Guía de API de EPORDO Connect (Español)

Ítem	Contenido
Versión del documento	**1.1.5**
Fecha de publicación	2026-11-11
Audiencia	Desarrolladores del integrador, intermediarios / integradores, personal de entrega de proyectos
Producto	**EPORDO Connect** (plataforma en la nube para acceso de dispositivos de terceros)
Formato de entrega	Este origen Markdown + **PDF** descargable desde el portal (Ayuda → Descargar guía de API)

> Este documento describe la **API REST del lado del inquilino** (prefijo de ruta `/connect/v1`). La comunicación entre el dispositivo y la nube la realiza Connect; el integrador **no necesita** analizar ni implementar el protocolo binario / de texto original del dispositivo.

> Los ejemplos usan por defecto una Base URL de estilo producción; en pruebas de integración, sustitúyala por la dirección proporcionada en el entorno de entrega.

> **Convención:** Las rutas siguientes son relativas a la Base URL. Los nombres de campos JSON se rigen por este documento (por ejemplo, el cuerpo de negocio del evento es **`payload_json`**, que debe parsearse con `JSON.parse`).

Historial de revisiones

Versión	Fecha	Descripción
1.1.5	2026-11-11	§5.3.1: `door_mode` y `door_opened` en los cinco idiomas; `photo_status` es `absent`, `stored`, `omitted_too_large` u `omitted_invalid`
1.1.4	2026-11-10	§5.3: tipos vigentes `door.remote.recorded` y `device.presence`; ventana en línea de 10 minutos, barrido fuera de línea ~5 minutos, y nombres antiguos solo en el flanco correspondiente
1.1.3	2026-11-08	Mejora: cabecera §5.3 = `event_type` y se añade `door.open.recorded`; apéndice D: `GET /commands/{id}` + `POST .../cancel`
1.1.2	2026-11-08	Corrección: el campo del evento es `event_type` (no `type`); la API de foto de captura devuelve JSON/`data_base64` (no un flujo de imagen directo); no existe `GET /webhooks/{id}`; no existe `DELETE /device-groups/{id}` (usar `status=DISABLED`); la retención se alinea a unos 3–14 días
1.1.1	2026-11-08	Verificación frente a producción: se completan rutas y ejemplos de API Key / grupos; verificación de firma Webhook y outbox; ejemplos de `payload_json` de eventos y retención; ejemplo de fan-out de personal; índice
1.1.0	2026-11-08	Se añade visión general del sistema y relaciones de comunicación; se complementan ejemplos cURL y de escenarios en cada capítulo; se añade desvinculación de dispositivos; se añade historial de revisiones; entrega PDF

1.0.0	2026-11-07	Primera versión pública: reclamación, eventos, Webhook/MQTT, comandos estándar, fan-out, resumen de códigos de error

Si solo se corrigen erratas, el historial añadirá una versión de parche; si cambia el comportamiento de la API, se subirá la versión menor y se anotará en el capítulo correspondiente.

Índice

1. Visión general del sistema
2. Entorno y autenticación (incluye gestión de API Key)
3. Formato unificado de errores
4. Dispositivos
5. Eventos
6. Webhook / MQTT
7. Comandos estándar
8. Fan-out por lista de números de serie
9. Grupos de dispositivos y Scoped Key
10. Límites de tasa y cuota diaria
11. Flujo de integración recomendado

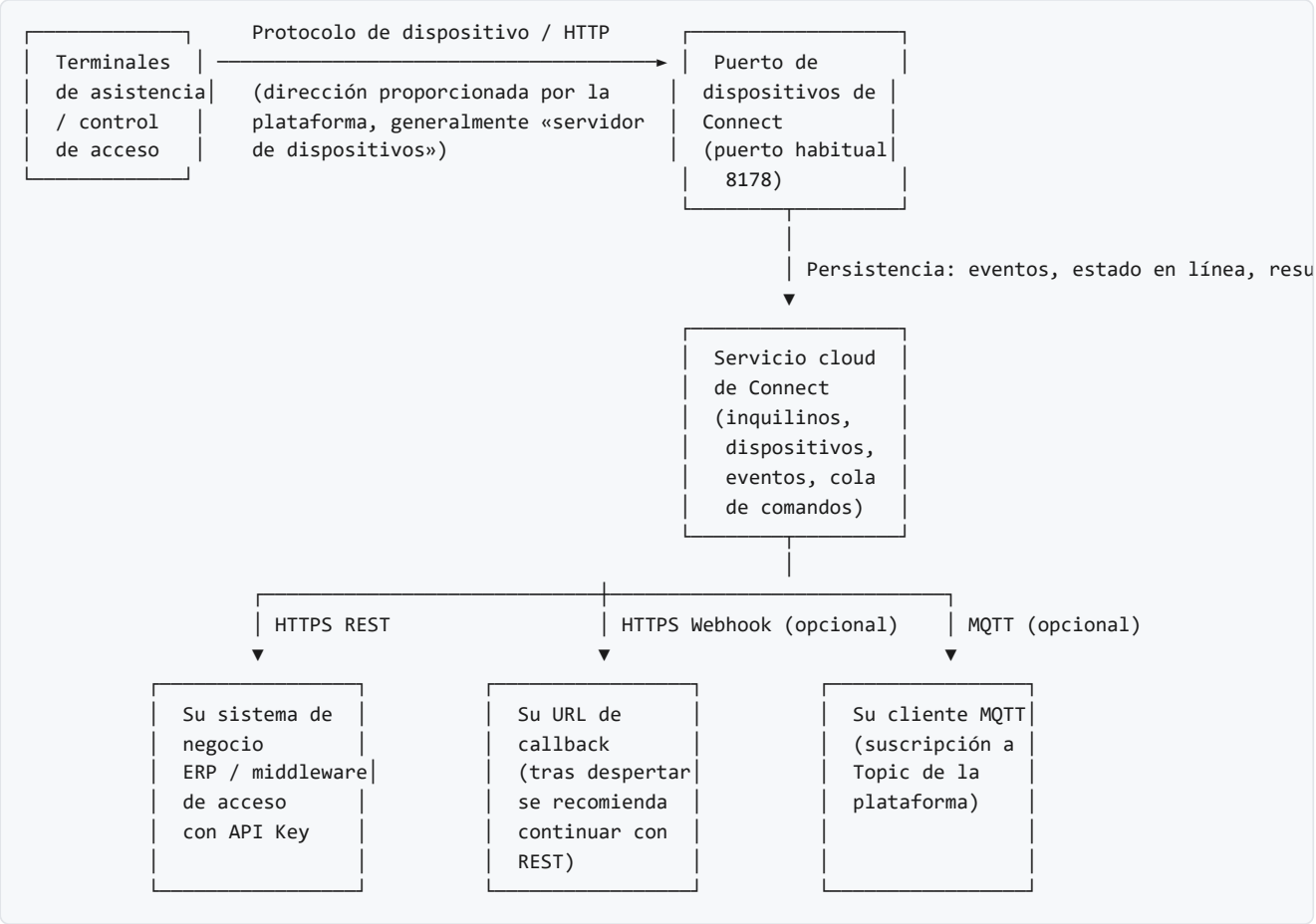
- Apéndice A Códigos de error · Apéndice B kind de credenciales · Apéndice C Soporte · Apéndice D Referencia rápida de API

---

1. Visión general del sistema (lea esta sección primero)

1.1 Posición de Connect en la arquitectura

Puede entenderse como la «capa de adaptación en la nube entre los dispositivos y su sistema de negocio»:



No mezcle estas dos líneas:

Línea	Quién se conecta con quién	Qué debe hacer usted
Línea de dispositivos	Terminal → servidor de dispositivos de Connect (ejemplo: `http://api.bnetlink.com:8178/`)	En el menú del dispositivo, indique la dirección dada por la plataforma; **no** construya mensajes de dispositivo en su código
Línea de negocio	Su servidor → REST de Connect (ejemplo: `https://api.bnetlink.com/connect/v1`)	Llame a las APIs de este documento con **Bearer API Key**

El portal (navegador) permite al personal de negocio reclamar, ver eventos y probar comandos; **para la integración formal use su propio backend + API Key**; no incruste la Key en páginas públicas.

1.2 Protocolo de comunicación del dispositivo (perspectiva del integrador)

El terminal usa el protocolo de dispositivo acordado por el fabricante (p. ej. máquina dinámica BS, máquina ordinaria FK, etc.) para comunicarse con el **servidor de dispositivos** de Connect. Connect se encarga de:

1. Identificar el número de serie y el inquilino propietario;
2. Organizar reportes de fichaje / registro, etc., en un **JSON de evento** unificado;
3. Traducir los **comandos estándar** que usted envía por REST a mensajes que el dispositivo puede ejecutar, ponerlos en cola y escribir el resultado.

Por tanto, en la documentación de integración aparecen nombres estándar como `set-user`, `delete-user`, **no** cadenas de códigos de comando internos del dispositivo. Solo necesita:

- Reclamar el número de serie `sn`;
- Obtener eventos / recibir Webhook o MQTT;

- Invocar comandos estándar y consultar el estado de `command_id`.

1.3 Ciclo de vida típico (ejemplo)

Ejemplo: una tienda instala un nuevo terminal facial y lo conecta a su middleware de asistencia.

1. En fábrica, el fabricante ya escribió ese número de serie en el «inventario vendible» y lo vinculó a su inquilino.
2. Usted (o el portal) llama a `POST /devices`, `sn=ABC123` → **reclamación correcta**.
3. En la configuración del dispositivo se indica la dirección del servidor de dispositivos y se conecta a la red → comienza el heartbeat / reporte.
4. Un empleado pasa la cara → Connect genera el evento `punch.recorded` → puede obtenerlo con `GET /events`, o ser despertado primero por Webhook/MQTT y luego extraer.
5. Desde el ERP envía el empleado «1001»: `POST /devices/ABC123/commands/set-user`.
6. Consulta periódicamente `GET /commands/{command_id}` hasta `SUCCEEDED` o `FAILED`.
7. Si la tienda retira el equipo: `DELETE /devices/ABC123` **desvincula**; el número de serie vuelve a no reclamado y podrá reclamarse de nuevo; los eventos históricos de fichaje se conservan.

1.4 Qué hace / no hace Connect

Hace:

- Reclamar / desactivar / activar / desvincular dispositivos;
- Recibir eventos de fichaje, cambios de registro, etc.;
- Enviar comandos estándar (sincronizar hora, abrir puerta, personal, franjas de acceso, etc.);
- Webhook opcional, MQTT opcional.

No hace:

- Turnos, cálculo salarial, motor de reglas de asistencia (eso corresponde al sistema de negocio u otros productos);
- Guardar por usted el cursor de «hasta qué evento se leyó» (guárdelo en su propia base de datos);
- Convertir automáticamente formatos de plantillas faciales / de huella entre firmwares.

1.5 Tres bloques que el integrador debe cuidar a diario

Capacidad	Descripción
Dispositivos	Reclamar `sn`, listar, desactivar / activar, desvincular
Eventos	Obtener; Webhook y/o MQTT opcionales como despertador o canal paralelo
Comandos	Sincronizar hora, consultar estado, abrir puerta, personal, franjas; se puede enviar en lote por lista de números de serie (fan-out)

---

2. Entorno y autenticación

2.1 Base URL

Entorno	Prefijo REST (ejemplo)
Producción	Lo proporciona el contrato / la entrega, con forma `https://api.bnetlink.com/connect/v1`
Pruebas locales (si está habilitado)	`http://127.0.0.1:8080/connect/v1`

Las rutas siguientes son relativas a este prefijo. Por ejemplo, `GET /me` = `GET {base}/me`.

Servidor de dispositivos (para el terminal), ejemplo: `http://api.bnetlink.com:8178/` (según la entrega).

2.2 Autenticación

Excepto `GET /health`, las cabeceras de la petición deben incluir:

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

Notas:

- Una API Key corresponde a un **inquilino** y solo puede acceder a los datos de ese inquilino.
- La Key se crea en la página «Claves» del portal o se entrega al abrir la cuenta; el **texto en claro solo aparece una vez**; guárdelo con cuidado.
- El **secreto de firma Webhook** y la **contraseña de suscripción MQTT no** son la misma cadena que la API Key.
- La cuenta de inicio de sesión del portal (usuario + contraseña) solo sirve para el navegador y **no** sustituye a la API Key para llamar a REST.

2.3 Comprobación de salud (ejemplo)

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

Éxito:

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

2.4 Inquilino actual (ejemplo)

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

En caso de éxito, la respuesta incluye `tenant_id`, plan, ámbito de la Key, etc. (los campos según la respuesta real).

Ejemplo: Si el `scope_type` devuelto es de ámbito de grupo (`GROUPS`), algunas APIs a nivel de cuenta (como reclamar, desvincular, gestionar Webhook / claves / grupos) devolverán `403` `forbidden_scope`.

2.5 Gestión de API Key (REST)

La página «Claves» del portal se apoya en las siguientes APIs. Al crear, el texto en claro **solo se devuelve una vez**.

```
# Lista (sin texto en claro)
curl -s "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"

# Crear Key de cuenta completa
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\"}"

# Crear Key limitada a grupos (group_ids deben ser grupos ACTIVE de este inquilino)
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]}"

# Desactivar / reactivar (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\"}"
```

Ejemplo de creación correcta (guarde de inmediato `api_key`):

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

Nota: no se puede desactivar la **última** Key `ACTIVE` del inquilino (`409` `last_active_api_key`). Las Keys de grupo **no** pueden iniciar sesión en el portal.

---

3. Formato unificado de errores

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

HTTP	Significado (resumen)
400	Parámetros no válidos
401	Sin Key / Key inválida o desactivada
403	Permisos insuficientes (p. ej. Key de grupo llamó a una API a nivel de cuenta)
404	El recurso no existe o no pertenece a este inquilino
409	Conflicto (reclamación duplicada, dispositivo desactivado, conflicto de idempotencia, etc.)
429	Límite por minuto o cuota diaria de comandos alcanzada

Ejemplo: Enviar apertura de puerta a un dispositivo desactivado → HTTP `409`, `code` = `device_disabled`.

Los `code` habituales se detallan en el Apéndice A.

---

4. Dispositivos

4.1 Reclamar dispositivo

```
POST /devices
```

```
{ "sn": "TEST-SN-001", "alias": "Torniquete recepción" }
```

```
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\":\"Torniquete recepción\"}"
```

Ejemplo de éxito:

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

Notas:

- Solo se pueden reclamar números de serie que estén **en el inventario del fabricante ya vinculados a este inquilino y aún no reclamados**.
- Protocolo / modelo / firmware vienen del preset del fabricante; el cuerpo de la petición **no** puede falsificarlos. El modelo de venta es el nombre corto `P5` o `P7`. El `dev_model` del Header no lo cambia.
- Número de serie desconocido → `404` `device_not_in_stock`.
- Asignado a otro inquilino → `403` `device_not_for_tenant`.
- Ya reclamado por este inquilino → `409` `device_already_registered`.
- El nombre de campo externo es siempre `sn`.

Ejemplo: La etiqueta del embalaje es `B515…`, pero el fabricante aún no ha escrito ese número en su inventario → la reclamación dará `device_not_in_stock`; contacte al fabricante para el preset y reintente.

4.2 Lista / detalle de dispositivos

```
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 lista admite cursor: `next_cursor` es opaco; en la página siguiente páselo tal cual en `cursor=`.

4.3 Desactivar / activar

```
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"
```

- **Desactivar**: sigue perteneciendo a esta cuenta, pero ya no se pueden enviar comandos (`409` `device_disabled`). Los eventos reportados por el dispositivo suelen poder seguir persistiendo (según el registry).
- **Activar**: restaura la posibilidad de enviar comandos.
- Desactivar ≠ desvincular.

4.4 Desvincular dispositivo

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

Ejemplo de éxito:

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

Efectos:

- El dispositivo se elimina de esta cuenta; el inventario del fabricante vuelve a «no reclamado» y **puede reclamarse de nuevo**;
- El registro de personal de esa máquina en la plataforma se borra; los comandos en cola se cancelan;
- **Los eventos históricos de fichaje se conservan**;
- Las sub-Keys de ámbito de grupo **no** pueden llamarlo (`403`).

4.5 Personal y credenciales en el dispositivo (registro actual en la plataforma)

Método	Ruta	Descripción	
GET	`/devices/{sn}/users`	Lista de personal; opcionales `scope=device\`	compatible `,` q`
GET	`/devices/{sn}/users/{userId}`	Detalle de personal y metadatos de credenciales	
GET	`/devices/{sn}/users/{userId}/credentials/{kind}/{slot}`	Cuerpo de la credencial (`data_base64`)	

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

`kind` solo admite: `photo` , `face` , `fingerprint` , `palm` , `card` , `password` (véase Apéndice B).

---

5. Eventos

Connect **no guarda** su cursor de lectura. Registre en su propia base de datos el `event_id` / tiempo ya procesado.

5.1 Obtener eventos

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

Obtener el último día por hora de persistencia (ejemplo; sustituya por ISO-8601 real):

```
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"
```

Resumen de reglas:

- Los parámetros de tiempo son ISO-8601; cuando inicio y fin van emparejados, el intervalo no supera **7 días**.
- `from` / `to` = hora de ocurrencia del negocio; `received_from` / `received_to` = hora de persistencia en la plataforma.
- Cada ítem de la lista incluye la **cadena** `payload_json` (forma textual del JSON de negocio) y **no** incluye la imagen grande de captura.
- La plataforma conserva eventos en caliente según la configuración del inquilino: por defecto unos **7 días**, rango configurable unos **3–14 días** (según contrato / configuración de la plataforma); el cómputo de retención usa `received_at`. Tras caducar, `GET /events/{id}` puede pasar a no encontrado. Persista a tiempo.
- En las consultas de lista, «cada par de ventanas temporales como máximo 7 días» es una limitación de **ventana de consulta**, distinta de los días de retención en disco.

Forma de un ítem de lista (ilustrativa; nombres de campo alineados con producción):

```
{
  "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\":17617
}
```

Haga un `JSON.parse` de `payload_json`. Tenga en cuenta que el campo externo es `event_type`, no `type`. El cuerpo HTTP de las notificaciones Webhook / MQTT es el **JSON de negocio tras parsear** `payload_json` (no el envoltorio de fila de lista completo).

Ejemplo: El Webhook falló de madrugada y se perdieron 20 registros. Al amanecer, use `received_from` para cubrir el intervalo de la noche anterior, pagine la extracción y deduplique por `event_id` al escribir en la base de negocio.

5.2 Evento individual y foto de captura

```
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"
```

La API de foto solo está disponible cuando el fichaje ya tiene la captura persistida (habitualmente `photo_status=stored`). La respuesta es **JSON**, no un flujo de bytes de imagen directo, por ejemplo:

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

Decodifique `data_base64` y guárdelo en disco. Sin foto → `404` `photo_not_found`; evento inexistente → `404` `event_not_found`. La lista / el detalle no incluyen bytes de imagen.

5.3 Tipos de evento habituales

event_type	Significado
`punch.recorded`	Fichaje
`enroll.data.received`	Resumen de registro de personal / cambio de credenciales
`command.completed` / `command.failed`	Estado final del comando
`door.state.changed`	Cambio de estado del contacto magnético de puerta (reporte activo del dispositivo)
`door.remote.recorded`	Acción remota de puerta (`mode` : pulse/hold/close; no es un fichaje)
`device.presence`	Flanco en línea / fuera de línea (`online` : 1/0)

Las filas nuevas usan solo los tipos de arriba. `online` de `GET /devices` es `1` si `last_seen_at` está dentro de **10 minutos**; si no, `0`. `device.presence` solo se escribe en el flanco: un reporte escribe `online=1`; un barrido de unos **5 minutos** escribe `online=0`, así que el evento fuera de línea aparece unos **10–15 minutos** después del último paquete. `occurred_at` es el último reporte; `received_at` es cuando Connect anotó el flanco. Los nombres antiguos `device.online` / `device.offline` y `door.open.recorded` / `door.close.recorded` siguen filtrando y suscribiendo, pero solo el flanco correspondiente.

5.3.1 Campos de `payload_json` de fichaje (tras parsear)

Campo	Tipo	Descripción
`event_id`	string	Id del evento, clave primaria de deduplicación
`sn`	string	Número de serie del dispositivo
`user_id`	string	Identificador de personal
`occurred_at`	number	Hora de ocurrencia del negocio, marca de tiempo en **milisegundos**
`device_time`	string	Cadena de hora local del dispositivo (habitualmente `yyyyMMddHHmmss`)
`verify_mode`	string	Descripción del modo de verificación (p. ej. Face / Card)
`door_mode`	string	Opcional. Texto de puerta del dispositivo (p. ej. `hand_open` / `open`)
`door_opened`	boolean	**Solo P5 / BS.** `true` si el dispositivo envió un `doorMode` no vacío (cerradura accionada); `false` si falta o está en blanco. Solo el resultado, sin motivo de rechazo. `remote_open` / `remote_close` no son fichajes
`has_log_photo`	boolean	Si declara tener captura
`photo_status`	string	`absent` , `stored` , `omitted_too_large` o `omitted_invalid`

5.4 Resumen del evento más reciente (opcional)

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

Sirve para ver a grandes rasgos si hay eventos nuevos; **no** es un cursor de marca de agua y no sustituye su propio punto de continuación de lectura.

---

6. Webhook (opcional)

Tras registrar una URL de callback HTTPS, cuando haya eventos Connect hará POST hacia usted (con firma).

Recomendación: use el Webhook solo como **despertador**; los datos autoritativos siguen basándose en la continuación con `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\":[]}"
```

Otros: `GET /webhooks` (lista), `PUT /webhooks/{id}`, `POST /webhooks/{id}/test`.

No existe un `GET /webhooks/{id}` independiente: obtenga los campos de detalle por `id` desde el resultado de la lista.

`PUT` puede cambiar `url` / `event_types` / `status` (`ACTIVE` | `DISABLED`); al cambiar `secret` indique el nuevo valor; **si se omite se conserva el secret original**. `event_types` vacío significa suscribirse a todos los tipos.

Puntos clave:

- En producción, la URL pública debe ser **HTTPS** y no puede apuntar a red privada / loopback.
- `secret` de 16–128 caracteres, elegido por usted; **no** use la API Key como secret de Webhook.
- La lista / el detalle **ya no devuelven** el secret en claro; guárdelo al crearlo.
- El cuerpo HTTP del callback = JSON de negocio del evento (igual que tras parsear `payload_json`), **no** la foto.

Verificación de firma (obligatoria)

Connect usa HMAC-SHA256:

```
cadena de firma = timestamp + "." + raw_body
X-Epordo-Signature = hex_lower( HMAC_SHA256(secret, cadena de firma) )
```

Las cabeceras también incluyen: `X-Epordo-Timestamp`, `X-Epordo-Event-Id`, `X-Epordo-Event-Type`.

El receptor debe verificar la firma con los **bytes originales del body**; cualquier **2xx** indica éxito; si falla se reintentará y al final puede pasar a `DEAD`.

Ejemplo (pseudocódigo):

```
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
```

Cola de entrega (diagnóstico)

Método	Ruta	Descripción
GET	`/webhooks/outbox`	Filas de cola: `PENDING` / `FAILED` / `SENT` / `DEAD`
GET	`/webhooks/deliveries`	Auditoría de cada intento (código de estado, duración, resumen de error)
POST	`/webhooks/outbox/{id}/replay`	Solo `FAILED` / `DEAD` pueden reproducirse

Ejemplo: Tras recibir el Webhook, solo registre «hay datos nuevos» y llame de inmediato a `GET /events` con la ventana temporal guardada la vez anterior; así, aunque se pierda un callback, no se pierden datos de negocio.

6.1 MQTT (opcional; puede usarse junto con Webhook)

Connect puede enviar el mismo JSON de evento que el Webhook a un Topic fijo. Puede usarse **en paralelo** con Webhook; deduplique con `event_id`.

Ítem	Descripción
Topic	`epc/{tenant_id}/events/{event_type}` (fijo de la plataforma)
Carga	El mismo JSON de negocio que evento / Webhook
QoS	0 o 1 (por defecto 1)
Broker	Lo proporciona la plataforma; en la página MQTT del portal **emita una cuenta de suscripción** para obtener usuario / contraseña
Conexión	En servidor suele usarse TCP; en navegador suele usarse WSS (direcciones concretas según las indicaciones del portal)

```
# Registrar los eventos a enviar (uno por inquilino)
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}"

# Emitir cuenta de suscripción (la contraseña solo se muestra una vez)
curl -s -X POST "https://api.bnetlink.com/connect/v1/mqtt/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Notas: la cuenta de suscripción **no** es la API Key ni el secret de Webhook. La comunicación del dispositivo sigue por el servidor de dispositivos y **no** cambia a MQTT directo del dispositivo.

---

7. Comandos estándar

7.1 Crear y consultar

```
# Enviar (ejemplo: sincronizar hora)
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 "{}"

# Consultar estado
curl -s "https://api.bnetlink.com/connect/v1/commands/{command_id}" \
  -H "Authorization: Bearer YOUR_API_KEY"

# Solo QUEUED se puede cancelar
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/{command_id}/cancel" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Campos habituales al crear con éxito:

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

Campo	Descripción
`command_id`	p. ej. `cmd_` + hexadecimal
`state`	`QUEUED` → `SENT` → `SUCCEEDED` / `FAILED` / `CANCELLED`
`result`	En estado final puede haber un objeto de resultado (según el comando)

- Cabecera opcional `X-Idempotency-Key` (≤128): misma Key + mismo cuerpo → se devuelve el comando original; misma Key + cuerpo distinto → `409` `idempotency_conflict`.
- Con el dispositivo en línea: los comandos ordinarios fallan tras unos 3 minutos sin progreso; en lecturas por paquetes, el ocio entre paquetes es de unos 10 minutos. Como máximo una `SENT` a la vez por máquina.
- Con el dispositivo fuera de línea, el comando permanece en `QUEUED` y se envía al conectar (sigue sujeto a timeouts y cuota).

Ejemplo: Por inestabilidad de red reintentó dos veces la apertura de puerta. Si ambas usan el mismo `X-Idempotency-Key` y el mismo JSON, solo se encola una vez de verdad, evitando abrir la puerta dos veces.

7.2 Listado de comandos

Nombre estándar	Sufijo de ruta	Función
`set-time`	`set-time`	Sincronizar hora
`query-status`	`query-status`	Consultar estado del dispositivo
`unlock`	`unlock`	Apertura remota / mantener abierta / cerrar
`set-user`	`set-user`	Enviar registro de una persona (con plantillas opcionales)
`delete-user`	`delete-user`	Eliminar una o varias personas
`list-user-ids`	`list-user-ids`	Leer de vuelta la lista de identificadores de personal

`get-user`	`get-user`	Leer de vuelta el personal y persistirlo en el registro de la plataforma
`set-access-schedule`	`set-access-schedule`	Franjas de acceso
`clear-managers`	`clear-managers`	Borrar marcas de administrador (sin borrar personas)
`pull-logs`	`pull-logs`	Recuperar fichajes por ventana temporal

Actualmente **no se ofrecen**: interfaz de registro remoto, ajuste de timbre, etc. No los integre.

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` es opcional, formato `yyyyMMddHHmmss`; si se omite, se usa el reloj local del servidor al emitir el comando.

7.4 `query-status`

Sin cuerpo. En éxito, `result.model` es el `dev_model` de este intercambio (P5 y P7 envían `P5`), no el modelo de venta `model_code`. También incluye firmware, capacidad, etc. (los ítems no soportados por el dispositivo pueden ser `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	Significado
`pulse` (por defecto)	Abrir una vez y cerrar según el retardo del dispositivo
`hold`	Mantener abierta
`close`	Cerrar

`SUCCEEDED` solo indica que el dispositivo confirmó la ejecución; **no** equivale a que el contacto magnético esté abierto.

7.6 `set-user` (una persona · una máquina)

```
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\":\"Juan\",\"role\":\"user\",\"card\":\"123456\"}"
```

Ejemplo de campos completos:

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

- `user_id` es obligatorio; `fingerprint` / `palm` como máximo 10 ranuras; **un array vacío significa vaciar esa modalidad**.
- Connect **no** convierte formatos de plantilla.
- Para franjas de acceso use `set-access-schedule`; no las meta en este comando.

7.7 `delete-user` (varias personas · una máquina)

```
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\"]}"
```

- Un comando puede eliminar a varias personas (máximo 100); no envíe a la vez `user_id` y `user_ids`.
- **No** se ofrece «vaciar todo el personal de la máquina».

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` también admite `{"all":true}` (persistencia silenciosa, para evitar tormenta de eventos).
- Tras persistir ids concretos suele generarse `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\",\"100
```

Campo	Descripción
`time_groups`	Obligatorio. Claves `sun` … `sat`; valores son arrays de cadenas `HHmmHHmm`, **1–6 tramos por día**
`user_ids`	Opcional; omitido / vacío = todo el personal del dispositivo

Las franjas solo existen en el dispositivo. Para comprobar, puede enviar de nuevo `get-user` y mirar `access_schedules`.

Ejemplo: Lunes mañana 09:00–12:00, tarde 14:00–18:00 → `"mon": ["09001200", "14001800"]`.

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

- `clear-managers`: sin cuerpo; borra marcas de administrador.
- `pull-logs`: `{"from":"…","to":"…"}` (8 o 14 dígitos, intervalo ≤31 días).

---

8. Fan-out por lista de números de serie

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

Campo común: `sns` (array de cadenas no vacío).

8.1 Principios

En un mismo dispositivo, la misma operación debe agruparse en **un** comando en la medida de lo posible (según la capacidad del dispositivo), no «una HTTP por persona».

API	Forma recomendada	Comportamiento de cola
Enviar personal	`users:[{user_id,…}]`	**Una por SN**
Eliminar personal	`user_ids:[…]`	**Una por SN**
Franjas de acceso	`time_groups` compartido + `user_ids` opcional	**Una por SN**

8.2 Límites (por defecto)

Ítem	Por defecto
Número de `sn` distintos	60 (con plantillas grandes puede bajar a 30)
Número de ids de personal	100
Número de comandos de cola generados	200

Si se supera → `400` `fan_out_too_large`.

8.3 Envío de personal a varias máquinas (ejemplo)

Se recomienda `users[]` (cada persona puede tener nombre / tarjeta / plantilla propios); **por cada SN solo se encola un** comando:

```
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\":\"Juan\",\"card\":\"123\"},{\"user_
```

Las plantillas grandes (cara, etc.) pueden superar el búfer de una sola máquina; pagine o reduzca las plantillas antes de enviar.

8.4 Eliminar varias personas en varias máquinas (ejemplo)

```
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\"]}"
```

Forma de la respuesta:

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

Consulte periódicamente cada `command_id`. `skipped.reason` habituales: `device_not_found`, `device_disabled`, relacionados con cuota.

8.5 Misma franja de acceso en varias máquinas (ejemplo)

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

---

9. Grupos de dispositivos y Scoped Key

Prefijo de ruta: `/device-groups`. Crear / renombrar / activar-desactivar grupos requiere una Key de cuenta completa (ACCOUNT).

No existe `DELETE /device-groups/{id}`: para desactivar, use `PUT` y establezca `status` en `DISABLED`.

Método	Ruta	Descripción	
GET	`/device-groups`	Lista	
POST	`/device-groups`	Crear: `{"code":"store-a","name":"Tienda A"}`	
GET	`/device-groups/{id}`	Detalle + dispositivos miembros	
PUT	`/device-groups/{id}`	Cambiar `name` / `status` (`ACTIVE` \	`DISABLED` ; no se puede cambiar `code`)
PUT	`/device-groups/{id}/devices/{sn}`	Añadir un dispositivo ya reclamado	
DELETE	`/device-groups/{id}/devices/{sn}`	Quitar miembro	
POST	`/device-groups/{id}/commands/set-time`	Sincronizar hora en lote dentro del grupo	

POST	`/device-groups/{id}/commands/query-status`	Consultar estado en lote dentro del grupo
POST	`/device-groups/{id}/commands/unlock`	Abrir puerta en lote dentro del grupo (`mode` igual que en una máquina)

```
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\":\"Tienda 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"
```

Las respuestas de lote a nivel de grupo son siempre **200** (se permite éxito parcial), con forma similar a fan-out: `queued[]` / `skipped[]` .

Notas:

- Personal / eliminación / franjas de acceso **no** tienen lote a nivel de grupo; use el fan-out del §8 (`sns` explícitos).
- Se puede emitir una API Key con `scope_type=GROUPS` para el grupo (véase §2.5).
- Las Keys de grupo **no** pueden iniciar sesión en el portal ni reclamar / desvincular / gestionar Webhook, MQTT, claves ni los grupos mismos.
- Si el estado del grupo es `DISABLED` , el envío a nivel de grupo puede devolver `409` `group_disabled` .

---

10. Límites de tasa y cuota diaria (resumen)

- **Por minuto**: tasa de peticiones por Key; al alcanzar el tope, `429` `rate_limited` (la respuesta puede incluir `Retry-After`).
- **Cuota diaria (día UTC)**: apertura de puerta, lectura completa de vuelta, recuperación, envío / eliminación de personal, franjas y otros «comandos caros»; un uso anómalo puede desactivar automáticamente la Key (`quota_exceeded_key_revoked`).
- Las reproducciones idempotentes suelen **no** consumir cuota de nuevo. Las cifras concretas dependen del contrato / la configuración de la plataforma.

---

11. Flujo de integración recomendado

11.1 Bucle mínimo (fichaje)

1. Obtenga la API Key y confirme con `GET /me` .
2. Reclame con `POST /devices` ; configure en el dispositivo la dirección del servidor de dispositivos y póngalo en línea.
3. (Opcional) Registre Webhook o MQTT.
4. Periódicamente o tras despertar, continúe la lectura con `GET /events` ; guarde el progreso usted mismo.
5. Si necesita la captura, use `GET /events/{id}/photo` .

11.2 Envío de personal

1. El lado de negocio mantiene personal y plantillas.
2. Una máquina: `set-user` ; varias máquinas: `fan-out/set-user` .
3. Misma regla de acceso: `set-access-schedule` o fan-out con `time_groups` compartido.

4. Verificación: `list-user-ids` / `get-user`.

11.3 Flota de la misma serie

- Los modelos de venta siguen separados (P5 y P7).
- No haga un `get-user` completo en cada máquina.
- Tome el ERP o una máquina representativa como autoridad; en las demás, concilie con `list-user-ids` y complete huecos.
- Lista `GET /devices/{sn}/users?scope=compatible`: solo cruza equipos si el fabricante asignó la misma familia y el firmware es el mismo y no está vacío. El `dev_model` del Header no cambia el modelo de venta. Firmware vacío: no hay cruce.

Apéndice A. Códigos de error comunes

code	Escenario típico
`missing_bearer` / `empty_key` / `invalid_key`	Fallo de autenticación
`missing_sn`	Reclamación sin número de serie
`device_not_in_stock`	El número de serie no está en el inventario del fabricante
`device_not_for_tenant`	El inventario no está asignado a esta cuenta
`device_not_found`	El dispositivo no existe o está fuera del ámbito de permisos
`device_already_registered` / `sn_already_registered`	Registro duplicado
`device_disabled`	Dispositivo desactivado
`invalid_set_user` / `invalid_delete_user` / `invalid_access_schedule`	Cuerpo del comando no válido
`invalid_fan_out` / `fan_out_too_large`	Parámetros de lote erróneos o fuera de límite
`command_not_found` / `command_not_cancelable`	Consulta / cancelación de comando
`idempotency_conflict`	Conflicto de clave de idempotencia
`rate_limited` / `quota_exceeded` / `quota_exceeded_key_revoked`	Límite de tasa / cuota
`invalid_webhook_url` / `invalid_webhook_secret`	Parámetros de Webhook
`forbidden_scope`	Key de grupo llamó a una API a nivel de cuenta
`last_active_api_key`	Intento de desactivar la última Key válida
`api_key_not_found`	El id de clave no existe
`mqtt_already_exists`	El inquilino ya registró salida MQTT; no repita el POST
`outbox_not_replayable`	El estado de la cola de entrega no admite reproducción
`group_not_found` / `group_disabled`	El grupo no existe o está desactivado

Apéndice B. kind de credenciales

kind	Ranura	Descripción
`photo`	0	Foto de luz visible
`face`	0	Plantilla facial
`fingerprint`	0–9	Huella
`palm`	0–9	Palma / vena de palma
`card`	0	Número de tarjeta (UTF-8)
`password`	0	Contraseña del dispositivo (UTF-8)

Un bloque decodificado que supere unos **1MB** será rechazado. La plataforma no convierte plantillas entre firmwares.

Apéndice C. Documentación y soporte

Material	Descripción
Este documento / PDF	Guía de API pública en español (el PDF se descarga desde «Ayuda» del portal)
Portal	Reclamación en navegador, prueba de comandos, emisión de cuentas de suscripción MQTT, etc.
ApiPost / Postman	Colección de importación entregada con el proyecto (anexo técnico)
Comentarios de incidencias	Canales de soporte comercial / técnico

---

Apéndice D. Referencia rápida de API

Categoría	Método y ruta			
Salud / identidad	`GET /health` · `GET /me` · `GET	POST /credentials · PUT /credentials/{id}`		
Dispositivos	`POST	GET /devices · GET	DELETE /devices/{sn} · POST .../disable	enable · GET .../users...`
Eventos	`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` (sin GET por id independiente)	

MQTT	`GET	POST /mqtt · PUT	DELETE /mqtt/{id} · POST .../test · GET	POST	DELETE /mqtt/credentials` · outbox/deliveries/replay
Comandos	`POST /devices/{sn}/commands/{name}` · `GET /commands/{id}` · `POST /commands/{id}/cancel`				
Lote	`POST /commands/fan-out/set-user	delete-user	set-access-schedule`		
Grupos	`GET	POST /device-groups · GET	PUT /device-groups/{id} · `miembros PUT/DELETE · a nivel de grupo` set-time / query-status / unlock` (sin DELETE del grupo en sí)		

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

Fin del documento · Guía de API de EPORDO Connect v1.1.5