointage)

1. Obtenir une API Key, confirmer avec `GET /me` .
2. `POST /devices` pour revendiquer ; renseigner l'adresse du serveur d'équipements et mettre en ligne.
3. (Optionnel) Enregistrer Webhook ou MQTT.
4. Périodiquement ou après réveil, reprendre avec `GET /events` ; stockez vous-même la progression.
5. Si une capture est nécessaire, `GET /events/{id}/photo` .

11.2 Envoi de personnel

1. Le côté métier maintient le personnel et les modèles.
2. Mono-machine `set-user` ; multi-machines `fan-out/set-user` .
3. Même jeu de règles d'accès : `set-access-schedule` ou fan-out partageant `time_groups` .
4. Vérification : `list-user-ids` / `get-user` .

11.3 Parc de la même série

- Les modèles vendus restent distincts (P5 et P7).
- Ne faites pas un `get-user` complet sur chaque machine.
- Prenez l'ERP ou une machine représentative comme référence ; sur les autres, réconciliez avec `list-user-ids` puis comblez les trous.
- Liste `GET /devices/{sn}/users?scope=compatible` : croisement seulement si le fabricant a indiqué la même famille et que le firmware est identique et non vide. Le `dev_model` du Header ne change pas le modèle vendu. Firmware vide : pas de croisement.

Annexe A. Codes d'erreur courants

code	Scénario typique
`missing_bearer` / `empty_key` / `invalid_key`	Échec d'authentification
`missing_sn`	Revendication sans numéro de série
`device_not_in_stock`	Numéro de série absent du stock constructeur
`device_not_for_tenant`	Stock non attribué à ce compte
`device_not_found`	Équipement inexistant ou hors périmètre
`device_already_registered` / `sn_already_registered`	Enregistrement en double
`device_disabled`	Équipement déjà désactivé
`invalid_set_user` / `invalid_delete_user` / `invalid_access_schedule`	Corps de commande invalide
`invalid_fan_out` / `fan_out_too_large`	Paramètres batch erronés ou hors limite
`command_not_found` / `command_not_cancelable`	Interrogation / annulation de commande
`idempotency_conflict`	Conflit de clé d'idempotence
`rate_limited` / `quota_exceeded` / `quota_exceeded_key_revoked`	Limitation / quota
`invalid_webhook_url` / `invalid_webhook_secret`	Paramètres Webhook
`forbidden_scope`	Key de groupe sur une interface de niveau compte
`last_active_api_key`	Tentative de désactiver la dernière Key active
`api_key_not_found`	Id de clé inexistant
`mqtt_already_exists`	Sortie MQTT déjà enregistrée pour le locataire ; ne pas POST en double
`outbox_not_replayable`	État de file de livraison non rejouable
`group_not_found` / `group_disabled`	Groupe inexistant ou désactivé

Annexe B. kind des credentials

kind	Emplacement	Description

`photo`	0	Photo en lumière visible
`face`	0	Modèle facial
`fingerprint`	0–9	Empreinte
`palm`	0–9	Paume / veine de paume
`card`	0	Numéro de carte (UTF-8)
`password`	0	Mot de passe équipement (UTF-8)

Un bloc décodé dépassant environ **1 Mo** est refusé. La plateforme ne convertit pas les modèles entre firmwares.

Annexe C. Documentation et support

Ressource	Description
Ce document / PDF	Guide API public (PDF téléchargeable depuis « Aide » du portail)
Portail	Revendication navigateur, essais de commandes, émission de comptes d'abonnement MQTT, etc.
ApiPost / Postman	Collection d'import fournie à la livraison projet (annexe technique)
Remontée de problèmes	Canaux support commercial / technique

Annexe D. Référence rapide des interfaces

Catégorie	Méthode et chemin				
Santé / identité	`GET /health` · `GET /me` · `GET	POST /credentials · PUT /credentials/{id}`			
Équipements	`POST	GET /devices · GET	DELETE /devices/{sn} · POST .../disable	enable · GET .../users...`	
Événements	`GET /events` · `GET /events/{id}` · `GET /events/{id}/photo` · `GET /sync/status`				
Webhook	`GET	POST /webhooks · PUT /webhooks/{id} · POST .../test · GET .../outbox	deliveries · POST .../outbox/{id}/replay` (pas de GET by id isolé)		
MQTT	`GET	POST /mqtt · PUT	DELETE /mqtt/{id} · POST .../test · GET	POST	DELETE /mqtt/credentials` · outbox/deliveries/replay
Commandes	`POST /devices/{sn}/commands/{name}`				

	· `GET /commands/{id}` · `POST /commands/{id}/cancel`		
Batch	`POST /commands/fan-out/set-user	delete-user	set-access-schedule`
Groupes	`GET	POST /device-groups · GET	PUT /device-groups/{id} · `membres` `PUT/DELETE · lots de groupe` set-time / query-status / unlock` (pas de DELETE du groupe lui-même)

Commandes standard `{name}` : `set-time`, `query-status`, `unlock`, `set-user`, `delete-user`, `list-user-ids`, `get-user`, `set-access-schedule`, `clear-managers`, `pull-logs`.

Fin du document · Guide API EPORDO Connect v1.1.5