COMFORTPASS

DEVELOPER REFERENCE

# Partner API v2

Search, book, amend and cancel airport Meet & Greet, lounge access, transfers and sleeping pods at 900+ airports, from your own platform.

| PRODUCTION | SANDBOX |
| --- | --- |
| https://b2b-api.comfortpass.travel/partner/v2 | https://sandbox-api.comfortpass.travel/partner/v2 |
| **FORMAT** | **AUTHENTICATION** |
| REST · JSON · HTTPS | AUTH-TOKEN header |

Version 2 · comfortpass.travel

# Contents


Coverage

Getting access

Auth

GET /airports

GET /services

GET /prices

Account balance

POST /bookings

GET /bookings/:id

POST /bookings/:id/checkout

POST /bookings/:id/amend

POST /bookings/:id/cancel

GET /bookings/:id/voucher

Webhooks

GET /baggage/catalog

GET /baggage/quote

Base URL: `https://b2b-api.comfortpass.travel/partner/v2`

## Coverage

One integration covers every service below. Figures are live airport counts (November 2026):

| SERVICE | AIRPORTS |
|---|---|
| Meet & Greet (meet and assist: arrival, departure, transit) | 887 |
| Lounge access | 243 |
| Private airport transfers | 224 |
| Sleeping pods | 47 |
| Fast Track (self-service priority security / passport, QR voucher or reservation) | 65 |

Baggage Delivery (door to door) is offered by city rather than by airport — see `GET /baggage/catalog`. The full airport list with public "from" prices is published at https://b2b.comfortpass.travel/coverage (also as `/coverage.json` and `/coverage.md`).

Covered airports include, among others: Jeddah (JED), Riyadh (RUH), Madinah (MED), Dammam (DMM), Dubai (DXB), Doha (DOH), Abu Dhabi (AUH), Cairo (CAI), Istanbul (IST), London Heathrow (LHR), Paris (CDG), Frankfurt (FRA), Amsterdam (AMS), Madrid (MAD), Rome (FCO), Singapore (SIN), Bangkok (BKK), Kuala Lumpur (KUL), Hong Kong (HKG), Delhi (DEL), Mumbai (BOM), New York (JFK), Los Angeles (LAX), Chicago (ORD), Miami (MIA) and Washington (IAD). `GET /airports` and `GET /services` return live availability and prices for any airport.

## Getting access

There are three access levels. Each one is a key you send in the `AUTH-TOKEN` header.

### 1. Demo — try it now, no account

A shared, read-only key is published here so you can call the catalogue immediately:

```
EXAMPLE
AUTH-TOKEN: cpb2b_demo_comfortpass_public
Base URL:   https://sandbox-api.comfortpass.travel/partner/v2
```

The demo key only answers the four read endpoints — `GET /airports`, `GET /services`, `GET /prices` and `GET /baggage/catalog` — on the sandbox, and is rate-limited per caller. It cannot create bookings, read account data or touch production. Prices come back at the public retail price (no partner markup, since there is no partner behind it). Anything else returns `403 API_KEY_DEMO_READ_ONLY`.

Every demo response carries an `X-ComfortPass-Demo` header, and each successful JSON response an extra `_demo` object, both pointing to https://b2b.comfortpass.travel/register — so it is always clear you are on the public demo and how to get your own sandbox key.

### 2. Sandbox — your own free test account

1. **Create your account** at https://b2b.comfortpass.travel/register and confirm your email.
2. Open *API Access* in the partner dashboard: your **sandbox key** (`cpb2b_test_…`) and the sandbox base URL are there straight away.

The sandbox is the same API with the same catalogue and your own prices (your markup included), running on Stripe test mode: pay with card `4242 4242 4242 4242` (any future expiry, any CVC) and no real money moves. `traveler_link`, `balance`, `embedded` and `native` all return real Stripe test-mode responses. You can create bookings, amend and cancel them, and watch statuses and webhooks — all against test data that never reaches our operations team.

### 3. Production — after a short review

1. **Request production access** from the same *API Access* page once your integration works end to end.
2. **We approve it** and your **production key** (`cpb2b_live_…`) appears in the dashboard. You get an email when it is active.

Until step 4 the production API answers `403 API_KEY_PRODUCTION_NOT_APPROVED`. Production bookings are real: they are fulfilled and charged.

Sandbox and production are separate: different keys, different data. A sandbox key is rejected by production and the other way round, and bookings made in one never appear in the other.

### Base URLs

| ENVIRONMENT | BASE URL |
|---|---|
| Production | `https://b2b-api.comfortpass.travel/partner/v2` |
| Sandbox (and the demo key) | `https://sandbox-api.comfortpass.travel/partner/v2` |

---

## Auth

Every request needs your partner API key in the `AUTH-TOKEN` header:

EXAMPLE

```
AUTH-TOKEN: cpb2b_live_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Your sandbox and production keys are shown in the ComfortPass Partner Dashboard → API Access. Keep them private. The demo key above is public on purpose.

Auth errors:

| HTTP | CODE | MEANING |
|---|---|---|
| 401 | `API_KEY_MISSING` | No `AUTH-TOKEN` header sent |
| 403 | `API_KEY_INVALID` | Key doesn't match any account (also: a sandbox key sent to production, or the reverse) |

| HTTP | CODE | MEANING |
|---|---|---|
| 403 | `API_KEY_PRODUCTION_NOT_APPROVED` | Production access hasn't been approved for this account yet — use the sandbox |
| 403 | `API_KEY_DEMO_READ_ONLY` | The demo key was used for anything other than the four read endpoints — create a free account for a sandbox key |
| 403 | `API_KEY_DEMO_SANDBOX_ONLY` | The demo key was sent to production — it works on the sandbox base URL only |
| 403 | `API_KEY_INACTIVE` | Key exists but has been revoked |
| 403 | `API_KEY_EXPIRED` | Key had an expiry date and it has passed |
| 429 | `RATE_LIMITED` | Rate limit reached — back off and retry (see below) |

### Rate limits

| KEY | LIMIT |
|---|---|
| Production | 60 requests/minute per key |
| Sandbox | 60 requests/minute, plus 2,000 requests/day and 50 bookings/day per key (daily counters reset 00:00 UTC) |
| Demo | 30 requests/minute and 500 requests/day per caller address |

A `429` carries a `Retry-After` header (seconds). Repeatedly sending an unrecognized key from the same address is throttled separately — a `429 RATE_LIMITED` after ~20 bad keys in 10 minutes.

---

## GET /airports

Search airports by name, city, or IATA code. Returns up to 50 active airports.

### Query params

| PARAM | REQUIRED | NOTES |
|---|---|---|
| `q` | no | Free-text match against name/city/code. Omit to list all active airports (capped at 50). |

### Example

```
REQUEST
GET /partner/v2/airports?q=london
AUTH-TOKEN: cpb2b_live_...
```

RESPONSE


```
{
  "airports": [
    {
      "code": "LHR",
      "name": "London Heathrow Airport (LHR)",
      "city": "London",
      "country": "United Kingdom",
      "slug": "london-heathrow-lhr",
      "servicesMeetgreet": true,
      "servicesLounge": true,
      "servicesTransfer": true,
      "servicesSleepingpods": true
    }
  ]
}
```


The 4 `services*` flags tell you which sections of `/services` will actually return packages for this airport — don't bother calling `/services` for a type flagged `false`. Baggage Delivery isn't airport-scoped, see `/baggage/catalog` below instead.

---

## GET /services

Priced packages for one airport across all airport-scoped service types, with your account's markup already added. The response also carries a `fastlane` array (0 or 1 entry) for airports where Fast Track is offered — its `code` is `fastlane-<IATA>`, booked the same way as the others (see `POST /bookings` → Fast Track).

### Query params

| PARAM | REQUIRED | NOTES |
|---|---|---|
| `airport` | yes | IATA code, e.g. `LHR`. 404s if unknown or inactive. |

### Example

REQUEST

```
GET /partner/v2/services?airport=LHR
AUTH-TOKEN: cpb2b_live_...
```

RESPONSE


```
{
  "airport": { "code": "LHR", "name": "London Heathrow Airport (LHR)" },
  "currency": "USD",
  "services": {
    "meetgreet": [
      {
        "code": "mng-1399",
        "title": "Arrival",
        "inclusions": "Greeter monitors the real-time status of your flight, ...",
        "price": 345
      }
    ],
    "lounge": [
      {
        "code": "lounge-3036",
        "title": "The Lounge",
        "terminal": "Terminal 5",
        "maxHours": 6,
        "imageUrl": "https://comfortpass.travel/cdn/images/lounges/260.webp",
        "price": 180
      }
    ],
    "transfer": [
      {
        "code": "transfer-84",
        "title": "Sedan to London",
        "destinationCity": "London",
        "carType": "Sedan",
        "maxPassengers": 3,
        "maxLuggage": null,
        "price": 460
      }
    ],
    "sleepingpods": [
      {
        "code": "sleepingpods-60",
        "title": "Twin Room",
        "terminal": "terminal 2 and 3",
        "hoursOfService": 12,
        "extraHourPrice": null,
        "imageUrl": "https://comfortpass.travel/cdn/images/pods/60.webp",
        "gallery": ["https://comfortpass.travel/cdn/images/pods/60-1.webp"],
        "price": 515
      }
    ]
  }
}
```


`imageUrl` / `gallery` can be `null` / `[]` — not every package has a photo yet. All image URLs are our own CDN, safe to hotlink directly in your UI (no auth needed, no expiry).

`transfer[].price` is **one-way** — a `Return` booking is exactly 2× this (see `POST /bookings`). `sleepingpods[].extraHourPrice` is `null` when that pod has no hourly-extension option; a non-null value is the price per extra hour on top of `price`.

Errors: `VALIDATION_MISSING_AIRPORT` (400, no `airport` param), `AIRPORT_NOT_FOUND` (404).

### GET /prices

Re-confirm a single package's price by its `code` (from `/services`).

#### Query params

| PARAM | REQUIRED | NOTES |
|---|---|---|
| `code` | yes | e.g. `mng-1399`, `lounge-3036`, `transfer-84`, `sleepingpods-60` — exactly as returned by `/services`. |

#### Example

```
REQUEST
GET /partner/v2/prices?code=mng-1399
AUTH-TOKEN: cpb2b_live_...
```

```
RESPONSE
{ "code": "mng-1399", "title": "Arrival", "currency": "USD", "price": 345 }
```

Errors: `VALIDATION_MISSING_CODE` (400, no `code` param), `VALIDATION_INVALID_CODE` (400, unrecognized code prefix), `PRICE_NOT_FOUND` (404, no active package with that id).

## Account balance

`POST /bookings` supports four payment models, chosen per-request with `paymentMethod`:

- `traveler_link` (default if you omit the field) — you get back a `checkoutUrl`; redirect the traveler there to pay us directly. Zero setup, but the traveler ends up on our Stripe-hosted checkout page, not yours. Good for a quick start or a proof of concept.
- `balance` — you pay us directly from a prepaid balance you top up in advance. The traveler never sees ComfortPass at all: you build your own checkout, collect payment from your own customer however you like, and settle with us in the background. This is the mode to use once you're building a real, branded site — not just testing the catalog.
- `embedded` — we create a Stripe Checkout Session in `ui_mode: 'embedded'` and return its `clientSecret` + `publishableKey`. Mount it yourself with Stripe.js's `initEmbeddedCheckout({ clientSecret })` on your own web page — no ComfortPass-hosted page at all, but still Stripe's own checkout UI embedded in yours. ComfortPass stays merchant of record. Web only — a Checkout Session's `client_secret` does not work with Stripe's native mobile SDKs, use `native` for that.
- `native` — we create a raw Stripe PaymentIntent (not a Checkout Session) and return its `clientSecret` + `publishableKey`, for use with Stripe's native iOS/Android SDK (PaymentSheet / `ApplePayContext`). This is the only mode that supports a true native Apple Pay sheet inside your own app. ComfortPass stays merchant of record; see "Apple Pay setup" below for how your app's Apple Merchant ID pairs with our Stripe account.

`embedded` and `native` don't currently support RUSH/short-notice bookings (see `RUSH_REQUIRES_BALANCE_OR_LINK` below) — only standard bookings for now.

Fund your balance with `POST /account/topup`, check it with `GET /account/balance`.

### GET /account/balance

```
REQUEST
GET /partner/v2/account/balance
AUTH-TOKEN: cpb2b_live_...
```

```
RESPONSE
{ "currency": "USD", "balanceUsd": 0, "creditLimitUsd": 0, "availableUsd": 0 }
```

`availableUsd` = `balanceUsd + creditLimitUsd` — this is the number `POST /bookings` checks against when you use `paymentMethod: 'balance'`. `creditLimitUsd` is `0` by default (prepaid only); we can grant a credit line for trusted accounts on request.

### POST /account/topup

Creates a Stripe Checkout session that adds funds to your balance. Not tied to any booking.

```
REQUEST
POST /partner/v2/account/topup
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{ "amountUsd": 500 }
```

```
RESPONSE
{ "checkoutUrl": "https://checkout.stripe.com/c/pay/cs_..." }
```

Open `checkoutUrl` and pay — your balance updates once the payment clears (usually within seconds). `successUrl` / `cancelUrl` are optional overrides; default to your dashboard.

Errors: `VALIDATION_INVALID_AMOUNT` (400, `amountUsd` missing or not a positive number), `PARTNER_ACCOUNT_INVALID` (403, account suspended).

---

## POST /bookings

Create a real booking. Once paid, it's fulfilled exactly like a normal comfortpass.travel order: the service is booked with the local provider, the voucher becomes available, your commission is recorded (for `traveler_link` bookings — `balance` bookings pay the base price, no separate commission since you set your own margin before charging your customer). Body shape depends on `serviceCode`.

### Meet & Greet / Lounge / Transfer / Sleeping Pods

| FIELD | REQUIRED | NOTES |
|---|---|---|
| `airportCode` | yes | IATA code, must match the airport you looked the package up under. |
| `serviceCode` | yes | From `/services`. |
| `paymentMethod` | no | `traveler_link` (default), `balance`, `embedded`, or `native` — see "Account balance" above. |

| FIELD | REQUIRED | NOTES |
|---|---|---|
| `date` | yes | `YYYY-MM-DD`. |
| `passengers` | yes | Array of `{ firstName, lastName, type }`, `type` is `adult` \| `child` \| `infant`. At least one. |
| `contactEmail` | yes | |
| `contactPhone` | no | |
| `direction` | M&G/Lounge only | `Arrival` or `Departure`. Ignored for transit M&G packages. |
| `flightNumber` / `flightTime` | no | |
| `connectingFlight` | Transit M&G only | |
| `specialRequests` | no | |
| `locale` | no | Defaults to `en`. |
| `tripType` | Transfer only | `Oneway` (default) or `Return` — doubles `totalAmount`. |
| `exactDropOffLocation` | **Transfer, required** | Free-text address. |
| `returnDate` / `returnTime` | Transfer, if `tripType: "Return"` | |
| `commentsForDriver` | Transfer only | |
| `additionalHours` | Sleeping Pods only | Only has effect if `/services` showed `extraHourPrice` for that pod. |

### Example

```
REQUEST
POST /partner/v2/bookings
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{
  "airportCode": "LHR",
  "serviceCode": "mng-1399",
  "direction": "Arrival",
  "date": "2026-08-15",
  "flightNumber": "BA123",
  "flightTime": "14:30",
  "passengers": [{ "firstName": "John", "lastName": "Smith", "type": "adult" }],
  "contactEmail": "traveler@example.com",
  "contactPhone": "+447700900000"
}
```

RESPONSE

```
{
  "bookingId": "01KY1Z2XEXEXZAAQ372FTD62RH",
  "bookingNumber": 10796,
  "status": "awaiting_payment",
  "totalAmount": 345,
  "currency": "USD",
  "checkoutUrl": "https://checkout.stripe.com/c/pay/cs_..."
}
```

With `"paymentMethod": "balance"` added to the same request, the response is already paid — `checkoutUrl: null` is success here, not a failure to create a session:

RESPONSE

```
{
  "bookingId": "01KY5J496CT5N9YCA66VQ2MBXF",
  "bookingNumber": 10806,
  "status": "payment_confirmed",
  "totalAmount": 108,
  "currency": "USD",
  "checkoutUrl": null
}
```

With `"paymentMethod": "native"` (or `"embedded"`), the response carries a `clientSecret` instead of a `checkoutUrl` — `checkoutUrl` is `null`, that's expected, not a failure:

RESPONSE

```
{
  "bookingId": "01KY5J496CT5N9YCA66VQ2MBXG",
  "bookingNumber": 10807,
  "status": "awaiting_payment",
  "totalAmount": 345,
  "currency": "USD",
  "checkoutUrl": null,
  "clientSecret": "pi_3Abc..._secret_XYZ",
  "publishableKey": "pk_live_..."
}
```

For `native`, pass `clientSecret` to `STPApplePayContext` / `PaymentSheet` (iOS) or the equivalent Android SDK call, initialized with `publishableKey` — not your own Stripe account's key, ComfortPass's. For `embedded`, pass `clientSecret` to Stripe.js's `initEmbeddedCheckout({ clientSecret })` on your own page.

Errors: `VALIDATION_MISSING_FIELDS` (400), `VALIDATION_INVALID_CODE` (400), `AIRPORT_NOT_FOUND` (404), `PACKAGE_NOT_FOUND` (404), `PARTNER_ACCOUNT_INVALID` (403), `INSUFFICIENT_BALANCE` (402, `paymentMethod: 'balance'` but your `availableUsd` can't cover `totalAmount` — nothing was created, top up and retry), `RUSH_REQUIRES_BALANCE_OR_LINK` (400, a RUSH/short-notice booking was attempted with `paymentMethod: 'embedded'` or `'native'` — use `balance` or omit `paymentMethod` for those instead).

### Apple Pay setup (for `native`)

Apple Pay inside your own app needs your app's Apple Merchant ID paired with the Stripe account that actually processes the payment — ours, since ComfortPass is merchant of record. Both sides do one step

each:

1. **You** register an Apple Merchant ID (e.g. `merchant.app.yourapp`) under your own Apple Developer Program account — this stays yours, not ours.
2. **We** go to Stripe Dashboard → Settings → iOS Certificate Settings → "Add new application" and download a CSR (Certificate Signing Request) file, and send it to you.
3. **You** use that CSR to create an Apple Pay certificate on the Apple Developer portal, under your Merchant ID, and send the resulting certificate back to us.
4. **We** upload it to the same Stripe iOS Certificate Settings entry to complete pairing.
5. **You** initialize Apple Pay in your app with your own `merchantIdentifier` from step 1, combined with the `publishableKey` from the `POST /bookings` response (ours, not yours) — Apple Pay then settles through our Stripe account automatically.

(Stripe's own native iOS Apple Pay guide covers steps 1-4 in more detail if useful.)

### Baggage Delivery

No `airportCode` — country/city instead. Price is computed **live** at booking time (the same call `/baggage/quote` makes internally) — don't pass a price, and don't assume the price you saw from an earlier `/baggage/quote` call is still exact.

| FIELD | REQUIRED | NOTES |
|---|---|---|
| `serviceCode` | yes | `baggagedelivery-<cityId>`, e.g. from `/baggage/catalog`. |
| `countryId` | yes | From `/baggage/catalog`. |
| `serviceTypeId` / `deliveryTypeId` | yes | From `/baggage/catalog`. |
| `bags` | no | Defaults to 1. |
| `date` | yes | `YYYY-MM-DD`. |
| `passengers` / `contactEmail` / `contactPhone` | same as above | |
| `pickupAddress` | yes | |
| `dropoffAddress` | yes | |
| `pickupName` / `dropoffName` | no | e.g. hotel name. |
| `reservationName` / `reservationNumber` | no | Hotel reservation the bags are tied to. |
| `terminalDetails` | no | When an airport leg is involved. |
| `flightNumber` | no | |

```
REQUEST
POST /partner/v2/bookings
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{
  "serviceCode": "baggagedelivery-11",
  "countryId": 16,
  "serviceTypeId": 208,
  "deliveryTypeId": 1,
  "bags": 2,
  "date": "2026-08-15",
  "pickupAddress": "Sydney Airport Terminal 1, Sydney",
  "dropoffAddress": "488 George St, Sydney NSW 2000",
  "reservationName": "John Smith",
  "reservationNumber": "RES12345",
  "passengers": [{ "firstName": "John", "lastName": "Smith", "type": "adult" }],
  "contactEmail": "traveler@example.com"
}
```

Response shape is identical to the airport-scoped case. Extra error: `QUOTE_FAILED` (502, the live price lookup failed — retry).

### Fast Track

Airport-scoped by IATA, not a numeric package id, so the `serviceCode` is `fastlane-<IATA>` (take it from the `fastlane` array of `GET /services`). Fast Track is self-service — a QR voucher or a name-bound reservation, no host or escort — and is priced per passenger. `paymentMethod` must be `traveler_link` (default) or `balance`; `embedded` and `native` are not supported for Fast Track.

| FIELD | REQUIRED | NOTES |
|---|---|---|
| `serviceCode` | yes | `fastlane-<IATA>`, e.g. `fastlane-NCE`. |
| `paymentMethod` | no | `traveler_link` (default) or `balance`. |
| `date` | yes | `YYYY-MM-DD` (flight date). |
| `passengers` | yes | One voucher/reservation per passenger. |
| `contactEmail` | yes | Where the voucher is emailed. |
| `direction` | no | `Arrival` or `Departure`. |
| `flightNumber` / `flightTime` / `airline` / `destinationIata` / `terminal` | depends on airport | Each airport requires a specific subset — `GET /services` indicates what Fast Track needs, and the API returns `VALIDATION_MISSING_FIELDS` listing any that are missing. |

```
REQUEST
POST /partner/v2/bookings
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{
  "serviceCode": "fastlane-NCE",
  "date": "2026-08-15",
  "airline": "AF",
  "destinationIata": "CDG",
  "terminal": "T2",
  "passengers": [{ "firstName": "John", "lastName": "Smith", "type": "adult" }],
  "contactEmail": "traveler@example.com"
}
```

Response shape is identical to the airport-scoped case. **Availability:** live in the sandbox; production roll-out in progress — on production, `fastlane-…` codes return `VALIDATION_INVALID_CODE` until it is switched on.

---

## GET /bookings/:id

Status of a booking you created. Only returns bookings created by your own API key — a booking id that belongs to a different partner (or doesn't exist) returns the same `BOOKING_NOT_FOUND` either way.

```
REQUEST
GET /partner/v2/bookings/01KY1Z2XEXEXZAAQ372FTD62RH
AUTH-TOKEN: cpb2b_live_...
```

```
RESPONSE
{
  "bookingId": "01KY1Z2XEXEXZAAQ372FTD62RH",
  "bookingNumber": 10796,
  "status": "awaiting_payment",
  "totalAmount": 345,
  "currency": "USD",
  "confirmationCode": null
}
```

`status` is one of: `awaiting_payment`, `payment_confirmed`, `awaiting_info`, `processing`, `confirmed`, `cancelled`, `failed`, `refunded`. `confirmationCode` is set once the booking reaches `confirmed`.

---

## POST /bookings/:id/checkout

Get a fresh payment link for a `traveler_link` booking whose original `checkoutUrl` was abandoned or expired (Stripe's default session expiry is 24h). Only valid while `status` is still `awaiting_payment` — use this instead of creating a new booking, so the traveler doesn't have to re-enter passenger/trip details.

**Body** (all optional)

| FIELD | NOTES |
|---|---|
| `locale` | Defaults to the locale the booking was created with. |

### Example

```
REQUEST
POST /partner/v2/bookings/01KY1Z2XEXEXZAAQ372FTD62RH/checkout
AUTH-TOKEN: cpb2b_live_...
```

```
RESPONSE
{ "checkoutUrl": "https://checkout.stripe.com/c/pay/cs_..." }
```

Errors: `BOOKING_NOT_FOUND` (404, not yours or doesn't exist), `BOOKING_NOT_RESUMABLE` (409, `status` has moved past `awaiting_payment` — already paid, cancelled, etc.), `balance` bookings never reach `awaiting_payment` so this endpoint doesn't apply to them.

---

## POST /bookings/:id/amend

Request a change to a booking you created: date, flight, passengers, contact details or special requests. Nothing is applied automatically — a change can affect availability and price, so our 24/7 concierge team confirms it with the local service provider, applies it, and comes back to you if the price changes. The booking's `status` is unaffected; poll `GET /bookings/:id` as usual.

### Body

| FIELD | REQUIRED | NOTES |
|---|---|---|
| `changes` | one of `changes` / `note` | Object with any of: `date` ( `YYYY-MM-DD` ), `flightNumber` , `flightTime` ( `HH:MM` ), `connectingFlight` , `direction` ( `Arrival` \| `Departure` ), `passengers` (full replacement list, same shape as `POST /bookings` ), `contactEmail` , `contactPhone` , `specialRequests` . |
| `note` | one of `changes` / `note` | Free text for anything the fields above don't cover. |

### Example

```
REQUEST
POST /partner/v2/bookings/01KY1Z2XEXEXZAAQ372FTD62RH/amend
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{ "changes": { "date": "2026-08-16", "flightNumber": "SV117", "flightTime": "09:40" }, "note":
"Flight rescheduled by the airline" }
```

```
RESPONSE
{ "status": "amendment_requested", "requestId": 48211, "hoursUntilService": 212.5 }
```

Returns `202` . Errors: `BOOKING_NOT_FOUND` (404), `VALIDATION_INVALID_CHANGES` (400, a field is malformed), `VALIDATION_NO_CHANGES` (400, neither `changes` nor `note` given), `AMENDMENT_NOT_ACTIONABLE` (409, booking already cancelled / refunded / failed).

## POST /bookings/:id/cancel

Request cancellation of a booking you created. Free cancellation applies up to 48 hours before the service (72 hours for airports in Australia).

**Body** (optional): `{ "reason": "..." }`

```
REQUEST
POST /partner/v2/bookings/01KY1Z2XEXEXZAAQ372FTD62RH/cancel
AUTH-TOKEN: cpb2b_live_...
```

- Paid from your **balance**: refunded to your balance immediately — `{ "status": "refunded", "refundedToBalance": 345 }`.
- Paid by **card** ( `traveler_link` / `embedded` / `native` ): `{ "status": "cancellation_requested" }` ; the refund to the original card is processed by our team and the booking then moves to `refunded` .

Errors: `BOOKING_NOT_FOUND` (404), `OUTSIDE_CANCELLATION_WINDOW` (409, includes `hoursUntilService` ), `RUSH_NOT_CANCELLABLE_VIA_API` (400, short-notice bookings are non-refundable once confirmed — contact us), `CANCELLATION_NOT_ACTIONABLE` (409, already cancelled / refunded / failed).

## GET /bookings/:id/voucher

Once a booking reaches `confirmed` , get a download link for the traveler's voucher plus a QR code encoding that same link — hand either to your customer.

```
REQUEST
GET /partner/v2/bookings/01KY1Z2XEXEXZAAQ372FTD62RH/voucher
AUTH-TOKEN: cpb2b_live_...
```

Before the booking is confirmed:

```
RESPONSE
{ "status": "pending", "message": "Voucher is being generated. Please try again later." }
```

Once confirmed:

```
RESPONSE
{
  "status": "ready",
  "voucherUrl": "https://comfortpass.travel/api/voucher/01KY1Z2XEXEXZAAQ372FTD62RH",
  "qrCodeDataUrl": "..."
}
```

`qrCodeDataUrl` is a ready-to-render `<img src="...">` — no separate QR library needed on your end. Same tenant isolation as `GET /bookings/:id` : a booking id that isn't yours returns `BOOKING_NOT_FOUND` (404).

## Webhooks

One event today: `booking.status_changed`, fired whenever a booking you created moves to a new status (see the `status` values under `GET /bookings/:id`). More event types may be added later — always check `event` in the payload rather than assuming it's the only one you'll ever receive.

Delivery is polled, not instant — a background dispatcher checks for new events roughly every 2 minutes, so expect a small delay rather than sub-second push.

### Register an endpoint — `POST /account/webhooks`

```
REQUEST
POST /partner/v2/account/webhooks
AUTH-TOKEN: cpb2b_live_...
Content-Type: application/json

{ "url": "https://your-server.example.com/webhooks/comfortpass" }
```

```
RESPONSE
{
  "id": "01M0X7ENSDR102ZVZHT7DM3BH5",
  "url": "https://your-server.example.com/webhooks/comfortpass",
  "secret": "whsec_xxxxxxxx",
  "status": "active"
}
```

`url` must be `https://`. `secret` **is returned exactly once, right here** — store it immediately. It's never re-displayed; if you lose it, delete the endpoint and register a new one. Registering only affects events from that point forward — you will not be backfilled with historical status changes for bookings created before the endpoint existed.

Errors: `VALIDATION_INVALID_URL` (400, missing or non-`https://` url).

### List your endpoints — `GET /account/webhooks`

```
RESPONSE
{ "webhooks": [{ "id": "01M0X7EN...", "url": "https://...", "status": "active", "createdAt": "2026-08-25 19:47:12" }] }
```

Never returns `secret` — only visible once, at creation.

### Remove an endpoint — `DELETE /account/webhooks/:id`

```
RESPONSE
{ "deleted": true }
```

`WEBHOOK_NOT_FOUND` (404) if the id isn't yours or doesn't exist.

### Payload shape

RESPONSE

```
{
  "event": "booking.status_changed",
  "bookingId": "01KY1Z2XEXEXZAAQ372FTD62RH",
  "bookingNumber": 10796,
  "status": "confirmed",
  "occurredAt": "2026-08-25 19:48:12"
}
```

### Verifying the signature

Every delivery carries an `X-ComfortPass-Signature` header: `sha256=<hex>`, an HMAC-SHA256 of the exact raw request body using your endpoint's `secret`. Recompute it yourself and compare — don't trust the payload otherwise:

EXAMPLE

```
const crypto = require('crypto');
const expected = 'sha256=' + crypto.createHmac('sha256', secret).update(rawBody).digest('hex');
if (expected !== req.headers['x-comfortpass-signature']) return res.status(401).end();
```

Use the **raw, unparsed** request body for this — re-serializing a parsed JSON object can produce a byte-for-byte different string (key order, whitespace) and break the comparison.

### Retry policy & idempotency

A delivery attempt is either a 2xx response (success) or anything else (network error, timeout, non-2xx) counts as a failure. Failed deliveries retry roughly every 2 minutes (the same dispatcher tick) for up to **8 attempts total**, then stop. There's no exponential backoff — each tick is just another attempt.

Idempotency: each `(endpoint, status change)` pair is delivered at most once per successful attempt — we never re-send an event you've already 2xx'd. If your endpoint is briefly down and misses an attempt, expect the *next* tick to retry the *same* event, not a fresh one. Make your handler idempotent regardless (e.g. key on `bookingId` + `status`) in case a delivery succeeds on your end but the response is lost in transit before we see the 2xx.

---

## GET /baggage/catalog

Countries/cities and the service/delivery type option lists for Baggage Delivery. Static data — cache it, no need to call this per traveler.

REQUEST

```
GET /partner/v2/baggage/catalog
AUTH-TOKEN: cpb2b_live_...
```

RESPONSE

```
{
  "countries": [
    { "id": 16, "name": "Australia", "iso": "AU", "cities": [{ "id": 11, "name": "Sydney" }] }
  ],
  "serviceTypes": [{ "id": "208", "name": "From Airport to Hotel" }],
  "deliveryTypes": [{ "id": "1", "name": "Standard" }]
}
```

---

## GET /baggage/quote

Live price for one Baggage Delivery route. Call this whenever the traveler changes city/type/bags — price is not baked into the catalog.

**Query params**: `countryId`, `cityId`, `serviceTypeId`, `deliveryTypeId` (all required), `bags` (optional, default 1).

REQUEST

```
GET /partner/v2/baggage/quote?countryId=16&cityId=11&serviceTypeId=208&deliveryTypeId=1&bags=2
AUTH-TOKEN: cpb2b_live_...
```

RESPONSE

```
{ "code": "baggagedelivery-11", "currency": "USD", "bags": 2, "price": 225 }
```

Errors: `VALIDATION_MISSING_FIELDS` (400), `QUOTE_FAILED` (502, live price lookup failed — retry).