



# myeztoll — API para socios: inicio rápido con ejemplos

---

|                                        |                                                                                                               |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------|
| API                                    | <a href="https://owner.myeztoll.com/api/partner/v1">https://owner.myeztoll.com/api/partner/v1</a>             |
| Referencia (pública, con «try it out») | <a href="https://owner.myeztoll.com/api/partner/v1/index">https://owner.myeztoll.com/api/partner/v1/index</a> |
| Versión de la guía                     | 2026-09-30 · API v1.4                                                                                         |
| Para                                   | los desarrolladores de los socios — el camino más corto de una clave a llamadas que funcionan                 |

**Referencia (pública, con «try it out»):** <https://owner.myeztoll.com/api/partner/v1/index>

## Qué es este documento

Cinco secciones cortas, una por vídeo de la lista **API para socios**. Cada una muestra las llamadas que de verdad necesita, con una petición y una respuesta con forma real. Todas las demás funciones — 54 en total, cada una con su ejemplo — están en la referencia de arriba y en el catálogo que sirve la propia API ( `GET /api/partner/v1` , sin clave).

## 1. Su clave y la primera llamada

**Cada llamada lleva una clave** en la cabecera `X-API-Key` .

| Credencial           | Aspecto                  | Actúa para                                                                          |
|----------------------|--------------------------|-------------------------------------------------------------------------------------|
| Clave de propietario | <code>aegispk-...</code> | Un propietario — la cuenta para la que se emitió                                    |
| Token de integración | <code>aegispi-...</code> | Muchos propietarios — cada llamada nombra al propietario en <code>X-Owner-Id</code> |

- Las claves las emitimos nosotros. Un propietario ve su propia clave, solo lectura, en su portal.
- Una **integración** actúa solo para los propietarios que la conectaron desde su portal. `GET /owners` los lista; un propietario que no le ha conectado (o se desconectó) responde `401` .
- **Pruebe primero en la zona de pruebas**, <https://dev-owner.myeztoll.com/api/partner/v1> , con su propia clave.

**El sobre.** Cada respuesta tiene la misma forma — salvo la descarga de una imagen de licencia o de un archivo de pruebas, que devuelve el archivo tal cual:

```
{ "success": true, "data": { ... }, "error": null }
```

Cuando `success` es `false` , `error` dice por qué.

### Convenciones

- Las **fechas** son UTC ISO-8601 ( `2026-07-31T14:05:00Z` ). Un rango incluye `from` y excluye `to` .
- **Id:** donde se acepte un id puede pasar **el nuestro o su propio** `externalId` . Las llamadas de creación aceptan `externalId` ; las respuestas lo devuelven.
- **Páginas:** cada lista es una página — `data = { items, page, pageSize, total, hasMore } . ? page=1&pageSize=100` (100 por defecto, máximo 1000). Siga pidiendo mientras `hasMore` sea `true` .

**Configure una vez** (zona de pruebas: `BASE=https://dev-owner.myeztoll.com/api/partner/v1` con su propia clave):

```
KEY=aegispk-... # or your aegispi-... integration token
BASE=https://owner.myeztoll.com/api/partner/v1
```

## Primera llamada — ¿de quién es esta clave?

```
curl -H "X-API-Key: $KEY" "$BASE/me"
```

```
{
  "success": true,
  "data": { "kind": "integration", "integrationId": "...", "name": "Your Platform",
            "connectedOwners": 12, "actingOwnerId": null },
  "error": null
}
```

## 2. Coches, conductores y reservas — con sus propios id

**Cree un coche** (ponga su propio id en `externalId`; desde entonces vale donde valga el nuestro):

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  $BASE/cars \
  -d '{"externalId":"CAR-001", "title":"Camry", "carTypeId":"...",
      "stateId":"NJ", "city":"Newark", "zipCode":"07102",
      "phoneNumber":"+12015550100", "carPlateNumber":"ABC1234",
      "carBrand":"Toyota", "color":"Black", "interiorColor":"Black",
      "year":2022}'
```

```
{ "success": true,
  "data": { "id": "5d73...", "externalId": "CAR-001",
            "title": "Camry", "plateNumber": "ABC1234" },
  "error": null }
```

`carTypeId` sale de `GET /car-types`; `stateId` de `GET /states`.

**Asocie el transpondedor de peaje** para que los peajes del coche se le asignen ( `tollProviderId` de `GET /toll-providers` ):

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  $BASE/cars/CAR-001/transponders \
  -d '{"tollProviderId":"...", "transponderNumber":"01234567890"}'
```

**Añada un conductor** (multipart; fotos de la licencia opcionales). Enviar el mismo correo otra vez actualiza al conductor en lugar de añadir otro:

```
curl -X POST -H "X-API-Key: $KEY" \
  -F firstName=John -F lastName=Doe -F email=john@acme.com -F externalId=DRV-001 \
  -F frontSideImage=@front.jpg \
  $BASE/drivers
```

**Registre una reserva** — un coche, un conductor y un periodo, todo con sus propios id. Es solo un registro: crearla no cobra a nadie.

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  $BASE/bookings \
  -d '{"externalId":"BK-001","carId":"CAR-001","renterId":"DRV-001",
    "dateFrom":"2026-08-01T00:00:00Z","dateTo":"2026-08-04T00:00:00Z"}'
```

```
{ "success": true,
  "data": { "id": "b1...", "externalId": "BK-001",
            "carId": "5d73...", "carExternalId": "CAR-001",
            "renterId": "9c1f...", "renterExternalId": "DRV-001",
            "status": 6, "statusName": "Approved" },
  "error": null }
```

**¿Se puede cobrar a este conductor?** `hasCard` = hay una tarjeta registrada; `isValid` = de verdad se puede cobrar (no caducada, no es un monedero que no podemos cobrar, no suspendida tras fallos).

```
curl -H "X-API-Key: $KEY" "$BASE/drivers/DRV-001/card"
```

Si no, `POST /bookings/BK-001/card-link` devuelve `cardRegistrationUrl` — una página donde el conductor añade una tarjeta.

**Las reservas son opcionales.** Sin ellas igualmente puede leer cada peaje por coche (sección 3).

---

## 3. Peajes y multas

---

**Cada peaje con precio de un periodo**, de las agencias de peaje y del GPS:

```
curl -H "X-API-Key: $KEY" \
  "$BASE/tolls?from=2026-07-01T00:00:00Z&to=2026-08-01T00:00:00Z&page=1&pageSize=100"
```

Una fila (recortada):

```
{
  "source": "agency",
  "externalCarId": "CAR-001",
  "plateNumber": "ABC1234", "plateState": "NJ",
  "occurredAtUtc": "2026-07-26T04:51:45Z",
  "agency": "GSP", "location": "KEY - Keyport",
  "agencyAmount": 0.79,
  "serviceFee": 5,
  "currency": "USD",
  "bookingSource": "turo", "bookingSourceReservationId": "41234567",
  "renterCharge": { "status": "charged", "amount": 6.45, "paidAmount": 18.2,
    "chargedAtUtc": "2026-07-28T09:12:03Z", "failureReason": null }
}
```

| Campo                               | Significado                                                                                                                                                                                                                                                                                            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>source</code>                 | <code>agency</code> (registro de la agencia) o <code>gps</code> (con precio a partir del recorrido del coche)                                                                                                                                                                                          |
| <code>agencyAmount</code>           | Lo que de verdad costó la carretera — cobre esto a su arrendatario                                                                                                                                                                                                                                     |
| <code>serviceFee</code>             | Nuestra tarifa aparte (puede ser nula). Nunca se suma al coste de la carretera                                                                                                                                                                                                                         |
| <code>bookingSource</code>          | De dónde vino la reserva ( <code>turo</code> , <code>hqrental</code> , <code>own</code> , ...), con su id de reserva                                                                                                                                                                                   |
| <code>renterCharge.status</code>    | ¿Lo devolvió el arrendatario?: <code>not_billed</code> , <code>estimate</code> , <code>pending</code> , <code>charged</code> , <code>declined</code> , <code>invoiced</code> , <code>reimbursed_by_platform</code> , <code>paid_cash</code> , <code>paid_manually</code> , <code>owner_absorbed</code> |
| <code>renterCharge.paymentId</code> | Los peajes cobrados en un mismo cargo lo comparten (por eso <code>paidAmount</code> puede superar <code>amount</code> )                                                                                                                                                                                |

## Otras formas de leerlos

- Un coche, cualquier periodo, sin reservas: `GET /cars/CAR-001/tolls?from=&to=` (peajes de agencia).
- Un alquiler: `GET /bookings/BK-001/tolls` — los peajes por GPS llevan un id que empieza por `gps` : .
- Multas: `GET /violations?from=&to=` — `amount` es la multa, `serviceFee` lo nuestro, `isPaid` si está pagada.

**Leer todas las páginas** (Node.js 18+):

```
const BASE = 'https://owner.myeztoll.com/api/partner/v1';

async function allTolls(key, from, to) {
  const rows = [];
  for (let page = 1; ; page++) {
    const url = `${BASE}/tolls?from=${from}&to=${to}&page=${page}&pageSize=1000`;
    const res = await fetch(url, { headers: { 'X-API-Key': key } });
    const body = await res.json(); // { success, data, error }
    if (!body.success) throw new Error(body.error);
    rows.push(...body.data.items);
    if (!body.data.hasMore) return rows; // keep asking while hasMore
  }
}
```

## 4. GPS y webhooks

**Dónde está ahora cada coche con rastreador** — GET /gps/positions. Cada elemento lleva lat, lon, speedMph, ignition y lo reciente que es: online y ageSeconds.

**Cuánto recorrió un coche** — GET /cars/CAR-001/gps/mileage?from=&to= → distanceMiles, distanceKm, positionCount. Para un alquiler: GET /bookings/BK-001/gps/mileage. Las posiciones se guardan 30 días; los peajes por GPS con precio son permanentes ( /tolls?source=gps ).

**Reciba avisos en lugar de preguntar** — registre un webhook:

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  -d '{"url":"https://example.com/hooks/myeZtoll"}' \
  "$BASE/webhooks"
```

```
{ "success": true,
  "data": { "id": "...", "url": "https://example.com/hooks/myeZtoll",
            "events": ["*"], "active": true, "secret": "whsec_..." },
  "error": null }
```

- "\*" = todos los eventos: toll.created, toll\_charge.updated, violation.created, violation.charge\_updated, owner.connected, owner.disconnected, dispute.evidence\_ready, dispute.evidence\_failed, dispute.deadline\_approaching.
- El **secreto se muestra una sola vez** — guárdelo. POST /webhooks/{id}/test envía un evento de prueba; GET /webhooks/{id}/deliveries es el registro de envíos.

**Compruebe cada envío.** Cabecera X-MyEzToll-Signature: t=<segundos unix>,v1=<hex>, donde v1 = HMAC-SHA256(secreto, "<t>.<cuerpo tal cual>"):

```
const crypto = require('crypto');

function verify(rawBody, header, secret) {
  const parts = Object.fromEntries(header.split(',').map((p) => p.split('=')));
  const signed = `${parts.t}.${rawBody}`; // "t.body", the body as received
  const expected = crypto.createHmac('sha256', secret).update(signed, 'utf8').digest('hex');
  const fresh = Math.abs(Date.now() / 1000 - Number(parts.t)) < 300; // reject replays
  return fresh && String(parts.v1).length === expected.length
    && crypto.timingSafeEqual(Buffer.from(expected), Buffer.from(parts.v1));
}
```

También se envían `X-MyEzToll-Event` (el tipo de evento) y `X-MyEzToll-Event-Id` — descarte repeticiones con él; un envío se reintenta con esperas crecientes durante hasta 1,5 días.

## 5. Sus cobros y disputas

Para cobros que hace **usted** en **su propia cuenta de Stripe**. Nunca llamamos a su cuenta de Stripe.

**1. Notifique el cobro** — qué peajes, multas y tarifas cubría:

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  -d '{"chargeId": "ch_3Qabc123", "amount": 25.64,
    "chargedAtUtc": "2026-09-20T14:02:11Z", "bookingId": "BK-001",
    "items": [{"type": "toll", "id": "34333995590"},
              {"type": "fee", "amount": 5, "label": "Service fee"}]}' \
  "$BASE/charges"
```

Cruzamos cada partida con nuestros registros y comprobamos la suma: si las partidas no suman el cobro, `warnings` lleva `sum_mismatch` («notifique sus propias tarifas como partidas de tipo fee»).

**2. Cuando Stripe abra una disputa sobre él**, notifíquela:

```
curl -X POST -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  -d '{"stripeDisputeId": "du_1UDk3z", "chargeId": "ch_3Qabc123",
    "reason": "product_not_received", "amount": 25.64,
    "evidenceDueBy": "2026-10-05T23:59:59Z"}' \
  "$BASE/disputes"
```

→ `"evidenceStatus": "collecting"`. Cuando el paquete está listo enviamos `dispute.evidence_ready`.

**3. Lea el paquete** — `GET /disputes/du_1UDk3z`. `evidence` usa **los nombres de campo de pruebas del propio Stripe**: pase los textos tal cual; por cada `{ "fileId": ... }` descargue el archivo (`GET /disputes/du_1UDk3z/files/{fileId}`), súbalo a su cuenta de Stripe (propósito `dispute_evidence`) y

ponga el id `file_...` de Stripe en su lugar. `recommendation` dice si conviene disputar o aceptar. Añada sus propios documentos con `POST /disputes/{id}/documents`.

#### 4. Envíela desde su lado y díganoslo:

```
curl -X PATCH -H "X-API-Key: $KEY" -H "Content-Type: application/json" \
  -d '{"status": "submitted"}' "$BASE/disputes/du_1UDk3z"
```

...y después `won` o `lost`.

---

## Siguientes pasos

- **Referencia con «try it out»:** <https://owner.myeztoll.com/api/partner/v1/index>
- **Documento OpenAPI:** <https://owner.myeztoll.com/api/partner/v1/openapi.json>
- **El catálogo, cada función con un ejemplo:** `GET https://owner.myeztoll.com/api/partner/v1`
- **Vídeos:** la lista **API para socios** (episodios 41–45).