DEVELOPER GUIDE

BUILD ON AVEO EOS.

Sell tickets on your own website, keep your CRM and systems in step, and run your own door apps on the AVEO EOS API. Version 2026-11-15.

37	20	16	3.1
ENDPOINTS	WEBHOOK EVENTS	PERMISSIONS	OPENAPI

CONTENTS

IN THIS DOCUMENT.

01 **QUICK START**
A key, your first request, your events and a checkout, in four steps.

02 **KEYS**
Secret, publishable and test keys, what each may do and where it may be used from.

03 **SELL ON YOUR SITE**
Hosted checkout, or your own checkout with Stripe's Payment Element.

04 **WEBHOOKS**
Hear about orders, check-ins and events the moment they happen, and trust only signed calls.

05 **THE RULES**
Idempotency, rate limits, paging, versions, request ids and errors.

06 **SANDBOX AND GOING LIVE**
Build against test data and Stripe test mode, then switch with a checklist.

01 QUICK START.

A key, your first request, your events and a checkout, in four steps.

01 Make a key

An admin opens API keys in the AVEO EOS menu, picks a ready-made set of permissions (or ticks their own), sets an expiry and makes the key. A secret key is shown once: keep it on your server.

02 Check it works

Ask the API who the key belongs to. Every answer is JSON and carries an X-Request-Id and the API version.

03 Read your events

Events come with their dates, venue and, with expand, their ticket types and live availability.

04 Start a checkout

Hold tickets for a buyer, then send them to the session's url, or complete it with your own checkout (part 03).

STEP 2: WHO IS THIS KEY?

```
curl https://eos.aveoevents.com/api/v1/ \
  -H "Authorization: Bearer eos_sk_..."
```

STEP 3: YOUR EVENTS, WITH THEIR TICKET TYPES

```
curl "https://eos.aveoevents.com/api/v1/events?expand=ticket_types" \
  -H "Authorization: Bearer eos_sk_..."
```

STEP 4: HOLD TWO TICKETS

```
curl https://eos.aveoevents.com/api/v1/checkout/sessions \
  -H "Authorization: Bearer eos_sk_..." \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{"event_id": 41, "items": [{"ticket_type_id": 9001, "quantity": 2}]}'
```

Base address: https://eos.aveoevents.com/api/v1. The whole API is also described in OpenAPI 3.1 at https://eos.aveoevents.com/api/v1/openapi.json, for code generators, Postman and Insomnia.

02 KEYS.

Secret, publishable and test keys, what each may do and where it may be used from.

KEY	STARTS	WHERE	MAY
Secret	eos_sk_	Your server only	Anything its permissions allow. Refused when sent from a web page.
Publishable	eos_pk_	Your web pages	Read the company and events, and start checkouts, only from the web addresses you allow.
Test	eos_sk_test_, eos_pk_test_	Your sandbox	The same, against your sandbox, with Stripe in test mode. Never works on your live company.

READY-MADE SETS.

SET	KIND	PERMISSIONS
Ticket sales on your website: Show events and take orders from your own site's server.	secret	company:read, events:read, checkout:write, orders:read, attendees:read
Browser (publishable): Safe to put in a web page: events and checkouts only, from your own web addresses.	publishable	company:read, events:read, checkout:write
Door scanning: Your own scanning app or turnstiles.	secret	events:read, attendees:read, checkins:read, checkins:write
Read only: Reporting, a CRM or a data warehouse.	secret	company:read, events:read, orders:read, attendees:read, checkins:read, discounts:read, reports:read, webhooks:read
Full access: Everything the API can do.	secret	Every permission

EACH KEY'S SETTINGS.

PERMISSIONS	Only what the key needs. Your plan sets the most a key may hold.
EXPIRY	7 days, 30 days, 90 days, 6 months, 1 year, Never.

IP ADDRESSES	Limit a secret key to your servers' addresses.
WEB ADDRESSES	A publishable key works only from the web addresses you list.
REQUESTS A MINUTE	Each key has its own limit, up to your plan's.
LOG	Every request a key makes is kept for 30 days, with its status and timing.

BY PLAN.

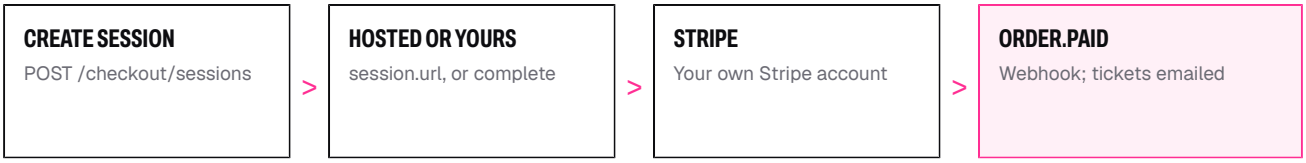
PLAN	API	REQUESTS A MINUTE, EACH KEY	WEBHOOK ENDPOINTS
Starter	Website checkout	60	None
Professional	Standard	120	2
Business	Full	300	10
Enterprise	Full	600	25
Corporate	Custom	1200	As agreed

Website checkout: read the company and events, and start checkouts. Standard: also read orders, guests, check-ins, discount codes and reports, send enquiries and manage webhooks. Full: every permission. The dedicated API package add-on gives Starter, Professional and Business every permission, a higher rate and up to 25 webhook endpoints.

03 SELL ON YOUR SITE.

Hosted checkout, or your own checkout with Stripe's Payment Element.

A checkout session holds the tickets, usually for 15 minutes (until expires_at). Then send the buyer to the session's url, a hosted checkout on your AVEO EOS shop, or complete the session with the buyer's details and take the card on your own page.



ON YOUR SERVER: HOLD THE TICKETS (NODE)

```
const headers = { Authorization: `Bearer ${process.env.EOS_SECRET_KEY}`, "Content-Type":
"application/json" };
const res = await fetch("https://eos.aveoevents.com/api/v1/checkout/sessions", {
  method: "POST",
  headers: { ...headers, "Idempotency-Key": crypto.randomUUID() },
  body: JSON.stringify({ event_id: 41, items: [{ ticket_type_id: 9001, quantity: 2 }] }),
});
const session = await res.json();
// hosted checkout: send the buyer to session.url
```

YOUR OWN CHECKOUT: PLACE THE ORDER

```
const done = await fetch(`https://eos.aveoevents.com/api/v1/checkout/sessions/${session.id}/complete`, {
  method: "POST",
  headers: { ...headers, "Idempotency-Key": crypto.randomUUID() },
  body: JSON.stringify({ buyer: { first_name: "Sam", last_name: "Taylor", email: "sam@example.com" },
payment_method: "card" }),
});
const { payment } = await done.json(); // client_secret, publishable_key, stripe_account
```

IN THE BROWSER: TAKE THE CARD WITH STRIPE.JS

```
const stripe = Stripe(payment.publishable_key, { stripeAccount: payment.stripe_account });
const elements = stripe.elements({ clientSecret: payment.client_secret });
elements.create("payment").mount("#payment");
// when the buyer presses Pay
await stripe.confirmPayment({ elements, confirmParams: { return_url: "https://yoursite.com/thanks" } });
```

A card order is paid, and its tickets emailed, as soon as Stripe confirms: fulfil on order.paid, not on the browser's word. Free orders, and orders of tickets paid at the door, are done at once. The card payment is the order plus AVEO's booking fee (once per order): the session's amount_total and booking_fee say how much, and your pages should show each ticket type's price_with_booking_fee from the start. Ticket types with an age limit need each guest's date_of_birth. Cancel a session you no longer need and its tickets go straight back on sale.

04 WEBHOOKS.

Hear about orders, check-ins and events the moment they happen, and trust only signed calls.

Add an endpoint (an https address) with POST /webhook-endpoints, using a key with the Manage webhook endpoints permission, and choose the events it hears; none chosen means all. The signing secret (whsec_…) is in the answer once; the API keys page lists your endpoints. AVEO EOS calls your address with a JSON body:

THE BODY

```
{
  "id": "evt_5c1d8e2f9a3b7c4d6e0f1a2b",
  "type": "order.paid",
  "api_version": "2026-11-15",
  "created_at": "2026-11-20T10:15:01.000Z",
  "company_id": 12,
  "livemode": true,
  "data": { "object": { "object": "order", "id": "O-7KQ2M4X", ... } }
}
```

data.object is the same object the API returns for it: an order for order.*, a check-in for attendee.checked_in. livemode is false for calls from your sandbox.

CHECK THE SIGNATURE WITH THE STANDARD WEBHOOKS LIBRARY (NODE, EXPRESS)

```
import express from "express";
import { Webhook } from "standardwebhooks";

const wh = new Webhook(process.env.EOS_WEBHOOK_SECRET); // whsec_...
const app = express();

app.post("/hooks/eos", express.raw({ type: "application/json" }), (req, res) => {
  let event;
  try {
    event = wh.verify(req.body.toString("utf8"), req.headers); // the raw body, as sent
  } catch {
    return res.status(400).end();
  }
  if (event.type === "order.paid") {
    // event.data.object is the order
  }
  res.status(200).end();
});
```

OR BY HAND

```
import { createHmac, timingSafeEqual } from "node:crypto";

function verify(secret, id, timestamp, body, header) {
  if (Math.abs(Date.now() / 1000 - Number(timestamp)) > 300) return false; // over five minutes old
  const key = Buffer.from(secret.replace(/^whsec_/, ""), "base64");
  const expected = createHmac("sha256", key).update(`${id}.${timestamp}.${body}`).digest("base64");
  return header.split(" ").some((part) => {
    const [version, signature] = part.split(",");
    return version === "v1" && signature?.length === expected.length &&
timingSafeEqual(Buffer.from(signature), Buffer.from(expected));
  });
}
```

- Answer with a 2xx within 10 seconds; do the work afterwards. Redirects are not followed.
- A call that fails is tried again up to 12 times, waiting from 15 seconds up to an hour between tries: about four hours in all.
- The same call can arrive twice: use webhook-id to do each one only once.
- Every delivery is logged with its status (GET …/deliveries); send any one again with POST …/deliveries/{id}/retry, and test an endpoint to receive a ping.

05 THE RULES.

Idempotency, rate limits, paging, versions, request ids and errors.

IDEMPOTENCY	Send an Idempotency-Key (1 to 255 characters) with requests that create something. The first answer is kept for 24 hours and given back for the same request, with Idempotent-Replayed: true. The same key with a different request is refused.
RATE LIMITS	Each answer carries X-RateLimit-Limit, X-RateLimit-Remaining and X-RateLimit-Reset (the Unix time the minute starts again). Over the limit, the answer is 429 with Retry-After in seconds.
PAGING	Lists take limit (1 to 100, default 25) and give back has_more and next_cursor; pass it as cursor for the next page.
VERSIONS	Every answer carries AVEO-Version (now 2026-11-15). New fields, endpoints and webhook event types are added from time to time without a new version, so write code that ignores fields it does not know.
REQUEST IDS	Every answer carries X-Request-Id; errors repeat it as request_id. Quote it when you ask AVEO for help.
BODIES	JSON objects, up to 64 KB.

AN ERROR

```
{
  "error": {
    "type": "invalid_request_error",
    "code": "unavailable",
    "message": "General admission is sold out.",
    "param": "items[0].ticket_type_id",
    "request_id": "req_...",
    "doc_url": "https://aveoevents.com/eos/api/documents/errors#unavailable"
  }
}
```

TYPE	STATUS	WHEN
invalid_request_error	400, 404, 409, 422	Something in the request is wrong: a missing or bad parameter, an unknown id, or a state the action does not allow.
authentication_error	401	No key, a key that does not exist, or one that has expired or been revoked.
permission_error	403	The key is fine but lacks the permission, or the address or web page it was used from is not allowed.

TYPE	STATUS	WHEN
idempotency_error	409	An Idempotency-Key was reused with a different request.
rate_limit_error	429	Too many requests for this key in the last minute. Wait for the time in Retry-After.
api_error	500, 502	Something went wrong on our side or in the ticket engine. Safe to retry with the same Idempotency-Key.

Every error code, with what to do about it, is in the API reference and at aveoevents.com/eos/api/documents/errors.

06 SANDBOX AND GOING LIVE.

Build against test data and Stripe test mode, then switch with a checklist.

From Professional, every company has a sandbox: a company of its own with its own data, shop and team, linked to yours. Admins open it from Sandbox in the menu. Nothing done in it reaches your live events, orders, customers, reports or payouts.

- Keys made in the sandbox start eos_sk_test_ and eos_pk_test_, and never work on your live company.
- Card payments go through Stripe in test mode, with Stripe's test cards: paid, declined and 3-D Secure.
- Webhook calls from the sandbox say livemode false; its emails have [Test] in the subject.

GOING LIVE.

01 Make live keys
In your live company, with only the permissions each use needs, an expiry and your servers' addresses.

02 Add live endpoints
Keep the new signing secrets on your server and check every call's signature.

03 Send Idempotency-Keys
On every request that creates something, so a retry never makes two.

04 Handle 429 and retries
Wait for Retry-After; retry 5xx answers with the same Idempotency-Key.

05 Trust webhooks, not browsers
Fulfil orders on order.paid, and do each webhook-id only once.

06 Watch the log
Each key's recent requests are on the API keys page; each endpoint's deliveries are one request away.

EVERY ENDPOINT.

The AVEO EOS API reference (PDF) lists every endpoint, field, object, error code and webhook event, as does aveoevents.com/eos/api.

AVEO

BUILD WITH AVEO.

Questions about the API, a webhook or going live? Write to AVEO with your request id and we will help.

ASK FOR A TRIAL	aveoevents.com/eos
DOCUMENTS AND GUIDES	documents@aveoevents.com
SUPPORT	support@aveoevents.com
API	aveoevents.com/eos/api