Stealth Health — Clinical Partner API Integration Guide

Version: 0.1 (Draft) **Date:** March 2, 2026 **Base URL:** `https://api.stealth.health` (production) | `https://sandbox.stealth.health` (development) **Status:** Proposal / Scoping

> ***See also:*** *Partner API Integration Guide (Referral Tier) — a lighter integration where no PHI is shared with the partner.*

Table of Contents

1. Overview
2. How This Differs from the Referral Integration
3. HIPAA & Compliance Requirements
4. Authentication & Security
5. Integration Flow
    - 5.3 Alternate Flow: Partner-Submitted Prescriptions
    - 5.4 Alternate Flow: Partner-Submitted Intake
6. API Reference — Patients
7. API Reference — Appointments
8. API Reference — Intake Responses
9. API Reference — Messages
    - 9.1 GET /partner/patients/:patient_id/messages
    - 9.2 Partner-submit policy & patient-authored messages
    - 9.3 Embeddable messaging module
10. API Reference — Transactions
11. API Reference — Referrals & Products
    - 11.1 Partner-Submitted Prescriptions (alternate flow)
    - 11.2 Controlled Substances by Jurisdiction
12. API Reference — Lab Orders & Requisitions
    - 12.1 GET /partner/patients/:patient_id/lab-orders
    - 12.2 GET /partner/lab-orders/:lab_order_id
    - 12.3 GET /partner/lab-orders/:lab_order_id/requisition
    - 12.4 GET /partner/lab-orders/:lab_order_id/results
    - 12.5 Lab order status lifecycle
    - 12A. API Reference — Patient Documents
    - 12B. API Reference — Wearables Data
13. Webhook Events

14. Data Models
15. Error Handling
16. Rate Limits & Environments
17. Partner Implementation Best Practices
    - 17.1 Credential Storage & Rotation
    - 17.2 Webhook Signature Verification
    - 17.3 Idempotency & Retry Handling
    - 17.4 Minimum Necessary & Local Redaction
    - 17.5 Logging, Monitoring & SIEM
    - 17.6 Sandbox-to-Production Cutover
18. Appendix: Status Lifecycle

---

1. Overview

This guide describes the **Clinical Partner** integration tier. It is designed for telemedicine platforms that operate as an extension of the Stealth Health clinical workflow and need visibility into patient data, questionnaire responses, appointment status, and transaction history.

Use case: A partner telemedicine site uses Stealth Health's enrollment pages and physician network to evaluate and prescribe for their customers. The partner needs to:

- Know which of their customers have completed enrollment.
- View the patient's account information (name, contact, demographics).
- Access the full intake questionnaire responses.
- Track appointment status (pending review, approved, denied, fulfillment).
- View transaction and payment history for each patient.

Because this integration exposes **protected health information (PHI)**, it carries additional compliance, legal, and technical requirements compared to the Referral Tier.

---

2. How This Differs from the Referral Integration

	Referral Tier	Clinical Partner Tier
PHI exposure	None — de-identified references only	Yes — patient identity, intake responses, appointment details
BAA type	Referral source (limited)	Full Business Associate Agreement
Patient data access	Status + timestamps only	Full patient profile, intake, appointment, transactions
Questionnaire responses	Not available	Full intake Q&A
Prescription details	Not available	Medication, dosage, quantity, prescriber
Transaction data	Amount + status only	Full payment breakdown, line items, Stripe references
Audit requirements	Standard API logging	PHI access audit trail with 6-year retention
Encryption	TLS in transit	TLS in transit + AES-256 at rest on partner side (required)
Compliance review	Self-attestation	Stealth Health compliance review + annual re-certification

3. HIPAA & Compliance Requirements

This section describes both **what Stealth Health does on the platform side** to satisfy the HIPAA Security and Privacy Rules and **what the partner is contractually responsible for** as a downstream Business Associate.

3.1 Business Associate Agreement (BAA)

A **full BAA** is required before credentials are issued. The BAA designates the partner as a **Business Associate** with the obligations enumerated in § 3.4 and explicitly covers:

- Permitted uses and disclosures of PHI received via the API.
- Mandatory safeguards (administrative, physical, technical).
- Subcontractor flow-down requirements.
- Breach reporting timelines (24 hours from suspected breach).
- Return / destruction of PHI on contract termination.
- Annual re-attestation of safeguards (see § 3.6).

3.2 What Stealth Health Does to Protect PHI

The platform implements the following controls. Partners can reference this list directly in their own HIPAA risk assessment and security questionnaires.

Transport security

- TLS 1.2+ enforced on every public endpoint (`api.stealth.health`, `sandbox.stealth.health`, and webhook receivers). HSTS is set with a one-year max-age.
- Certificate management is handled by Google Cloud Load Balancer; certs are rotated automatically.
- HTTP traffic is redirected to HTTPS at the edge; non-TLS requests are never accepted.

Storage security

- All PHI is stored on **Google Cloud** infrastructure in the `us-east5` region (Columbus, Ohio), with AES-256 encryption at rest using Google-managed keys (FIPS 140-2 validated).
- Backups (daily point-in-time recovery + on-demand exports) are retained encrypted with the same KMS protections.
- Pharmacy / fulfillment files (PDF prescriptions, lab requisitions) are stored in object storage with uniform bucket-level access and signed-URL distribution; raw object URLs are never exposed.

Authentication & key handling

- API keys are issued as `sk_live_<64-hex>` / `sk_test_<64-hex>` and **never stored in plaintext**. Only a SHA-256 hash of the key is persisted.
- Key comparison is constant-time so the auth path is not susceptible to timing oracles.
- Key rotation is supported with a **dual-hash window**: the previous key is honored until `previous_key_expires_at`, allowing zero-downtime rotation.
- Tier enforcement: clinical-only endpoints reject referral-tier keys with `403 CLINICAL_ACCESS_REQUIRED` before any handler logic runs.

Webhook integrity

- Every outbound event is signed with **HMAC-SHA256** using the partner's `webhook_secret`. The signature is sent in the `X-Stealth-Signature: sha256=<hex>` header.
- Partners MUST verify this header on every delivery (see § 17.2). The platform does not retry into endpoints that do not respond `2xx` — failed events transition through `pending → pending_retry` with backoff `[30s, 5m, 30m, 2h, 12h]` (max 6 attempts) and are persisted in our internal event store for replay.
- Each event carries a unique `event_id` (`evt_<24-hex>`); partners SHOULD treat this as the idempotency key.

Rate limiting & abuse protection

- Per-partner sliding-window rate limit: **300 req/min in production**, **60 req/min in sandbox**. Limit-exhausted requests return `429 RATE_LIMIT_EXCEEDED` with `Retry-After`.
- Optional IP allowlisting available on request (configured on your partner account).

Audit trail (server-side)

Every PHI-bearing read or mutation produced by the platform is recorded in an immutable, append-only audit log. The versioned audit schema captures:

Field	Purpose
`action`	e.g. `appointment.created`, `prescription.signed`, `phi.viewed`
`category`	`appointment` / `order` / `prescription` / `lab` / `message` / `patient_profile` / `phi_view` / `payment` / `auth` / `system`
`entityType` + `entityId`	e.g. `prescription` + `PRX-A1B2C3`
`parentEntityId`	e.g. parent appointment for a prescription
`actor.uid` / `actor.email` / `actor.role`	`admin` / `doctor` / `prescriber` / `pharmacy` / `patient` / `system`
`source` + `sourceName`	Originating service + handler name
`summary`	Human-readable one-line description
`diff[]`	Field-level before/after for mutations
`phiRedacted`	`true` when PHI fields were stripped before logging
`correlationId`	Request-scoped UUID; lets you trace partner request → all downstream writes
`ipAddress` / `userAgent`	Source identification

Coverage is enforced automatically in our CI pipeline: any change that writes to an audited entity without an accompanying audit hook fails the build. Logs are retained for **6 years** (HIPAA § 164.530(j)) and are queryable by partner on request.

Logical separation & access control

- Production and sandbox run in fully isolated environments with non-overlapping credentials. Sandbox contains only synthetic data — never real PHI.
- Internal staff access to PHI is gated by SSO + MFA + role-based authorization (`admin`, `pharmacy`, `prescriber`, etc.) and is itself audit-logged under `category: "phi_view"`.

- Engineers do not have direct read access to the production datastore; production debugging goes through admin tooling that emits audit logs.

Sub-processors

The current sub-processor list (a copy travels with the BAA and is updated on change):

Vendor	Purpose	Region
Google Cloud (compute, database, object storage, KMS)	Primary data + compute	`us-east5` (Columbus, Ohio)
Twilio (SendGrid, Twilio Programmable Messaging)	Patient/doctor email + SMS	US
Stripe	Payment processing	US
RxVortex / Wells Pharmacy	US prescription fulfillment	US
Junction Health	Lab order routing	US
Airtable	Operational record-keeping (de-identified)	US
Resend	Transactional email (no PHI body)	EU/US

All sub-processors with PHI access have executed BAAs.

3.3 Where PHI Crosses the Wire

Data class	Endpoint(s)	Notes
Patient demographics, contact	`GET /partner/patients/:id`	Returned only to clinical-tier keys whose partner created the originating referral.
Intake responses (Q&A)	`GET /partner/patients/:id/intake`	Includes free-text answers; treat as full PHI.
Appointments, prescriptions	`GET /partner/appointments/:id`	Includes prescriber identity, medication, dosage.
Messages	`GET /partner/patients/:id/messages`	Doctor ↔ patient conversation transcripts.
Lab orders, requisitions, biomarker results	`GET /partner/patients/:id/lab-orders`, `GET /partner/lab-orders/:id`, `GET /partner/lab-orders/:id/requisition`, `GET /partner/lab-orders/:id/results`	Lab panel selections, requisition PDFs, and final biomarker values are all PHI. Production access requires a signed lab-orders BAA addendum (a lab-orders entitlement on your partner account); sandbox is open to every clinical-tier partner. Requisition PDFs are streamed inline from our lab partner at request time — no persistent server-side copy exists in v1. The list and detail endpoints intentionally do NOT include biomarker values; those are reachable only via `/results`, which emits a dedicated audit row on every access.
Patient documents (binary files)	`POST /partner/patients/:id/documents`, `GET /partner/patients/:id/documents`, `GET /partner/documents/:id`	Partner-uploaded clinical files (lab PDFs, requisitions, ID photos) are PHI. Production access requires a signed documents BAA addendum (`features.documents` on your partner account); sandbox is open to every clinical-tier partner. Upload is base64-in-JSON, ≤ 10 MiB, PDF/PNG/JPEG/HEIC/TIFF only. See § 12A.
Inline patient creation	`POST /partner/prescriptions` (`inline_patient` mode)	Partner is asserting they have a HIPAA-permissible reason to disclose this PHI to Stealth Health.
Webhook event payloads	All `*.created`, `*.updated`, `prescription.received`, `transaction.*`, and the `referral.*` shipping events (`referral.awaiting_shipment` /	Signed; same retention rules as above.

	`shipped` / `in_transit` / `out_for_delivery` / `delivered`)	

The **referral tier never receives the items above** — only `referral_id`, status timestamps, and aggregate transaction status.

3.4 Partner Compliance Obligations

Requirement	Detail
Encryption at rest	All PHI persisted by the partner must be encrypted with AES-256 or equivalent. Cloud-vendor managed keys are acceptable.
Encryption in transit	TLS 1.2+ for all internal services that touch PHI received via the API.
Access controls	Role-based access; only personnel with documented need-to-know may view PHI. MFA required for any console with PHI access.
Audit logging	Log all PHI access events (who, what, when, source IP) and retain for 6 years. See § 17.5.
Minimum necessary	Only request and store the minimum PHI needed. Do not call `GET /partner/patients/:id/intake` if you only need the appointment status.
Breach notification	Notify `security@stealth.health` within 24 hours of a suspected breach.
Annual review	Stealth Health conducts an annual review of the partner's PHI handling against the items in this table.
Data retention	PHI must be purged within 30 days of contract termination or patient opt-out. Aggregate de-identified analytics may be retained.
Subcontractor flow-down	Any partner sub-processor that handles PHI received from this API must execute a BAA with the partner.
Workforce training	Annual HIPAA training for any workforce member with access to PHI received via the API.

3.5 Breach Notification (Both Directions)

Direction	Trigger	Channel	Timeline
Partner → Stealth Health	Any unauthorized access, disclosure, loss, or compromise of PHI received via this API.	`security@stealth.health` (PGP key on request) + the security contact named in the BAA.	Within 24 hours of discovery; full incident report within 60 days.
Stealth Health → Partner	Any incident affecting PHI sourced from a referral the partner created.	Security contact named on your partner account.	Within 24 hours of confirmation; ongoing updates per BAA.

3.6 Annual Review & Cutover Checklist

Each calendar year, partners are asked to re-attest to the items in § 3.4 and to confirm:

1. Sub-processor list is current.
2. Workforce HIPAA training is up to date.
3. Webhook signing secret has been rotated within the last 12 months.
4. Audit-log retention is verifiable (a sample log can be produced on request).
5. Disaster-recovery plan covers PHI received via this API.

---

4. Authentication & Security

Authentication is identical to the Referral Tier but with an additional scope header:

```
GET /partner/patients HTTP/1.1
Host: api.stealth.health  (or sandbox.stealth.health)
X-Partner-ID: ptr_acme_health
X-Api-Key: sk_live_7f3a...redacted
X-Access-Tier: clinical
Content-Type: application/json
```

Header	Purpose
`X-Partner-ID`	Identifies the partner (public, safe to log)
`X-Api-Key`	Authenticates the request (secret)
`X-Access-Tier`	Must be `clinical` — requests to clinical-tier endpoints without this header return `403`

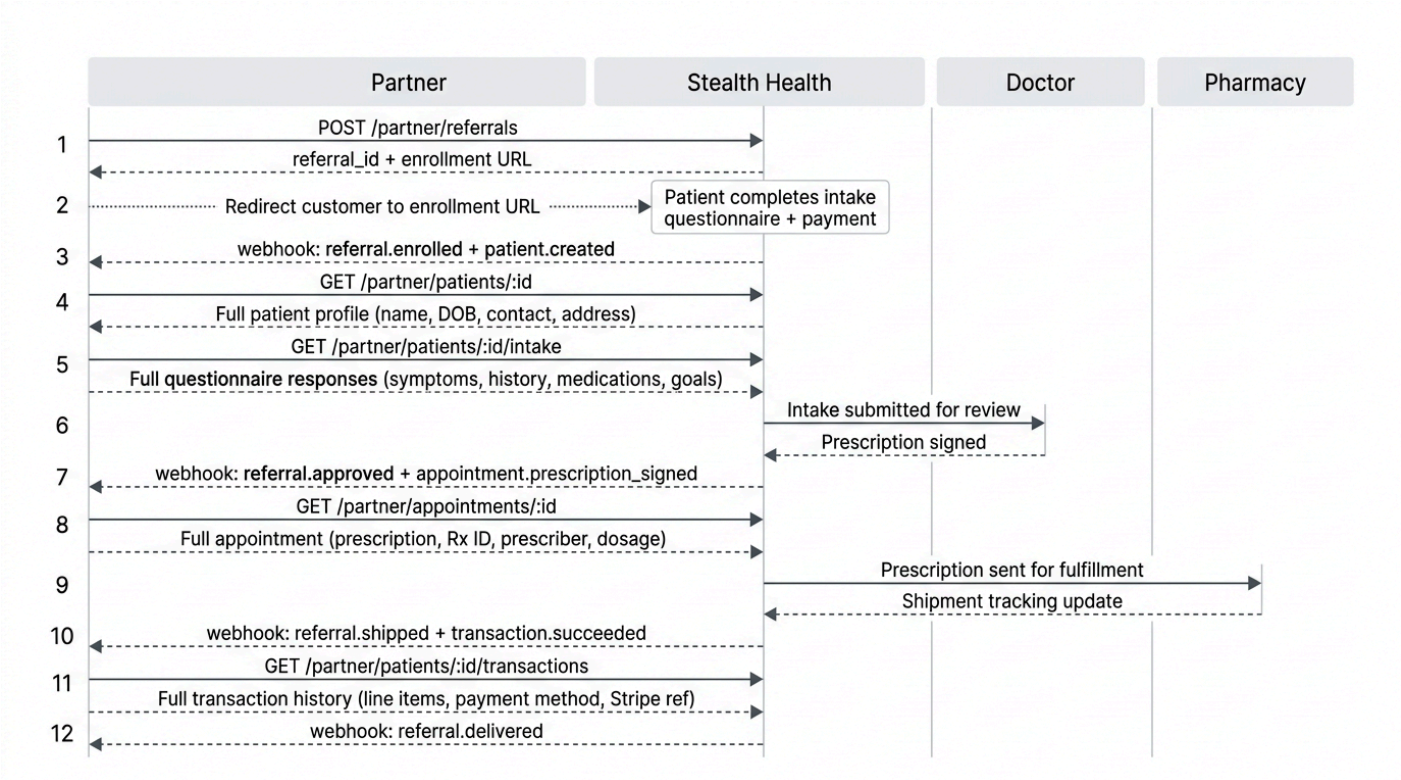
All other security features (key rotation, webhook signatures, TLS, IP allowlisting) are the same as the Referral Tier guide, Section 3.

---

5. Integration Flow

The enrollment flow is the same as the Referral Tier — the partner creates a referral, the customer completes enrollment, and doctors review the intake. The difference is in what the partner can query afterward.

5.1 Sequence Diagram



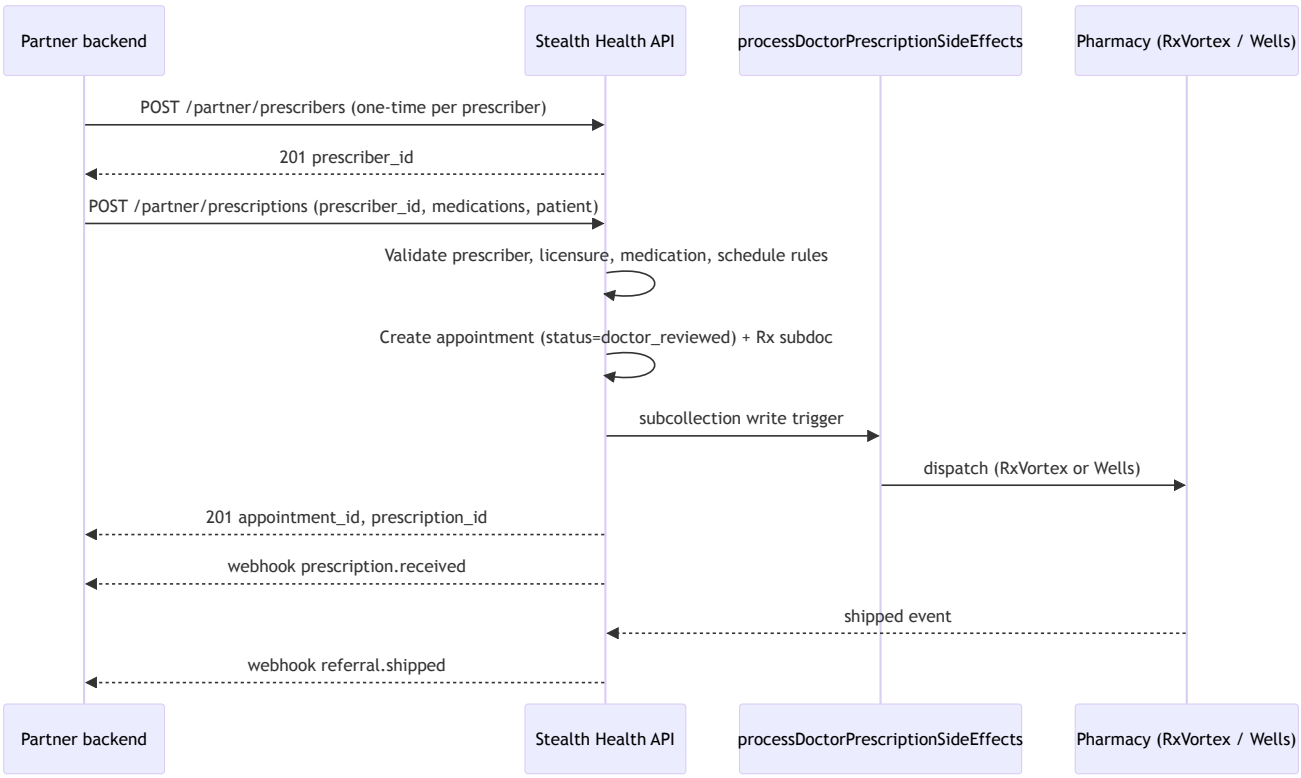
> ***Two prescribing flows are supported:***
>
> 1. ***Stealth-prescriber flow (default)*** *— described below: Stealth Health doctors review the intake and sign the prescription.*
> 2. ***Partner-prescriber flow*** *— Clinical-tier partners that operate their own prescribing physicians can register those prescribers and submit pre-signed prescriptions directly via* `POST /partner/prescriptions` *. See § 5.3 and § 11.1.*

5.2 Step-by-Step

1. **Partner creates a referral** → `POST /partner/referrals` (same as Referral Tier).
2. **Customer enrolls** → completes intake + payment on co-branded enrollment page.
3. **Partner receives webhook** → `referral.enrolled` (same as Referral Tier).
4. **Partner queries patient data** → `GET /partner/patients/:patient_id` — full profile.
5. **Partner queries intake responses** → `GET /partner/patients/:patient_id/intake` — full Q&A.
6. **Doctor reviews** → approves or denies.
7. **Partner receives webhook** → `referral.approved` with appointment + prescription detail.
8. **Partner queries appointment** → `GET /partner/appointments/:appointment_id` — full appointment record.
9. **Partner queries messages** → `GET /partner/patients/:patient_id/messages` — doctor/patient message threads.
10. **Partner queries transactions** → `GET /partner/patients/:patient_id/transactions` — payment history.
11. **Fulfillment proceeds** → partner receives shipping webhooks with tracking details.

5.3 Alternate Flow: Partner-Submitted Prescriptions

Clinical Partners that employ their own prescribing physicians (e.g. a partner telemedicine platform whose doctors evaluate the patient inside the partner's UI) can skip Stealth Health's clinician review entirely. Stealth Health acts as the **fulfillment network only** — accepting a pre-signed prescription, validating prescriber licensure and jurisdiction, and routing to the appropriate pharmacy.



Step-by-step:

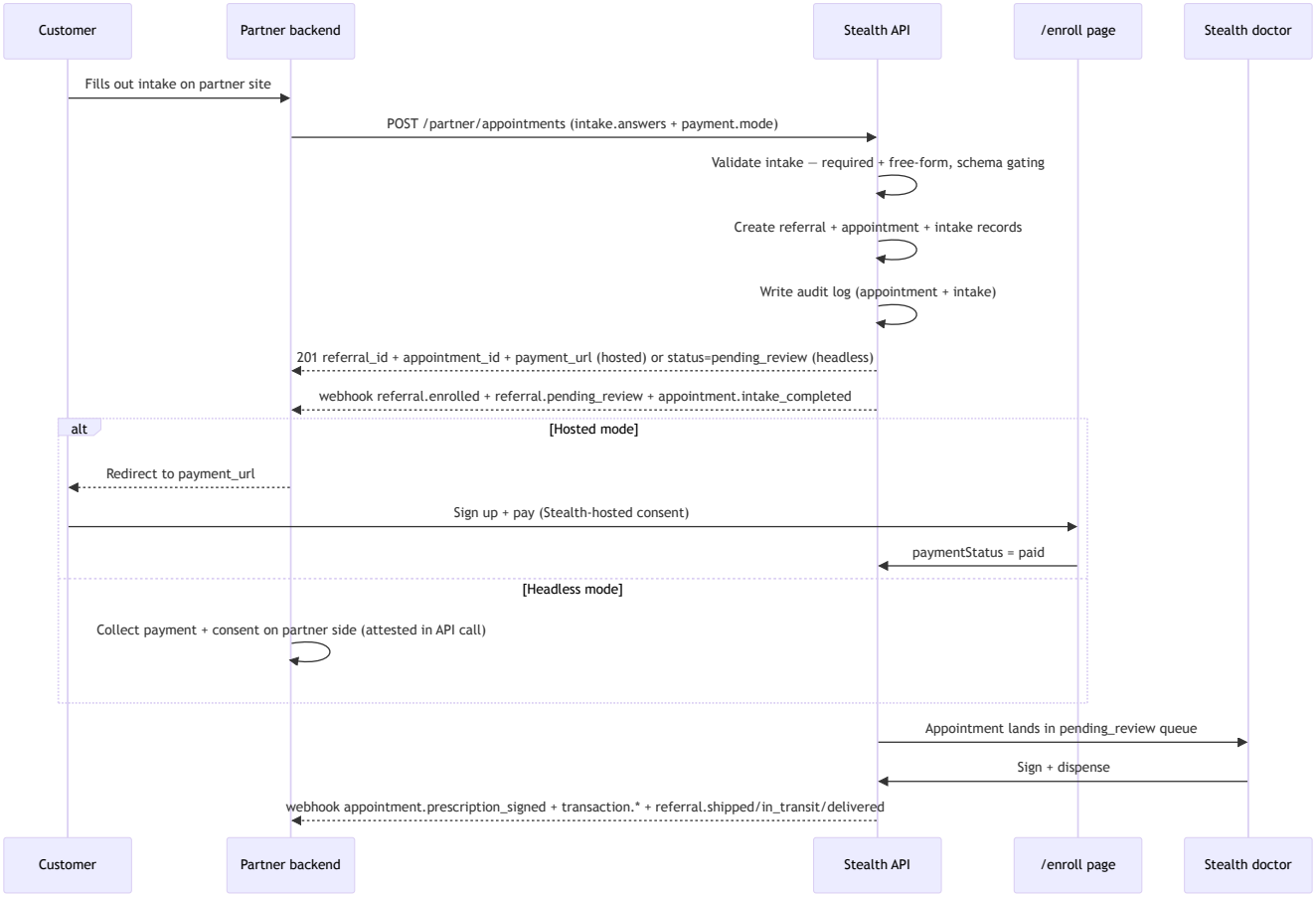
1. **One-time per prescriber** → `POST /partner/prescribers` registers each licensed prescriber + their state/provincial licenses + (US) DEA number.
2. **Per prescription** → `POST /partner/prescriptions` with the `prescriber_id`, the `medications[]`, and one of `patient_id` / `referral_id` / `inline_patient` (for first-time patients).
3. Stealth Health validates the prescriber is active, licensed in the patient's jurisdiction, and (for Schedule II–V in the US) carries a DEA. See § 11.2.
4. Stealth Health creates an internal `appointment` with `status: doctor_reviewed` and writes the prescription record. The existing fulfillment trigger picks it up and routes to RxVortex / Wells exactly as it does for Stealth-prescribed orders.
5. Partner receives `prescription.received` immediately, then the standard `transaction.*` and `referral.*` shipping webhooks (`referral.awaiting_shipment` → `referral.shipped` → `referral.in_transit` → `referral.delivered`) as the order progresses. For clinical-tier partners, the `referral.shipped` / `in_transit` / `delivered` payloads carry `carrier` and `tracking_number` in `data`. There is no `fulfillment.*` event family — listen for `referral.*`.

Out of scope for v1 (deferred to v2):

- EPCS — electronic prescribing of controlled substances. v1 accepts pre-signed PDF prescriptions out-of-band.
- DIN-level / narcotic-specific gating in Canada.
- Real-time PDMP checks.
- Batch submission.

5.4 Alternate Flow: Partner-Submitted Intake (Stealth doctor reviews partner-collected questionnaire)

Some clinical-tier partners want to keep the patient inside their own UI through the questionnaire step but still have **Stealth Health's** physician network do the review and prescribe. The `POST /partner/appointments` endpoint accepts a partner-collected intake response, runs hybrid validation (required clinical fields + partner free-form), creates a referral + appointment + intake document, and either (a) redirects the patient to Stealth's hosted `/enroll/{category}` page for payment + final consent (`hosted` mode) or (b) records that the partner already collected payment and consent on their side (`headless` mode, requires the BAA's Headless Consent Addendum).



Hybrid validation contract. The intake schema for each `product_category` is published in the `GET /partner/products` response under `required_intake_keys[]` and

`recommended_intake_keys[]`. Required fields gate the submit; recommended fields surface in the discovery response but never reject. Anything outside the schema is preserved verbatim under `category: "partner_freeform"` so the reviewing physician sees it on the chart.

The `COMMON_REQUIRED` baseline applies to every product_category:

Key	Type	Notes
`date_of_birth_attestation`	`boolean`	Partner attests they collected and verified DOB at intake.
`jurisdiction`	`string`	ISO 3166-2 subdivision (`TX` , `ON` , …). Cross-checked against patient.
`current_medications`	`string[]`	Empty array means "none" — the field still must be present.
`known_allergies`	`string[]`	Empty array means "none".
`medical_conditions`	`string[]`	Empty array means "none".
`consent_telehealth`	`boolean`	Hard reject if not `true` .
`consent_phi_release`	`boolean`	Hard reject if not `true` .

Per-category `required` examples (full list in `GET /partner/products`):

Category	Required additions
`trt-cream`	`prior_testosterone_level_ng_dl` (number), `symptom_duration` (enum)
`trt-injection`	`prior_testosterone_level_ng_dl` , `symptom_duration`
`weight-loss`	`height_cm` , `weight_kg` , `diabetes_status` (enum)
`ed`	`ed_severity` (enum), `cardiovascular_history` (string[])

Payload shape. A minimal hosted-mode submit looks like:

```
{
  "partner_reference": "tim-2026-05-14-001",
  "product_category": "trt-cream",
  "inline_patient": {
    "first_name": "Jane",
    "last_name": "Patient",
    "email": "jane@example.com",
    "dob": "1985-04-12",
    "address": { "street": "5 Capitol Way", "city": "Austin", "jurisdiction": "TX",
"postal_code": "73301", "country": "US" }
  },
  "intake": {
    "schema_version": 1,
    "partner_form_id": "trt_intake_v3",
    "answers": [
      { "key": "date_of_birth_attestation", "value": true },
      { "key": "jurisdiction", "value": "TX" },
      { "key": "current_medications", "value": ["levothyroxine"] },
      { "key": "known_allergies", "value": [] },
      { "key": "medical_conditions", "value": ["hypogonadism"] },
      { "key": "consent_telehealth", "value": true, "consent_document_version": "v3" },
      { "key": "consent_phi_release", "value": true, "consent_document_version": "v2" },
      { "key": "prior_testosterone_level_ng_dl", "value": 240 },
      { "key": "symptom_duration", "value": "6_to_12_months" },
      { "key": "favorite_lift", "question_text": "Favorite lift?", "value": "Deadlift" }
    ]
  },
  "payment": { "mode": "hosted" }
}
```

> **`payment.mode` defaults to `"hosted"` when omitted.** *The mode (and, for `embedded_checkout` / `authorize_only`, the `return_url` / `amount_cents`) must be nested under the `payment` object — top-level `payment_mode` / `return_url` keys are ignored. If the `payment` object is missing or `payment.mode` is absent, the request is treated as `hosted` and the `201` returns a hosted `payment_url`. To use embedded checkout (immediate capture or authorize-only), you must send the `payment.mode` explicitly.*

For `payment.mode = "headless"` the partner additionally sends:

```
{
  "consent_attestation": {
    "patient_ip": "203.0.113.45",
    "patient_user_agent": "PartnerApp/1.2 (iOS 17)",
    "signed_at": "2026-05-14T18:42:00Z",
    "document_version": "stealth-tos-v3",
    "document_sha256": "9b2c…",
    "captured_by": "partner_widget@1.0.4"
  }
}
```

The `patient_ip` is bucketed to a `/24` (IPv4) or `/64` (IPv6) before persistence — Stealth never stores raw client IPs from a partner-hosted intake.

For `payment.mode = "embedded_checkout"` the partner instead receives a Stripe **Embedded Checkout** `client_secret` and renders the payment form inside their own page (Stripe's `@stripe/react-stripe-js` `<EmbeddedCheckout>` or `stripe.initEmbeddedCheckout`). This lets the partner collect the full shipping

- billing address natively in Stripe (no separate address form) while keeping the customer on a partner-branded page:

```
{
  "payment": {
    "mode": "embedded_checkout",
    "return_url": "https://app.partner.com/checkout/complete?session={CHECKOUT_SESSION_ID}",
    "amount_cents": 19900,
    "product_name": "TRT Program — Initial"
  }
}
```

- `return_url` (**required**, must be `https://`) — where Stripe redirects after the embedded session completes.
- `amount_cents` (optional, positive integer minor units) — overrides the catalog-resolved price. Omit to let Stealth resolve the program's default amount; a `CHECKOUT_AMOUNT_UNRESOLVED` (400) is returned if neither a catalog amount nor `amount_cents` is available.
- `product_name` (optional) — the line-item label shown in Checkout.

The `201` response carries `payment.mode: "embedded_checkout"` with a `client_secret` and `session_id` instead of a `payment_url`, plus the `publishable_key` you mount Stripe with:

```
{
  "status": "awaiting_payment",
  "payment": {
    "mode": "embedded_checkout",
    "client_secret": "cs_test_…_secret_…",
    "session_id": "cs_test_…",
    "publishable_key": "pk_test_…",
    "stripe_account": "acct_…",
    "payment_status": "awaiting_payment"
  }
}
```

- `publishable_key` — **Stealth's** Stripe publishable key for the account this session was created on. **Use this value to initialize Stripe.js** — Stealth is the merchant of record, so you do *not* (and must not) use a publishable key of your own. The key is account-scoped to `stripe_account`, so `loadStripe(publishable_key)` is sufficient — you don't pass a separate `stripeAccount` option. The correct key is selected server-side per the session's mode (test vs live) and currency/region (US vs CA), so always read it from the response rather than hard-coding one.
- `stripe_account` — the connected Stripe account the session belongs to (informational; for logging/debugging).

Mount it with Stripe's Embedded Checkout:

```
import { loadStripe } from "@stripe/stripe-js";
import { EmbeddedCheckoutProvider, EmbeddedCheckout } from "@stripe/react-stripe-js";

const stripePromise = loadStripe(payment.publishable_key);

<EmbeddedCheckoutProvider
  stripe={stripePromise}
  options={{ clientSecret: payment.client_secret }}
>
  <EmbeddedCheckout />
</EmbeddedCheckoutProvider>;
```

When the customer completes payment, Stealth's Stripe webhook reconciles the Stripe-collected billing + shipping address **back onto the appointment** (`patientAddress`, `shippingAddress`, `patientCountry`, `patientJurisdiction`) before the case reaches clinician review — so the prescriber always sees a complete shipping destination. This write is HIPAA-audited (`appointment.partner_checkout.address_applied`) and idempotent. Replaying the same idempotent submit mints a fresh `client_secret` (Stripe Checkout client secrets are single-session).

`authorize_only` — pre-authorize the card, capture only on clinical approval

`payment.mode: "authorize_only"` is `embedded_checkout` with **manual capture**: completing the embedded session places an **authorization hold** on the customer's card (funds reserved, *not* charged). The charge is only captured after Stealth's reviewing physician approves the case; if the patient is clinically ineligible, the hold is voided and **the patient is never charged**. This supports intake flows that put the payment step ahead of the final clinical screening without risking a charge-then-refund cycle for ineligible patients.

Request shape is identical to `embedded_checkout` (same `return_url` / `amount_cents` / `product_name` fields, same `<EmbeddedCheckout>` mount):

```
{
  "payment": {
    "mode": "authorize_only",
    "return_url": "https://app.partner.com/checkout/complete?session={CHECKOUT_SESSION_ID}",
    "amount_cents": 19900,
    "product_name": "TRT Program — Initial"
  }
}
```

The `201` response mirrors the `embedded_checkout` block with `payment.mode: "authorize_only"` and `payment_status: "awaiting_authorization"`.

Lifecycle:

Step	Trigger	Appointment `paymentStatus`	Webhook
1. Submit	`POST /partner/appointments`	`awaiting_payment`	`referral.enrolled`, `referral.pending_review`, `appointment.intake_completed`
2. Card authorized	Customer completes embedded Checkout	`authorized` (funds held)	—
3a. Clinically approved	Stealth physician signs the prescription	`paid` (hold captured)	`referral.approved`, `appointment.prescription_signe` `transaction.succeeded` (with `capture_of_authorization: true`
3b. Clinically ineligible	Stealth physician declines the case	`authorization_released` (hold voided)	`referral.denied`, `prescription.rejected`, `transaction.authorization_relea`

Things to know:

- **Authorization window.** Card authorizations expire (typically **7 days** for most card networks). Clinical review normally completes well inside that window; if a hold lapses before review, the capture fails, the failure is audited, and the case is handled as unpaid (`referral.payment_due` fires so you can re-collect). Don't submit `authorize_only` cases you expect to sit in review for more than a few days.
- The address reconciliation described above happens at **authorization** time (step 2), so the reviewing physician still sees the real Stripe-collected address.
- The hold amount is fixed at authorization. Partial captures are not supported in v1 — the captured amount always equals the authorized amount.
- **Per-partner opt-in.** `authorize_only` is gated by `features.partner_submitted_intake.allow_authorize_only` on your partner account (scoped to your API key, e.g. `ptr_…`), so enabling it for one partner changes nothing for anyone else. Production requires the flag (`AUTHORIZE_ONLY_NOT_ENABLED`, 403, when absent); sandbox is open to every clinical-tier partner.
- Idempotent replays re-mint a fresh manual-capture `client_secret` bound to the same appointment, exactly like `embedded_checkout` replays.

Patient resolution. Mirrors `POST /partner/prescriptions`. Provide exactly one of:

- `patient_id` — for follow-up cases (an existing Stealth patient already associated with this partner).
- `referral_id` — when the partner already created a referral via `POST /partner/referrals`.
- `inline_patient` — first-time patients. Stealth provisions the patient profile and referral records atomically.

Hosted vs headless. Both modes create the same appointment record and fire the same downstream webhooks (`referral.enrolled`, `referral.pending_review`, `appointment.intake_completed`). The differences are:

	Hosted	Headless
Customer touches Stealth UI?	Yes — redirected to `/enroll/{category}?ref=…&questionnaireResponseId=…`	No
Who collects consent?	Stealth `/enroll` page (Stealth-direct disclosures)	Partner UI (`consent_attestation` block, BAA Headless Addendum required)
Who collects payment?	Stealth (Stripe on `/enroll`)	Partner. Stealth records `paymentStatus: "partner_collected"` .
BAA addendum required?	None beyond standard Clinical Tier BAA.	**Headless Consent Addendum** enabled on your partner account.
Webhook trio fires when?	Immediately on POST (intake is complete on our side).	Immediately on POST.
Default for new partners?	Yes (sandbox + production).	Off — must be explicitly enabled.

Idempotency. A repeat submit with the same `(partner_id, partner_reference)` returns the original `appointment_id` and does not re-create any docs. Use this on retry from a network glitch — never to update intake (intake is immutable once submitted; create a new submission with a new `partner_reference` if the patient amends).

Validation error codes (full table in § 15):

- `INTAKE_REQUIRED_FIELD_MISSING` (422) — `error.details.missing_keys[]` lists the keys.
- `INTAKE_FIELD_INVALID` (422) — `error.details.invalid_fields[]` lists `{ key, reason }` .
- `INTAKE_CONSENT_DECLINED` (422) — `consent_telehealth` or `consent_phi_release` was `false` .
- `CONSENT_ATTESTATION_REQUIRED` (422) — headless mode without `consent_attestation` .
- `PAYMENT_MODE_INVALID` (400) — `payment.mode` not in `{"hosted", "headless", "embedded_checkout", "authorize_only"}` .
- `PAYMENT_MODE_CONFLICT` (409) — an idempotent replay (same `partner_reference`) explicitly sent a `payment.mode` that differs from the mode the original submission was created with. The payment mode is fixed at first submit and cannot be switched by re-POSTing; re-send with the original mode (or omit `payment.mode`), or use a new `partner_reference` for a new appointment. This guard exists so a replay can never

silently hand back a `hosted` `payment_url` to a caller that asked for `embedded_checkout`.

- `RETURN_URL_REQUIRED` (400) — `embedded_checkout` / `authorize_only` mode without an `https://` `payment.return_url`.
- `PAYMENT_AMOUNT_INVALID` (400) — `payment.amount_cents` is not a positive integer.
- `CHECKOUT_AMOUNT_UNRESOLVED` (400) — `embedded_checkout` / `authorize_only` mode and no catalog amount could be resolved; pass `payment.amount_cents` explicitly.
- `PARTNER_SUBMITTED_INTAKE_NOT_ENABLED` (403) — production access without the BAA addendum.
- `HEADLESS_NOT_ENABLED` (403) — production partner attempted headless without the Headless Consent Addendum.
- `AUTHORIZE_ONLY_NOT_ENABLED` (403) — production partner attempted `authorize_only` without the per-partner `allow_authorize_only` flag.
- `PHI_IN_QUERY_STRING` (400) — guard against partners that mistakenly stuff intake into the URL.

Out of scope for v1 (deferred):

- `/enroll` appointment-ID reuse — in `hosted` mode v1, the customer payment on `/enroll` lands on a fresh appointment doc. The partner-submitted-intake appointment carries the intake; the doctor portal's existing payment-status filter delays review until payment is reconciled. v1.1 will make `/enroll` appointment-aware so the same doc carries both intake and payment.
- Inline Stripe charge in `headless` mode (taking a `payment_method_id` and charging on Stealth's MID). v1 records the partner's attestation only.
- Patient-amends-intake. v1 treats intake as immutable post-submit.

---

6. API Reference — Patients

All clinical-tier endpoints are prefixed with `/partner` and require the `X-Access-Tier: clinical` header.

6.1 GET /partner/patients

List all patients associated with this partner.

Query Parameters:

Param	Type	Default	Description
`status`	string	(all)	Filter: `active`, `pending`, `inactive`
`product_category`	string	(all)	Filter by enrolled category
`search`	string	—	Search by name or email (partial match)
`created_after`	ISO 8601	—	Patients created after this timestamp
`created_before`	ISO 8601	—	Patients created before this timestamp
`limit`	integer	50	Max results per page (1–200)
`cursor`	string	—	Pagination cursor

Response (200 OK):


```
{
  "patients": [
    {
      "patient_id": "pat_8f2e9a1b",
      "partner_reference": "cust_12345",
      "first_name": "John",
      "last_name": "Doe",
      "email": "john.doe@example.com",
      "phone": "+15551234567",
      "date_of_birth": "1985-06-15",
      "gender": "male",
      "address": {
        "line1": "123 Main St",
        "city": "Austin",
        "state": "TX",
        "postal_code": "78701",
        "country": "US"
      },
      "product_categories": ["trt-cream"],
      "status": "active",
      "created_at": "2026-03-02T14:15:00Z"
    }
  ],
  "pagination": {
    "has_more": false,
    "next_cursor": null
  }
}
```

6.2 GET /partner/patients/:patient_id

Retrieve full profile for a single patient.

Response (200 OK):


```
{
  "patient_id": "pat_8f2e9a1b",
  "partner_reference": "cust_12345",
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.doe@example.com",
  "phone": "+15551234567",
  "date_of_birth": "1985-06-15",
  "gender": "male",
  "age": 40,
  "address": {
    "line1": "123 Main St",
    "city": "Austin",
    "state": "TX",
    "postal_code": "78701",
    "country": "US"
  },
  "product_categories": ["trt-cream"],
  "status": "active",
  "latest_appointment_id": "apt_c4d5e6f7",
  "latest_appointment_status": "approved",
  "total_appointments": 1,
  "created_at": "2026-03-02T14:15:00Z",
  "updated_at": "2026-03-03T09:30:00Z"
}
```


---

7. API Reference — Appointments

7.1 GET /partner/patients/:patient_id/appointments

List all appointments for a patient.

Query Parameters:

Param	Type	Default	Description
`status`	string	(all)	Filter: `pending_review`, `approved`, `denied`, `completed`
`limit`	integer	50	Max per page (1–200)
`cursor`	string	—	Pagination cursor

Response (200 OK):

```
{
  "appointments": [
    {
      "appointment_id": "apt_c4d5e6f7",
      "patient_id": "pat_8f2e9a1b",
      "referral_id": "ref_abc123xyz",
      "product_category": "trt-cream",
      "condition": "Testosterone Replacement Therapy",
      "status": "approved",
      "submitted_at": "2026-03-02T14:15:00Z",
      "reviewed_at": "2026-03-03T09:30:00Z",
      "prescription_summary": {
        "status": "signed",
        "medications": [
          {
            "name": "Testosterone Cream 200mg/mL",
            "dosage": "1mL applied topically daily",
            "quantity": 1,
            "quantity_unit": "tube (30mL)",
            "repeats": 3
          }
        ],
        "prescriber": "Dr. Smith",
        "signed_at": "2026-03-03T09:30:00Z"
      },
      "fulfillment": {
        "status": "shipped",
        "carrier": "USPS",
        "tracking_number": "9400111899223100001234",
        "estimated_delivery": "2026-03-07",
        "shipped_at": "2026-03-04T11:00:00Z"
      },
      "payment": {
        "status": "paid",
        "amount_cents": 14900,
        "currency": "USD",
        "paid_at": "2026-03-02T14:15:00Z"
      }
    }
  ],
  "pagination": {
    "has_more": false,
    "next_cursor": null
  }
}
```


7.2 GET /partner/appointments/:appointment_id

Retrieve a single appointment with full detail.

Response (200 OK):

```
{
  "appointment_id": "apt_c4d5e6f7",
  "patient_id": "pat_8f2e9a1b",
  "referral_id": "ref_abc123xyz",
  "partner_reference": "cust_12345",
  "product_category": "trt-cream",
  "condition": "Testosterone Replacement Therapy",
  "status": "approved",
  "submitted_at": "2026-03-02T14:15:00Z",
  "reviewed_at": "2026-03-03T09:30:00Z",
  "patient_snapshot": {
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "date_of_birth": "1985-06-15",
    "gender": "male",
    "address": {
      "line1": "123 Main St",
      "city": "Austin",
      "state": "TX",
      "postal_code": "78701",
      "country": "US"
    }
  },
  "intake_summary": {
    "total_questions": 18,
    "completed_questions": 18,
    "severity_score": null,
    "goals": "Improve energy, libido, and body composition"
  },
  "prescription_summary": {
    "status": "signed",
    "rx_id": "RX-2026-0302-001",
    "medications": [
      {
        "name": "Testosterone Cream 200mg/mL",
        "generic_name": "Testosterone Cypionate",
        "dosage": "1mL applied topically daily",
        "quantity": 1,
        "quantity_unit": "tube (30mL)",
        "repeats": 3,
        "notes": null
      }
    ],
    "prescriber": "Dr. Smith",
    "prescriber_license": "TX-MD-12345",
    "signed_at": "2026-03-03T09:30:00Z"
  },
  "fulfillment": {
    "status": "shipped",
    "carrier": "USPS",
    "tracking_number": "9400111899223100001234",
    "label_url": null,
```

```
    "estimated_delivery": "2026-03-07",
    "shipped_at": "2026-03-04T11:00:00Z",
    "delivered_at": null
  },
  "payment": {
    "status": "paid",
    "amount_cents": 14900,
    "currency": "USD",
    "method": "card",
    "paid_at": "2026-03-02T14:15:00Z",
    "stripe_payment_intent_id": "pi_3abc123def456"
  },
  "created_at": "2026-03-02T14:15:00Z",
  "updated_at": "2026-03-04T11:00:00Z"
}
```


---

8. API Reference — Intake Responses

8.1 GET /partner/patients/:patient_id/intake

Retrieve the full intake questionnaire responses for a patient's most recent appointment.

Query Parameters:

Param	Type	Default	Description
`appointment_id`	string	(latest)	Specific appointment; defaults to most recent

Response (200 OK):

```
{
  "patient_id": "pat_8f2e9a1b",
  "appointment_id": "apt_c4d5e6f7",
  "form_title": "TRT Intake Questionnaire",
  "submitted_at": "2026-03-02T14:15:00Z",
  "responses": [
    {
      "question": "What symptoms are you experiencing?",
      "answer": "Low energy, decreased libido, difficulty maintaining muscle mass",
      "category": "symptoms"
    },
    {
      "question": "How long have you been experiencing these symptoms?",
      "answer": "6–12 months",
      "category": "symptoms"
    },
    {
      "question": "Have you had your testosterone levels tested?",
      "answer": "Yes",
      "category": "medical_history"
    },
    {
      "question": "What was your most recent total testosterone level (ng/dL)?",
      "answer": "285",
      "category": "medical_history"
    },
    {
      "question": "Are you currently taking any medications?",
      "answer": "Lisinopril 10mg daily",
      "category": "medications"
    },
    {
      "question": "Do you have any known allergies?",
      "answer": "None",
      "category": "allergies"
    },
    {
      "question": "Do you have any of the following conditions? (select all that apply)",
      "answer": "None of the above",
      "category": "medical_conditions"
    },
    {
      "question": "What are your health goals for this treatment?",
      "answer": "Improve energy, libido, and body composition",
      "category": "goals"
    }
  ]
}
```

> *Responses are returned in the order they appeared on the questionnaire. The `category` field groups questions for easier parsing.*

---

9. API Reference — Messages

9.1 GET /partner/patients/:patient_id/messages

Retrieve all message threads and their messages for a patient. Only threads linked to appointments associated with this partner are returned.

Query Parameters:

Param	Type	Default	Description
`status`	string	(all)	Filter threads by `open`, `awaiting_patient`, `awaiting_clinician`, or `doctor_reviewed`.
`appointment_id`	string	—	Restrict to threads linked to one appointment (must be partner-owned).
`updated_after`	ISO 8601	—	Only return threads whose `last_message_at` is newer than this timestamp. Use with the `messageThread.updated` webhook for incremental sync.
`limit`	integer	50	Max threads per page (1–50). Each thread inlines up to **200 messages** in chronological order; threads exceeding that cap include `messages_truncated: true` and a `next_message_cursor` you can pass to the per-thread endpoint (roadmap item, see § 9.2).
`cursor`	string	—	Pagination cursor for the thread list.

Response (200 OK):

```
{
  "patient_id": "pat_8f2e9a1b",
  "threads": [
    {
      "thread_id": "thr_abc123",
      "appointment_id": "apt_c4d5e6f7",
      "subject": "TRT Follow-up",
      "status": "open",
      "participants": [
        { "type": "patient", "name": "John Doe", "role": "Patient" },
        { "type": "doctor", "name": "Dr. Smith", "role": "Physician" }
      ],
      "last_message_at": "2026-03-05T10:30:00Z",
      "last_message_preview": "Your lab results look great...",
      "created_at": "2026-03-03T09:30:00Z",
      "messages": [
        {
          "message_id": "msg_001",
          "sender_type": "doctor",
          "sender_name": "Dr. Smith",
          "sender_role": "Physician",
          "channel": "portal",
          "body": "Hi John, your lab results look great. I'm approving your prescription.",
          "attachments": [],
          "status": "read",
          "read_at": "2026-03-05T11:00:00Z",
          "created_at": "2026-03-05T10:30:00Z"
        },
        {
          "message_id": "msg_002",
          "sender_type": "patient",
          "sender_name": "John Doe",
          "sender_role": "Patient",
          "channel": "portal",
          "body": "Thank you, doctor!",
          "attachments": [],
          "status": "delivered",
          "read_at": null,
          "created_at": "2026-03-05T11:15:00Z"
        }
      ]
    }
  ],
  "total_threads": 1
}
```

Message Object Fields

Field	Type	Description
`message_id`	string	Unique message identifier
`sender_type`	string	`"doctor"`, `"patient"`, or `"system"`
`sender_name`	string	Display name of the sender
`sender_role`	string	`"Physician"`, `"Patient"`, `"Care Team"`
`channel`	string	Always `"portal"`
`body`	string	Message text content
`attachments`	array	File attachments: `{ name, type, url }`
`status`	string	`"delivered"` or `"read"`
`read_at`	ISO 8601 | null	When the message was first read
`created_at`	ISO 8601	When the message was sent

Thread Object Fields

Field	Type	Description
`thread_id`	string	Unique thread identifier
`appointment_id`	string | null	Linked appointment
`subject`	string | null	Thread subject line
`status`	string	`"open"`, `"awaiting_patient"`, `"awaiting_clinician"`, `"doctor_reviewed"`
`participants`	array	`{ type, name, role }` for each participant
`last_message_at`	ISO 8601	Timestamp of most recent message
`last_message_preview`	string	Truncated preview of the last message
`created_at`	ISO 8601	When the thread was created
`messages`	array	All messages in chronological order (up to 200 per thread)

Common errors:

Code	HTTP	Meaning
`PATIENT_NOT_FOUND`	404	Patient does not exist or is not associated with this partner.
`CLINICAL_ACCESS_REQUIRED`	403	Endpoint requires a clinical-tier API key.
`QUERY_ERROR`	500	Internal query failed — retry with backoff.

9.2 Partner-submit policy & patient-authored messages

Messaging is **read-only** for partners on the Clinical Tier today. There is no partner-submit endpoint (e.g. no `POST /partner/patients/:id/messages`) and one is not on the v1 roadmap. The reason is clinical, not technical:

- Inbound messages on a Stealth Health appointment become part of the patient's medical record. Authoring them requires a credentialed clinician acting under the Stealth Health BAA — not a partner workforce member or a partner system process. Allowing partner-system writes would (a) blur the chart-of-record boundary the BAA depends on, and (b) bypass our existing audit trail on patient messaging (see § 3.2 Audit trail).
- Partner workforce members who are themselves licensed clinicians and who need to participate in the chart should onboard through the Prescriber-Partner registry (§ 11.1.1) and message through the doctor portal SSO handoff below — not through a partner-system endpoint.

Recommended patterns when your patients need to send a message to the clinician:

Pattern	When to use	How it works
Patient SSO into the Stealth-hosted portal (default)	Your patient already has a partner-side identity but no separate Stealth login.	Use the Partner SSO OIDC handoff to mint a short-lived RS256 JWT scoped to that patient. Link the patient out `https://app.stealth.health/messages` (or a deep link to a specific thread). They land authenticated; the existing portal messaging UI captures the reply and emi a `messageThread.updated` webhook back to you. No second signup, no PHI in your stack.
Embeddable messaging module (`@wearestealthhealth/messaging-embed`)	You want the messaging surface to render inside your own app shell with one-click SSO and no redirect.	Drop-in React component or vanilla-JS UMD bundle tha frames `/embed/messages` and handles the auth handshake for you. See § 9.3 below. Requires your origi on the per-partner `embed_allowed_origins` allowlist (self-serve via your account manager / reporting portal). Only the patient identity crosses the iframe boundary — the parent page never sees `messages[*].body` or thre metadata.
Polling + read-only mirror	Your UI only needs to display the conversation, not let the patient reply.	Call `GET /partner/patients/:patient_id/messages updated_after=<timestamp>` on the `messageThread.updated` webhook fan-out (or every 6 if you can't host a receiver). Render the threads inside your UI. Add a "Reply in patient portal" CTA that deep-links into `https://app.stealth.health/messages/<thread_i` so the actual write lands on our side.

Webhook signal for incremental sync. When a new message lands in a thread that's visible to your partner ID, we fire `messageThread.updated` with `{ thread_id, appointment_id, patient_id, last_message_at }` — no message body. Use it as a poke to call `GET /partner/patients/:id/messages?updated_after=...` rather than polling on a fixed interval. See § 13.2 Additional Clinical Events.

9.3 Embeddable messaging module

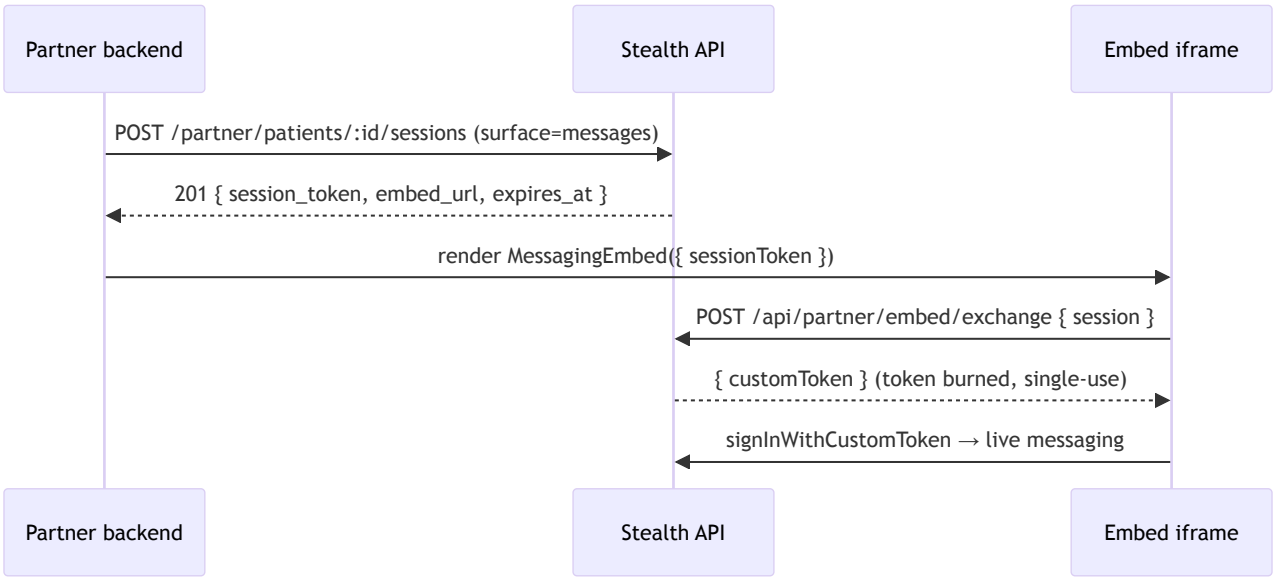
The `@wearestealthhealth/messaging-embed` package renders the secure patient messaging surface inside your app via an iframe, with a one-click SSO handoff. It ships in two forms:

- **npm / React** — `import { MessagingEmbed } from '@wearestealthhealth/messaging-embed'` .
- **Vanilla JS (UMD / CDN)** — `<script src="https://app.stealth.health/messaging-embed/latest/umd.js">`

`</script>` then `StealthHealthMessaging.render('#el', { sessionToken, ... })`.

Full integration contract (props, events, theming, ToS, mobile fallback, security model) lives in the package's `PARTNER_INTEGRATION.md`, provided with your messaging-embed integration package.

Auth handshake (two legs). The partner never receives a platform credential — only a short-lived, single-use opaque token.



Mint a session — `POST /partner/patients/:id/sessions` (clinical tier). Returns a `session_token` (prefix `emb_`, default 120s TTL, single-use) plus a ready-to-use `embed_url`:

```
{
  "session_token": "emb_…",
  "embed_url": "https://app.stealth.health/embed/messages?session=emb_…",
  "surface": "messages",
  "patient_id": "pat_123",
  "expires_at": "2026-05-14T18:44:00Z",
  "expires_in": 120
}
```

Pass the `session_token` to the embed component on every mount (mint a fresh one per load — they are single-use). Stealth verifies the partner owns the patient (via the originating referral) before minting.

`postMessage` event API. The iframe posts these events to the parent (`window.addEventListener('message', …)`), all `type`-prefixed `messaging:`:

Event	Payload	Meaning
`messaging:ready`	`{ threadCount }`	Threads loaded; UI interactive.
`messaging:unread-count`	`{ count }`	Unread thread count (drive a badge).
`messaging:tos-required`	`{ tosVersion }`	Patient must accept ToS before messaging.
`messaging:tos-accepted`	`{ tosVersion, acceptedAt }`	Patient accepted the ToS in-iframe.
`messaging:error`	`{ message }`	Auth/session failure; show your fallback.

ToS pre-display. Pass `tos: { version, url, require: true }` to gate the surface behind a Terms-of-Service acceptance during enrollment; acceptance is remembered per `(patient, version)` and bridged back via `messaging:tos-accepted`.

Theming. Pass `theme: { primaryColor, fontFamily }` to match your brand (applied via CSS variables inside the iframe).

Mobile fallback. Pass `mobileFallbackUrl` (e.g. an app deep link) and the iframe surfaces an "Open in the app instead" link on error / ToS gate.

CSP allowlist. The patient app emits `Content-Security-Policy: frame-ancestors 'self' <your origins>` on `/embed/*`, built from the union of every partner's `embed_allowed_origins`. Add your preprod + prod origins (bare hosts and `*.partner.com` wildcards accepted) via your account manager; they are managed in the reporting portal's white-label Partner API step. Until your origins are listed, the browser blocks framing.

---

10. API Reference — Transactions

10.1 GET /partner/patients/:patient_id/transactions

List all payment transactions for a patient.

Query Parameters:

Param	Type	Default	Description
`status`	string	(all)	`succeeded`, `pending`, `failed`, `refunded`
`created_after`	ISO 8601	—	Transactions after this timestamp
`limit`	integer	50	Max per page (1–200)
`cursor`	string	—	Pagination cursor

Response (200 OK):


```
{
  "patient_id": "pat_8f2e9a1b",
  "transactions": [
    {
      "transaction_id": "txn_a1b2c3d4",
      "appointment_id": "apt_c4d5e6f7",
      "type": "enrollment_payment",
      "status": "succeeded",
      "amount_cents": 14900,
      "currency": "USD",
      "description": "TRT Cream — enrollment + first fill",
      "payment_method": {
        "type": "card",
        "brand": "visa",
        "last4": "4242"
      },
      "stripe_payment_intent_id": "pi_3abc123def456",
      "created_at": "2026-03-02T14:15:00Z"
    }
  ],
  "summary": {
    "total_paid_cents": 14900,
    "total_refunded_cents": 0,
    "currency": "USD"
  },
  "pagination": {
    "has_more": false,
    "next_cursor": null
  }
}
```

10.2 GET /partner/transactions/:transaction_id

Retrieve a single transaction.

Response (200 OK):

```
{
  "transaction_id": "txn_a1b2c3d4",
  "patient_id": "pat_8f2e9a1b",
  "partner_reference": "cust_12345",
  "appointment_id": "apt_c4d5e6f7",
  "referral_id": "ref_abc123xyz",
  "type": "enrollment_payment",
  "status": "succeeded",
  "amount_cents": 14900,
  "currency": "USD",
  "description": "TRT Cream — enrollment + first fill",
  "line_items": [
    {
      "description": "Testosterone Cream 200mg/mL (30mL)",
      "amount_cents": 12900,
      "quantity": 1
    },
    {
      "description": "Physician consultation",
      "amount_cents": 2000,
      "quantity": 1
    }
  ],
  "payment_method": {
    "type": "card",
    "brand": "visa",
    "last4": "4242"
  },
  "stripe_payment_intent_id": "pi_3abc123def456",
  "refund": null,
  "created_at": "2026-03-02T14:15:00Z"
}
```


---

11. API Reference — Referrals & Products

The referral and product endpoints are identical to the Referral Tier. See Partner API Integration Guide, Section 6 for:

- `POST /partner/referrals` — Create a referral
- `GET /partner/referrals/:referral_id` — Get referral status
- `GET /partner/referrals` — List referrals
- `GET /partner/referrals/summary` — Reporting summary
- `GET /partner/products` — Available product categories
- `POST /partner/referrals/:referral_id/cancel` — Cancel a referral

In the Clinical Partner tier, the referral GET endpoints also include `patient_id` and `appointment_id` fields for cross-referencing:

```
{
  "referral_id": "ref_abc123xyz",
  "partner_reference": "cust_12345",
  "patient_id": "pat_8f2e9a1b",
  "appointment_id": "apt_c4d5e6f7",
  "product_category": "trt-cream",
  "status": "approved",
  "...": "..."
}
```

11.1 Partner-Submitted Prescriptions

Five endpoints power the partner-prescriber alternate flow. All require `tier: "clinical"` partner credentials.

11.1.1 POST /partner/prescribers

Register a prescribing physician one time. Subsequent prescription submissions reference the returned `prescriber_id`.

Request:

```
{
  "first_name": "Alice",
  "last_name": "Doctor",
  "email": "alice.doctor@example.com",
  "npi": "1234567890",
  "dea_number": "AB1234567",
  "partner_reference": "ALICE-001",
  "licenses": [
    { "country": "US", "jurisdiction": "TX", "license_number": "TX-MD-1", "expires_at": "2027-
06-30" },
    { "country": "CA", "jurisdiction": "ON", "license_number": "ON-MD-1", "expires_at": "2027-
06-30" }
  ]
}
```

Response (201 Created):

```
{
  "prescriber_id": "pres_a1b2c3d4e5f6",
  "status": "active",
  "first_name": "Alice",
  "last_name": "Doctor",
  "email": "alice.doctor@example.com",
  "npi": "1234567890",
  "dea_number": "AB1234567",
  "licenses": [
    { "country": "US", "jurisdiction": "TX", "license_number": "TX-MD-1", "expires_at": "2027-06-30" },
    { "country": "CA", "jurisdiction": "ON", "license_number": "ON-MD-1", "expires_at": "2027-06-30" }
  ],
  "created_at": "2026-04-23T12:00:00Z",
  "updated_at": "2026-04-23T12:00:00Z"
}
```


Validation rules:

- `first_name`, `last_name`, `email` required.
- `licenses` is a non-empty array of `{ country, jurisdiction, license_number, expires_at }`. `country` must be `US` or `CA`; `jurisdiction` must be a valid state/province code; `expires_at` must be a future ISO date.
- `dea_number` is optional but required for any later attempt to submit a US Schedule II–V prescription (see § 11.2).
- `npi` is optional; recommended for US prescribers.

11.1.2 GET /partner/prescribers

List all prescribers for the partner.

Query parameters:

Param	Description
`status`	Filter by `active` or `inactive`.
`limit`	1–200, default 50.

Response (200 OK):

```
{
  "prescribers": [ /* serialized prescriber objects */ ],
  "pagination": { "has_more": false, "next_cursor": null }
}
```

11.1.3 GET /partner/prescribers/:prescriber_id

Returns the single prescriber. Returns `404 PRESCRIBER_NOT_FOUND` if the prescriber does not exist or belongs to another partner.

11.1.4 POST /partner/prescribers/:prescriber_id/deactivate

Sets `status: "inactive"`. Subsequent `POST /partner/prescriptions` calls referencing this prescriber will return `422 PRESCRIBER_INACTIVE`. Re-activation is currently a manual operation — contact Stealth Health support.

11.1.5 POST /partner/prescriptions

Submit a pre-signed prescription. The handler creates the appointment, persists the prescription, and triggers the existing fulfillment pipeline (RxVortex / Wells / pharmacy email / Airtable / patient messaging).

Request shape — common fields:

```
{
  "prescriber_id": "pres_a1b2c3d4e5f6",
  "partner_reference": "PRX-12345",
  "notes": "Optional free-form note for our pharmacy team.",
  "medications": [
    { "code": "MED-TRT-CREAM", "quantity": 30, "dosage_instructions": "Apply 1g daily",
"repeats": 5 }
  ]
}
```

The patient is identified via **exactly one** of:

(a) Existing patient via `patient_id` — patient must already be associated with the partner via a referral.

```
{ "patient_id": "pat_8f2e9a1b", "...": "..." }
```

(b) Existing referral via `referral_id` — patient is resolved through the referral's `patient_profile_id`.

```
{ "referral_id": "ref_abc123xyz", "...": "..." }
```

(c) New patient inline via `inline_patient` — Stealth Health creates a new patient profile + a synthetic referral with `source: "partner_prescriber_inline"`, and proceeds.

```
{
  "inline_patient": {
    "first_name": "Pat",
    "last_name": "Inline",
    "email": "pat.inline@example.com",
    "phone": "+15125550199",
    "dob": "1990-05-10",
    "gender": "male",
    "address": {
      "street": "1 Main St",
      "city": "Austin",
      "jurisdiction": "TX",
      "postal_code": "73301",
      "country": "US"
    },
    "shipping_address": {
      "street": "1 Main St",
      "city": "Austin",
      "jurisdiction": "TX",
      "postal_code": "73301",
      "country": "US"
    }
  },
  "...": "..."
}
```

Response (201 Created):

```
{
  "appointment_id": "appt_partner_1714060800000_a1b2c3d4",
  "prescription_id": "partner_1714060800000",
  "rx_id": "PRX-1A2B3C4D5E6F",
  "referral_id": "ref_3a4b5c6d7e8f",
  "patient_id": "partner_inline_test_partner_pat_inline_example_com",
  "prescriber_id": "pres_a1b2c3d4e5f6",
  "fulfillment": { "status": "queued" },
  "medications": [
    {
      "code": "MED-TRT-CREAM",
      "product_name": "TRT Cream 200mg/mL",
      "quantity": 30,
      "dosage_instructions": "Apply 1g daily",
      "repeats": 5,
      "schedule": null,
      "pharmacy_location_id": "trt-pharmacy-us"
    }
  ],
  "created_at": "2026-04-23T12:01:00Z"
}
```


Validation order (each step short-circuits with the listed error code on failure):

1. **Auth** — clinical-tier partner key required, else `403 CLINICAL_ACCESS_REQUIRED`.
2. **Schema** — `prescriber_id`, non-empty `medications[]`, exactly one patient resolution mode. See § 15 for the full code matrix.
3. **Prescriber** — exists and belongs to partner (`PRESCRIBER_NOT_FOUND`), and is `active` (`PRESCRIBER_INACTIVE`).
4. **Patient resolution** — `patient_id` / `referral_id` lookups must be partner-owned (`PATIENT_NOT_FOUND` / `REFERRAL_NOT_FOUND`); `inline_patient` shape is validated (`INLINE_PATIENT_INVALID`).
5. **License match** — prescriber must hold an unexpired license matching the patient's `country` + `jurisdiction` (`PRESCRIBER_LICENSE_MISMATCH`).
6. **Medication catalog** — each `code` must exist in Stealth Health's catalog and be permitted in the patient's jurisdiction (`MEDICATION_NOT_FOUND`, `MEDICATION_NOT_AVAILABLE_IN_JURISDICTION`).
7. **Controlled substance** — see § 11.2.

11.2 Controlled Substances by Jurisdiction

Schedule II–V medications carry extra requirements based on the patient's country.

Jurisdiction	Requirement	On failure
US (any state)	Prescriber must (a) hold a US license in the patient's state and (b) carry a `dea_number` on their prescriber record.	`422 CONTROLLED_SUBSTANCE_REQUIRES_DEA`
Canada (any province)	Provincial license in the patient's province is sufficient. No DEA required.	`422 PRESCRIBER_LICENSE_MISMATCH` if license missing/expired.
Other countries	Not supported in v1.	`422 PRESCRIBER_LICENSE_MISMATCH`.

v2 roadmap for controlled substances:

- DIN-level gating for Canadian narcotics + benzodiazepines.
- EPCS-compliant electronic signature capture (replaces v1's pre-signed PDF model).
- Real-time PDMP checks against state databases.

---

12. API Reference — Lab Orders & Requisitions

Clinical-tier partners can read the lab orders placed for their patients, download the signed requisition PDF (the form the patient takes to a draw site or that ships with an at-home test kit), and read the final biomarker results once the lab releases them.

Lab orders are placed inside Stealth Health by the reviewing clinician — partners do not currently `POST` lab orders directly. If your prescriber-partner flow needs to attach a lab panel to a `POST /partner/prescriptions` submission, see the `lab_orders[]` block on the Partner-Submitted Prescription Object.

> ***Status:*** *Generally available in sandbox; production rollout gated on per-partner BAA addendum acknowledging the additional PHI surface (panel selection + biomarker values). Email* `partners@stealth.health` *to enable in production.*

Resource model. A lab order belongs to an `appointment_id`, which belongs to a `patient_id`. The order routes through one of our diagnostic-lab integrations (Junction Health → Quest, Labcorp, Sonora Quest, BioReference, or CRL for at-home test kits). The requisition PDF is published once the lab assigns a sample identifier; biomarker results land later, in one or two waves (`partial` → `final`).

12.1 GET /partner/patients/:patient_id/lab-orders

List all lab orders placed for a patient.

Query Parameters:

Param	Type	Default	Description
`status`	string	(all)	Filter: `ordered`, `awaiting_collection`, `in_transit_to_lab`, `at_lab`, `partial_results`, `completed`, `cancelled`, `exception`.
`collection_method`	string	(all)	Filter: `walk_in_test`, `at_home_phlebotomy`, `testkit`, `on_site_collection`.
`appointment_id`	string	—	Restrict to orders for one appointment.
`created_after`	ISO 8601	—	Orders placed after this timestamp.
`limit`	integer	50	Max results per page (1–200).
`cursor`	string	—	Pagination cursor.

Response (200 OK):


```
{
  "patient_id": "pat_8f2e9a1b",
  "lab_orders": [
    {
      "lab_order_id": "lab_3f8a2b1c9e4d",
      "appointment_id": "apt_c4d5e6f7",
      "referral_id": "ref_abc123xyz",
      "status": "completed",
      "detailed_status": "completed.results.final",
      "collection_method": "walk_in_test",
      "provider": "quest",
      "tests": [
        {
          "panel_id": "panel-trt-baseline",
          "name": "TRT Baseline Panel",
          "method": "venipuncture",
          "sample_type": "serum",
          "price_cents": 14900
        }
      ],
      "ordered_at": "2026-04-12T15:30:00Z",
      "collected_at": "2026-04-14T09:05:00Z",
      "resulted_at": "2026-04-15T18:22:00Z",
      "requisition_available": true,
      "results_available": true
    }
  ],
  "pagination": { "has_more": false, "next_cursor": null }
}
```

12.2 GET /partner/lab-orders/:lab_order_id

Retrieve a single lab order, including the full event timeline and (if a Patient Service Center appointment was booked) the PSC details.

Response (200 OK):

```
{
  "lab_order_id": "lab_3f8a2b1c9e4d",
  "patient_id": "pat_8f2e9a1b",
  "appointment_id": "apt_c4d5e6f7",
  "referral_id": "ref_abc123xyz",
  "partner_reference": "cust_12345",
  "status": "completed",
  "detailed_status": "completed.results.final",
  "collection_method": "walk_in_test",
  "provider": "quest",
  "lab_test_id": "lt_7a2c4f1e",
  "tests": [
    {
      "panel_id": "panel-trt-baseline",
      "name": "TRT Baseline Panel",
      "method": "venipuncture",
      "sample_type": "serum",
      "markers": [
        { "name": "Total Testosterone", "code": "TST", "loinc": "2986-8" },
        { "name": "Free Testosterone", "code": "FT", "loinc": "2991-8" },
        { "name": "Sex Hormone Binding Globulin", "code": "SHBG", "loinc": "13967-5" }
      ],
      "price_cents": 14900
    }
  ],
  "events": [
    { "status": "ordered", "occurred_at": "2026-04-12T15:30:00Z" },
    { "status": "awaiting_collection", "occurred_at": "2026-04-12T15:30:05Z" },
    { "status": "requisition_created", "occurred_at": "2026-04-12T15:31:11Z" },
    { "status": "sample_collected", "occurred_at": "2026-04-14T09:05:00Z" },
    { "status": "at_lab", "occurred_at": "2026-04-14T17:48:00Z" },
    { "status": "results.partial", "occurred_at": "2026-04-15T09:12:00Z" },
    { "status": "results.final", "occurred_at": "2026-04-15T18:22:00Z" }
  ],
  "psc_appointment": {
    "appointment_id": "psc_apt_b7e1",
    "provider": "quest",
    "status": "completed",
    "start_at": "2026-04-14T09:00:00Z",
    "iana_timezone": "America/Chicago",
    "location_name": "Quest Diagnostics — Austin Anderson Mill",
    "location_address": "13740 Research Blvd, Austin, TX 78750"
  },
  "payment": {
    "model": "patient_pays",
    "status": "paid",
    "amount_cents": 14900,
    "currency": "USD",
    "stripe_payment_intent_id": "pi_3ghi789jkl012"
  },
  "requisition_available": true,
  "results_available": true,
  "created_at": "2026-04-12T15:30:00Z",
```

```
  "updated_at": "2026-04-15T18:22:00Z"
}
```

Notes:

- `psc_appointment` is present only for `collection_method: "walk_in_test"` orders that have been booked through one of the lab's Patient Service Center networks (Quest, Sonora Quest today).
- `payment.model` is one of `patient_pays`, `doctor_pays`, or `included` (rolled into the enrollment fee). `included` orders do not generate a separate `transaction.*` webhook.

12.3 GET /partner/lab-orders/:lab_order_id/requisition

Returns the lab's requisition PDF inline (response `Content-Type: application/pdf`, `Content-Disposition: inline; filename="requisition-<lab_order_id>.pdf"`). The response body is the raw PDF bytes; nothing is persisted on Stealth Health's side beyond what the lab partner publishes through Junction.

```
curl -L \
  -H "X-Partner-ID: ptr_yourpartner" \
  -H "X-Api-Key: sk_live_..." \
  -H "X-Access-Tier: clinical" \
  "https://api.stealth.health/partner/lab-orders/lab_3f8a2b1c9e4d/requisition" \
  -o lab-requisition.pdf
```

Each request emits an audit row (`action: partner.lab_requisition.downloaded`) capturing the requesting `partner_id`, the `lab_order_id`, the parent `appointment_id`, and the byte count served. The PDF itself carries patient demographics + the panels ordered — treat it as PHI in your storage layer.

> ***Roadmap (v2): signed-URL mode.*** *A `?response=signed_url` mode that mints a 15-minute signed download URL (and a 90-day object-retention policy) is reserved for v2 once a partner asks for it. The current inline path is the secure default — there is no persistent server-side copy of the PDF to leak. If you need pre-signed downloads, email `partners@stealth.health`.*

Errors:

Code	HTTP	Meaning
`LAB_ORDER_NOT_FOUND`	404	Order does not exist or does not belong to a partner-owned patient.
`LAB_ORDER_ACCESS_REQUIRED`	403	Partner is in production without the lab-orders BAA addendum. Sandbox bypasses this check.
`REQUISITION_NOT_READY`	404	The lab has not published the requisition yet. Typical wait is 30–120 seconds after `ordered`. The `lab_order.requisition_ready` webhook fires when the PDF is available — prefer it to polling.
`REQUISITION_EXPIRED`	410	Requisition PDFs are retained for 90 days after `completed`. Older orders return `410` instead of serving stale PDFs.
`JUNCTION_UPSTREAM_ERROR`	502	Junction returned an error or timed out. Retry with exponential backoff.

12.4 GET /partner/lab-orders/:lab_order_id/results

Returns the structured biomarker results once the lab has released them. Available only when `results_available: true`.

Response (200 OK):

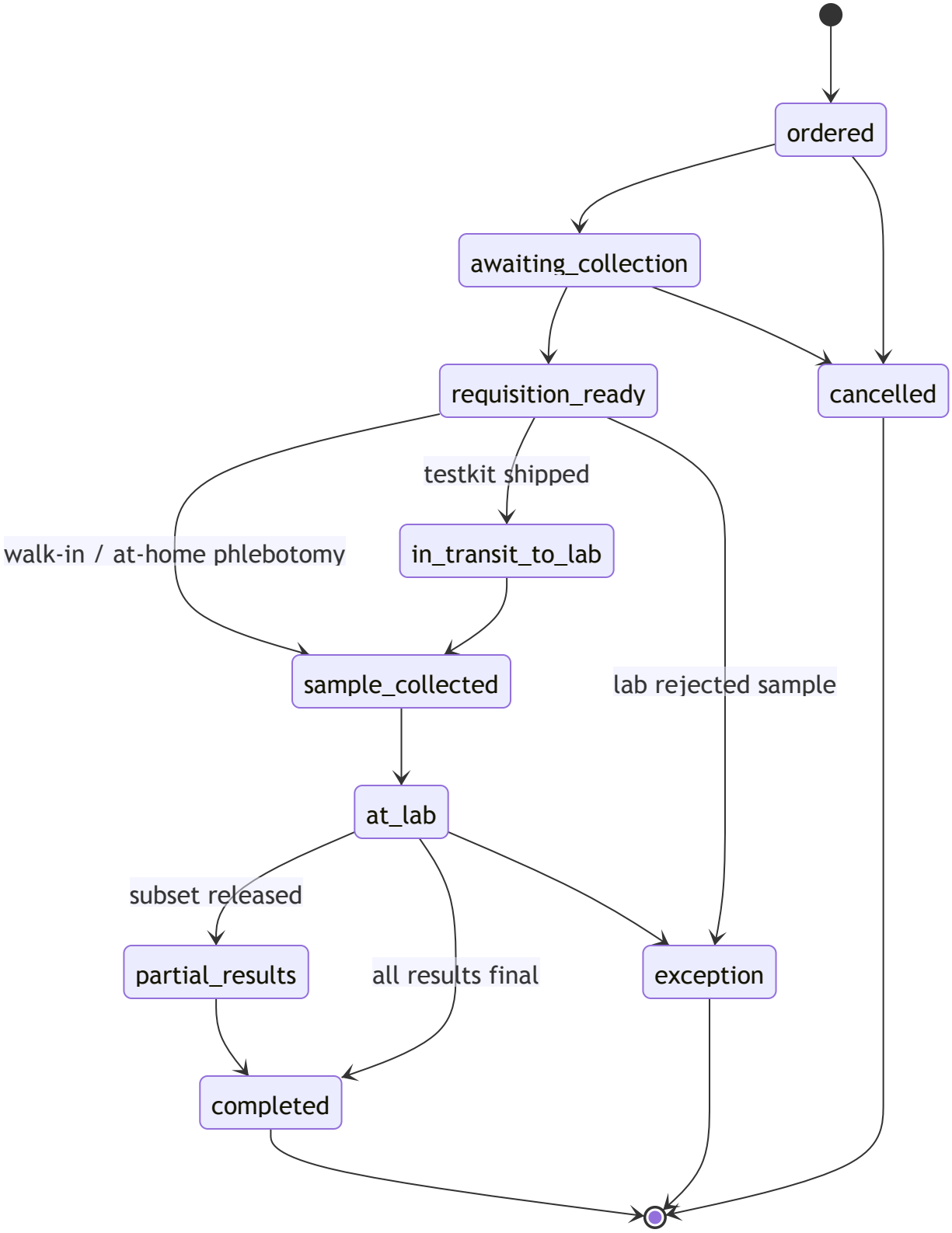
```
{
  "lab_order_id": "lab_3f8a2b1c9e4d",
  "patient_id": "pat_8f2e9a1b",
  "result_status": "final",
  "resulted_at": "2026-04-15T18:22:00Z",
  "results": [
    {
      "name": "Total Testosterone",
      "slug": "total-testosterone",
      "value": 412,
      "result": "412",
      "unit": "ng/dL",
      "type": "numeric",
      "loinc": "2986-8",
      "reference_range": "264–916",
      "min_range_value": 264,
      "max_range_value": 916,
      "is_above_max_range": false,
      "is_below_min_range": false,
      "interpretation": "normal",
      "notes": null,
      "timestamp": "2026-04-15T18:22:00Z"
    },
    {
      "name": "Free Testosterone",
      "slug": "free-testosterone",
      "value": 6.1,
      "result": "6.1",
      "unit": "pg/mL",
      "type": "numeric",
      "loinc": "2991-8",
      "reference_range": "8.7–25.1",
      "min_range_value": 8.7,
      "max_range_value": 25.1,
      "is_above_max_range": false,
      "is_below_min_range": true,
      "interpretation": "abnormal",
      "notes": null,
      "timestamp": "2026-04-15T18:22:00Z"
    }
  ],
  "missing_results": [
    {
      "name": "Sex Hormone Binding Globulin",
      "slug": "shbg",
      "loinc": "13967-5",
      "inferred_failure_type": "sample_quality",
      "note": "Lab flagged the SHBG aliquot as hemolyzed; redraw recommended."
    }
  ],
  "results_pdf_url": "https://storage.googleapis.com/stealth-health-prod-lab-
```

```
results/lab_3f8a2b1c9e4d.pdf?X-Goog-Algorithm=...&X-Goog-Expires=900"
}
```

Notes:

- `result_status` is `"partial"` while some markers are still pending and `"final"` once every ordered marker has either a value or a `missing_results[]` entry. The endpoint will return `result_status: "partial"` with the subset already available; clients should re-poll (or wait for the `lab_order.results_updated` / `lab_order.completed` webhooks) for the rest.
- `interpretation` is one of `normal`, `abnormal`, `critical`. The interpretation is the lab's, not Stealth Health's — we surface it verbatim.
- The PDF at `results_pdf_url` is the lab's full results document with reference ranges and the lab director's signature.
- `missing_results[].inferred_failure_type` is a coarse bucket — common values are `sample_quality`, `sample_volume`, `lab_assay_failure`, `partial_release`. Treat it as a hint, not a contract.

12.5 Lab order status lifecycle



`detailed_status` (returned on the order object) is a dotted-path refinement of `status` — for example `awaiting_collection.psc_scheduled`, `at_lab.results.pending`, `completed.results.partial`, `exception.sample_rejected.hemolyzed`. Partners SHOULD branch on `status` for top-level flow and treat `detailed_status` as a free-form descriptor for audit / display.

12A. API Reference — Patient Documents

Attach a **binary clinical document** (a lab PDF, a scanned requisition, an ID photo, etc.) to a patient you own. This is the channel to use when you have a *file* to deliver — `POST /partner/appointments` only accepts structured + free-text intake answers, so lab **values** can ride along as `recent_labs` text but the **source PDF** has to come through here.

All three endpoints are **clinical-tier only**. Sandbox partners may upload freely; **production** access requires the **documents BAA addendum** (`features.documents` on your partner account — set by Stealth after countersignature, no self-serve path). Without it, production calls return `403 DOCUMENT_ACCESS_REQUIRED`. The partner must already own the patient (an existing referral links the patient to your account), exactly like the lab-orders and messages endpoints.

Every upload and every read emits a HIPAA audit row — uploaded files are PHI artifacts.

12A.1 POST /partner/patients/:patient_id/documents

Upload is **base64-in-JSON** (the partner API is JSON-only; a presigned multipart mode is the planned v2 for large files). Decoded size is capped at **10 MiB**. Accepted `content_type` values: `application/pdf`, `image/png`, `image/jpeg`, `image/heic`, `image/tiff`.

Request:

```
curl -X POST \
  "https://api.stealth.health/partner/patients/pat_8f2e9a1b/documents" \
  -H "X-Partner-ID: ptr_yourco" \
  -H "X-Api-Key: sk_live_..." \
  -H "Content-Type: application/json" \
  -d '{
    "filename": "cbc-2026-05.pdf",
    "content_type": "application/pdf",
    "content_base64": "JVBERi0xLjQg..."   ,
    "document_type": "lab_result",
    "description": "CBC + hormone panel, drawn 2026-05",
    "appointment_id": "appt_partner_intake_1780273459899_94c6a760"
  }'
```

Field	Required	Notes
`filename`	no	Sanitized server-side; defaults to `document`.
`content_type`	**yes**	Must be in the allow-list above.
`content_base64`	**yes**	Base64 (a `data:` URL prefix is stripped if present). Decodes to ≤ 10 MiB.
`document_type`	no	One of `lab_result`, `requisition`, `identification`, `insurance`, `clinical_note`, `imaging`, `other`. Defaults to `other`.
`description`	no	≤ 500 chars.
`appointment_id`	no	If supplied, must be an appointment on a referral you own for this patient; otherwise defaults to the patient's referral appointment (or null).

Response (201 Created):


```
{
  "document_id": "doc_4a9c1f2e8b7d6c5a3f0e1d2c",
  "patient_id": "pat_8f2e9a1b",
  "appointment_id": "appt_partner_intake_1780273459899_94c6a760",
  "document_type": "lab_result",
  "filename": "cbc-2026-05.pdf",
  "content_type": "application/pdf",
  "size_bytes": 248173,
  "sha256": "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08",
  "description": "CBC + hormone panel, drawn 2026-05",
  "status": "stored",
  "created_at": "2026-06-02T17:04:00Z",
  "download_endpoint": "/partner/documents/doc_4a9c1f2e8b7d6c5a3f0e1d2c"
}
```


12A.2 GET /partner/patients/:patient_id/documents

Lists the documents you've uploaded for a patient (newest first). Returns the same projection as the upload response (metadata only — no bytes).

12A.3 GET /partner/documents/:document_id

Streams the stored file back inline (`Content-Type` + `Content-Disposition` match the upload). 404 (never 403) if the document belongs to another partner, so the existence of other partners' documents is never leaked.

12A.4 Error codes

Code	HTTP	Meaning
`DOCUMENT_ACCESS_REQUIRED`	403	Production partner without the documents BAA addendum. Sandbox bypasses this check.
`UNSUPPORTED_CONTENT_TYPE`	415	`content_type` is not in the allow-list.
`DOCUMENT_TOO_LARGE`	413	Decoded payload exceeds 10 MiB.
`DOCUMENT_INVALID`	400	Missing/invalid body, payload, `document_type`, or an `appointment_id` you don't own.
`DOCUMENT_NOT_FOUND`	404	No such document for this partner.
`PATIENT_NOT_FOUND`	404	Patient not found or not associated with this partner.

12B. API Reference — Wearables Data

Stealth Health patients can connect wearable devices (Fitbit, Oura, Garmin, WHOOP, Apple Health, Health Connect, Dexcom, and the rest of the Junction Sense catalog) inside the patient portal. The partner API exposes a **read-only** view of that data for patients you own — device connection status, per-day summaries (sleep, activity, body, workouts, heart rate, blood pressure, glucose), and intraday timeseries — plus three `wearable.connection.*` webhook events for connection lifecycle.

Access gate. Wearables telemetry is a distinct PHI class (continuous biometric monitoring), so it follows the same pattern as lab orders and documents:

- **Clinical tier only.** Referral-tier keys get `403 CLINICAL_ACCESS_REQUIRED`.
- **Production requires the wearables BAA addendum.** Until it's countersigned and `features.wearables` is enabled for your key, production calls return `403 WEARABLES_ACCESS_REQUIRED`. Email `partners@stealth.health` to start the addendum.
- **Sandbox is open** to every clinical-tier partner — connect demo providers via the patient-portal sandbox to generate synthetic data.
- **Patient scope.** All three endpoints are scoped to a `patient_id` you own (an existing referral links the patient to your key). Unknown or unowned patients return `404 PATIENT_NOT_FOUND` — never a 403 — so the existence of other partners' patients is not leakable.
- **Every read is audited.** Each successful call emits a per-access PHI audit row on the Stealth side, in addition to the partner-console access log.

Data freshness. Responses are served from Stealth's mirror of the Junction feed (the same store the reviewing clinician sees). Providers typically push within minutes of a device sync, but the device itself may sync to its vendor cloud on a slower cadence (hours for some ring/watch firmware). Treat timestamps, not response recency, as the source of truth.

12B.1 GET /partner/patients/:patient_id/wearables/connections

Lists the patient's device connections (one row per provider), including providers that were connected and later revoked.

```
curl -s "$BASE/partner/patients/$PATIENT_ID/wearables/connections" \
  -H "Authorization: Bearer $PARTNER_API_KEY" \
  -H "X-Access-Tier: clinical"
```

Response `200`:

```
{
  "patient_id": "pat_8f2e9a1b",
  "connections": [
    {
      "provider_slug": "oura",
      "provider_name": "Oura",
      "auth_type": "oauth",
      "status": "connected",
      "source": "web",
      "connected_at": "2026-06-01T12:00:00.000Z",
      "last_sync_at": "2026-07-13T09:30:00.000Z",
      "last_error_at": null,
      "error_message": null,
      "created_at": "2026-06-01T12:00:00.000Z",
      "updated_at": "2026-07-13T09:30:00.000Z"
    }
  ]
}
```

`status` is one of `connected`, `syncing`, `stale` (no provider event in 36+ hours), `error` (provider auth broke — the patient needs to reconnect), or `revoked`. `source` distinguishes web OAuth connections from native `apple_health_kit` / `health_connect` connections (`web`, `native_ios`, `native_android`, or `null` for legacy rows).

12B.2 GET /partner/patients/:patient_id/wearables/summaries

Per-day rollups across all the patient's connected providers — one object per calendar day, with a block per data type. This is the endpoint to build adherence/trend views on; it's cheap for both sides and stable across

providers.

Query param	Required	Notes
`start_date`	yes	`YYYY-MM-DD` (UTC calendar dates)
`end_date`	no	Inclusive; defaults to today (UTC). Future dates are clamped to today.

The window (`start_date` → `end_date`) must be **90 days or less** — larger requests return `400 WEARABLE_DATE_INVALID` . Page by consecutive windows for more history.

```
curl -s "$BASE/partner/patients/$PATIENT_ID/wearables/summaries?start_date=2026-07-
01&end_date=2026-07-07" \
  -H "Authorization: Bearer $PARTNER_API_KEY" \
  -H "X-Access-Tier: clinical"
```

Response `200` :

```
{
  "patient_id": "pat_8f2e9a1b",
  "start_date": "2026-07-01",
  "end_date": "2026-07-07",
  "summaries": [
    {
      "date": "2026-07-01",
      "sleep": { "provider_slug": "oura", "total_minutes": 452, "efficiency": 0.93, "stages": {
"deep": 80, "rem": 110 } },
      "activity": { "provider_slug": "oura", "steps": 10412, "calories_active": 512,
"calories_basal": 1650, "distance_meters": 8100, "floors_climbed": 12 },
      "body": null,
      "workouts": [
        { "provider_slug": "strava", "workout_type": "run", "start_at": "2026-07-01T11:00:00Z",
"end_at": "2026-07-01T11:45:00Z", "duration_minutes": 45, "calories_active": 480,
"average_heart_rate": 152 }
      ],
      "menstrual_cycle": null,
      "heart_rate": { "provider_slug": "oura", "resting_bpm": 58, "average_bpm": 72, "min_bpm":
51, "max_bpm": 148 },
      "blood_pressure_daily": null,
      "glucose_daily": null,
      "updated_at": "2026-07-02T04:30:00.000Z"
    }
  ]
}
```

Every block is present on every day (as `null` when that data type has no reading), so the response shape is stable regardless of which devices the patient owns. Days with no data at all are simply absent from

`summaries`.

12B.3 GET /partner/patients/:patient_id/wearables/timeseries

Intraday samples for one metric. Use this for high-resolution views (CGM curves, workout heart-rate traces); use § 12B.2 for anything daily.

Query param	Required	Notes
`metric`	yes	Junction metric slug: `heart_rate`, `resting_heart_rate`, `hrv`, `blood_pressure`, `blood_oxygen`, `respiratory_rate`, `body_temperature`, `glucose`, `weight`, `body_fat`, `steps`, `calories_active`, `calories_basal`, `distance`, `floors_climbed`, `vo2_max`, `stress_level`, `mindfulness_minutes`
`start_date`	yes	`YYYY-MM-DD` (UTC)
`end_date`	no	Inclusive; defaults to today (UTC)
`limit`	no	Samples per page. Default 1000, max 5000.
`cursor`	no	`next_cursor` from the previous page

The window must be **31 days or less** (intraday metrics are high-cardinality — CGMs emit hundreds of samples a day). An unknown but well-formed metric returns an empty `samples` array rather than an error.

```
curl -s "$BASE/partner/patients/$PATIENT_ID/wearables/timeseries?
metric=glucose&start_date=2026-07-10&limit=1000" \
  -H "Authorization: Bearer $PARTNER_API_KEY" \
  -H "X-Access-Tier: clinical"
```

Response `200`:

```
{
  "patient_id": "pat_8f2e9a1b",
  "metric": "glucose",
  "start_date": "2026-07-10",
  "end_date": "2026-07-14",
  "samples": [
    {
      "timestamp": "2026-07-10T00:03:12.000Z",
      "value": 104,
      "unit": "mg/dL",
      "type": "automatic",
      "provider_slug": "dexcom_v3",
      "source_device": "Dexcom G7"
    }
  ],
  "pagination": { "has_more": true, "next_cursor": "2026-07-10T06:41:55.000Z" }
}
```


`blood_pressure` samples additionally carry `systolic` / `diastolic` fields. Samples are returned oldest-first; `next_cursor` is the timestamp of the last sample on the page — pass it back as `cursor` to continue.

12B.4 Webhook events — `wearable.connection.*`

With `features.wearables` enabled, your webhook endpoint also receives connection lifecycle events (standard clinical envelope — `patient_id` at the top level, event fields under `data.*`):

Event	Trigger	`data` includes
`wearable.connection.created`	Patient connected a provider	`provider_slug`, `status`
`wearable.connection.revoked`	Patient (or the provider) deauthorized the connection	`provider_slug`, `status`
`wearable.connection.error`	Provider auth broke — the patient needs to reconnect	`provider_slug`, `status`

No data-drop events are emitted — a CGM would generate hundreds of webhooks per patient per day. Poll § 12B.2 on your own cadence (daily is plenty for summary data) and treat the connection events as the signal for when polling is worthwhile. As with the `lab_order.*` family, the webhook payload never carries measurements — data reads always go through the audited HTTP endpoints. Token refreshes (`connection.refreshed` upstream) are intentionally not forwarded.

12B.5 Error codes

Code	HTTP	Meaning
`WEARABLES_ACCESS_REQUIRED`	403	Production partner without the wearables BAA addendum. Sandbox bypasses this check.
`WEARABLE_DATE_INVALID`	400	Missing/malformed `start_date` / `end_date`, `end_date` before `start_date`, or the window exceeds the ceiling (90 days for summaries, 31 for timeseries).
`WEARABLE_METRIC_INVALID`	400	`metric` is missing or not a lowercase snake_case slug.
`PATIENT_NOT_FOUND`	404	Patient not found or not associated with this partner.

---

13. Webhook Events

Clinical Partner webhooks include the same events as the Referral Tier (see Referral Tier, Section 7) but with **expanded payloads** that include PHI.

13.1 Expanded Webhook Payload


```
{
  "event_id": "evt_1a2b3c4d",
  "event_type": "referral.approved",
  "referral_id": "ref_abc123xyz",
  "partner_reference": "cust_12345",
  "patient_id": "pat_8f2e9a1b",
  "appointment_id": "apt_c4d5e6f7",
  "data": {
    "status": "approved",
    "product_category": "trt-cream",
    "occurred_at": "2026-03-03T09:30:00Z",
    "patient": {
      "first_name": "John",
      "last_name": "Doe",
      "email": "john.doe@example.com"
    },
    "prescription": {
      "rx_id": "RX-2026-0302-001",
      "medications": [
        {
          "name": "Testosterone Cream 200mg/mL",
          "dosage": "1mL applied topically daily",
          "quantity": 1,
          "repeats": 3
        }
      ],
      "prescriber": "Dr. Smith",
      "signed_at": "2026-03-03T09:30:00Z"
    }
  },
  "metadata": {
    "campaign": "spring-2026"
  },
  "created_at": "2026-03-03T09:30:01Z"
}
```


13.2 Additional Clinical Events

In addition to all Referral Tier events, Clinical Partners also receive:

Event	Trigger	`data` includes
`patient.created`	Patient profile created after enrollment	`patient` (full pro
`patient.updated`	Patient updates their profile	`patient` (change
`appointment.intake_completed`	Intake questionnaire submitted	`appointment_id`
`appointment.prescription_signed`	A Stealth-employed (or partner) physician signed the prescription for a partner-owned appointment. Emitted from **all** sign paths: the doctor portal queue (`submitDoctorPrescription`), the standalone Rx flow (`submitStandalonePrescription`), the legacy JotForm pipeline, **and** the partner-submitted-Rx path (`POST /partner/prescriptions` — same handler fires both `prescription.received` and `appointment.prescription_signed` in that flow). Always paired with `referral.approved` at the same instant.	`appointment_id` `rx_id, medicati`
`transaction.succeeded`	Payment successfully processed. For `authorize_only` appointments this fires at **capture** time (clinical approval), with `transaction.capture_of_authorization: true` .	`transaction` (ful
`transaction.refunded`	Payment refunded	`transaction` , `re` `reason`
`transaction.authorization_released`	An `authorize_only` hold was voided because the reviewing physician marked the patient clinically ineligible. The patient was never charged (this is a release of held funds, not a refund). Paired with `referral.denied` / `prescription.rejected` at the same instant.	`transaction: {` `type, status:` `"authorization_` `}` , `reason`
`prescription.received`	Partner-submitted prescription accepted by `POST /partner/prescriptions` and queued for fulfillment	`appointment_id` `rx_id` , `prescrib` `medications[]`
`prescription.rejected`	Either (a) a partner-submitted prescription failed async re-validation, or (b) a Stealth physician declined to sign for a partner-owned appointment via the doctor portal (`rejectAppointment`). In path (b) this event is **always paired with** `referral.denied` at the same instant; partners can listen to either	`appointment_id?` `error_code` , `err`

	or both. `prescription_id` is `null` in path (b). See § 13.2.2 for the `error_code` enum.	
`messageThread.updated`	A clinician or patient posted a message into a thread linked to a partner-owned appointment. Body is **not** included — partner re-fetches via § 9.1.	`thread_id`, `appo` `patient_id`, `las` `sender_type`, `se`
`lab_order.created`	Clinician placed a lab order on a partner-owned appointment.	`data.lab_order_` `data.collectior` `data.provider`,
`lab_order.requisition_ready`	Lab has published the requisition PDF and it is downloadable via § 12.3.	`data.lab_order_` `data.requisitic` (reserved for v2 si in v1)
`lab_order.sample_collected`	Sample collected (walk-in, at-home phleb, or testkit registration).	`data.lab_order_` `data.collected_` `data.collectior`
`lab_order.results_updated`	Lab released results (partial or final). Biomarker values are NOT in the payload — partner re-fetches via § 12.4.	`data.lab_order_` `data.result_sta` `final`), `data.re`
`lab_order.completed`	All ordered markers final OR all remaining markers settled into `missing_results[]`.	`data.lab_order_` `data.resulted_a`
`lab_order.cancelled`	Order cancelled before sample collection (clinician revoked, patient cancelled, payment failed).	`data.lab_order_` `data.cancellati`
`lab_order.exception`	Lab rejected the sample or the order entered an unrecoverable error state (e.g. `exception.sample_rejected.hemolyzed`).	`data.lab_order_` `data.detailed_s` `data.exception_`
`wearable.connection.created`	Patient connected a wearable provider. Requires `features.wearables` — see § 12B.4.	`data.provider_s`
`wearable.connection.revoked`	Patient (or the provider) deauthorized a wearable connection. Requires `features.wearables`.	`data.provider_s`
`wearable.connection.error`	Wearable provider auth broke — the patient needs to reconnect. Requires `features.wearables`.	`data.provider_s`

13.2.1 `appointment.prescription_signed` payload

The medication list is **deliberately slim** — partner systems should treat the webhook as a "the Rx exists, go fetch" signal rather than a full clinical export. Fields not present in the slim shape (route of administration, dispense quantity unit, control schedule, NDC, prescriber NPI, etc.) are available via the appointment / prescription read endpoints if your tier needs them.


```
{
  "event_id": "evt_8a7c2b4f9d1e",
  "event_type": "appointment.prescription_signed",
  "referral_id": "ref_a1b2c3d4",
  "partner_reference": "PTR-12345",
  "patient_id": "patient_jane_doe_001",
  "appointment_id": "appt_xyz_001",
  "data": {
    "appointment_id": "appt_xyz_001",
    "prescription": {
      "rx_id": "RX-2026-0302-001",
      "medications": [
        { "name": "Testosterone Cypionate 200mg/mL", "dosage": "Inject 0.5mL IM weekly",
"quantity": "1", "repeats": "3" }
      ]
    },
    "product_category": "trt-cream",
    "occurred_at": "2026-05-03T12:00:01Z",
    "status": "approved"
  },
  "metadata": {},
  "created_at": "2026-05-03T12:00:01Z"
}
```


> ***Non-partner appointments do not emit.*** *This event only fires when* `findReferralByAppointmentId(appointmentId)` *returns a referral linked to an* `active` *clinical-tier partner. If a Stealth physician signs an Rx for a direct-to-consumer appointment that was never linked to a partner referral, the partner webhook fan-out is a no-op.*

13.2.2 `prescription.rejected` payload + `error_code` enum


```
{
  "event_id": "evt_9b8c3d5f0e2a",
  "event_type": "prescription.rejected",
  "referral_id": "ref_a1b2c3d4",
  "partner_reference": "PTR-12345",
  "patient_id": "patient_jane_doe_001",
  "appointment_id": "appt_xyz_001",
  "data": {
    "appointment_id": "appt_xyz_001",
    "prescription_id": null,
    "error_code": "MEDICAL_CONTRAINDICATION",
    "error_message": "Patient has a known medical contraindication for this therapy.",
    "product_category": "trt-cream",
    "occurred_at": "2026-05-03T12:00:01Z",
    "status": "denied"
  },
  "metadata": {},
  "created_at": "2026-05-03T12:00:01Z"
}
```


`error_code`	Source	Meaning
`MEDICAL_CONTRAINDICATION`	Stealth-doctor reject path	Reviewing physician identified a contraindication in the patient's intake or history. Detected from `rejectionReason` keyword `contraindic*`.
`INCOMPLETE_INFORMATION`	Stealth-doctor reject path	The clinical record is missing data required to make a safe prescribing decision. Detected from `rejectionReason` keyword `incomplete*`.
`NOT_A_CANDIDATE`	Stealth-doctor reject path	Catch-all denial when neither of the above keywords match. Partners should expect this to be the most common code.
`<re-validation codes>`	Partner-submitted-Rx async re-validation path	Reserved; not surfaced in v1's synchronous `POST /partner/prescriptions` flow. When the async re-validation path lands, expect schedule/license/NPI-specific codes here.

> ***`error_message`* is partner-handled PHI-adjacent free text.** *It carries the physician's `rejectionReason` verbatim. Do not log it at `info` on the partner side; treat it the same way you treat appointment intake notes.*

Payload conventions for the `lab_order.*` family. As with every clinical-tier webhook, `patient_id` and `appointment_id` are stamped at the *top level* of the envelope (next to `event_id` and `event_type`). All event-specific fields — including `lab_order_id` — live under `data.*`. This mirrors the existing `appointment.*` and `prescription.*` shape, so a single signature-verify + envelope-parse routine on your side covers every clinical event type. See the canonical payload below.

```
{
  "event_id": "evt_8a7c2b4f9d1e",
  "event_type": "lab_order.results_updated",
  "referral_id": "ref_a1b2c3d4",
  "partner_reference": "PTR-12345",
  "patient_id": "patient_jane_doe_001",
  "appointment_id": "appt_xyz_001",
  "data": {
    "lab_order_id": "lab_3f8a2b1c9e4d",
    "result_status": "final",
    "resulted_at": "2026-05-03T12:00:00Z",
    "product_category": "lab-panel",
    "occurred_at": "2026-05-03T12:00:01Z",
    "status": null
  },
  "metadata": {},
  "created_at": "2026-05-03T12:00:01Z"
}
```

Lab biomarker values are deliberately omitted from webhook bodies — they are PHI of a different sensitivity class and are gated behind the § 12.4 results endpoint. This keeps webhook payloads safe to log at info level on the partner side without an extra redaction pass.

13.3 Signature Verification & Retry Policy

Same as Referral Tier — see Referral Tier, Sections 7.4–7.5.

---

14. Data Models

14.1 Patient Object

```
{
  "patient_id": "string",
  "partner_reference": "string",
  "first_name": "string",
  "last_name": "string",
  "email": "string",
  "phone": "string",
  "date_of_birth": "string (YYYY-MM-DD)",
  "gender": "string — male | female | other",
  "age": "integer",
  "address": {
    "line1": "string",
    "line2": "string | null",
    "city": "string",
    "state": "string",
    "postal_code": "string",
    "country": "string — US | CA"
  },
  "product_categories": ["string"],
  "status": "string — active | pending | inactive",
  "latest_appointment_id": "string | null",
  "latest_appointment_status": "string | null",
  "total_appointments": "integer",
  "created_at": "ISO 8601",
  "updated_at": "ISO 8601"
}
```

14.2 Appointment Object

```
{
  "appointment_id": "string",
  "patient_id": "string",
  "referral_id": "string",
  "partner_reference": "string",
  "product_category": "string",
  "condition": "string",
  "status": "string — pending_review | approved | denied | completed",
  "submitted_at": "ISO 8601",
  "reviewed_at": "ISO 8601 | null",
  "patient_snapshot": { "...": "Patient object at time of submission" },
  "intake_summary": {
    "total_questions": "integer",
    "completed_questions": "integer",
    "severity_score": "number | null",
    "goals": "string | null"
  },
  "prescription_summary": {
    "status": "string — pending | signed | denied",
    "rx_id": "string | null",
    "medications": [
      {
        "name": "string",
        "generic_name": "string",
        "dosage": "string",
        "quantity": "integer",
        "quantity_unit": "string",
        "repeats": "integer",
        "notes": "string | null"
      }
    ],
    "prescriber": "string",
    "prescriber_license": "string",
    "signed_at": "ISO 8601 | null"
  },
  "fulfillment": {
    "status": "string | null",
    "carrier": "string | null",
    "tracking_number": "string | null",
    "estimated_delivery": "string (YYYY-MM-DD) | null",
    "shipped_at": "ISO 8601 | null",
    "delivered_at": "ISO 8601 | null"
  },
  "payment": {
    "status": "string — due | paid | refunded | not_applicable",
    "amount_cents": "integer | null",
    "currency": "string — USD | CAD",
    "method": "string | null",
    "paid_at": "ISO 8601 | null",
    "stripe_payment_intent_id": "string | null"
```

```
  },
  "created_at": "ISO 8601",
  "updated_at": "ISO 8601"
}
```

14.3 Intake Response Object

```
{
  "patient_id": "string",
  "appointment_id": "string",
  "form_title": "string",
  "submitted_at": "ISO 8601",
  "responses": [
    {
      "question": "string",
      "answer": "string",
      "category": "string — symptoms | medical_history | medications | allergies |
medical_conditions | goals | demographics | other"
    }
  ]
}
```

14.4 Message Thread Object

```
{
  "thread_id": "string",
  "appointment_id": "string | null",
  "subject": "string | null",
  "status": "string — open | awaiting_patient | awaiting_clinician | doctor_reviewed",
  "participants": [
    {
      "type": "string — doctor | patient | system",
      "name": "string",
      "role": "string — Physician | Patient | Care Team"
    }
  ],
  "last_message_at": "ISO 8601",
  "last_message_preview": "string",
  "created_at": "ISO 8601",
  "messages": [
    {
      "message_id": "string",
      "sender_type": "string — doctor | patient | system",
      "sender_name": "string",
      "sender_role": "string",
      "channel": "string — portal",
      "body": "string",
      "attachments": [
        {
          "name": "string",
          "type": "string | null",
          "url": "string"
        }
      ],
      "status": "string — delivered | read",
      "read_at": "ISO 8601 | null",
      "created_at": "ISO 8601"
    }
  ]
}
```

14.5 Transaction Object

```
{
  "transaction_id": "string",
  "patient_id": "string",
  "partner_reference": "string",
  "appointment_id": "string | null",
  "referral_id": "string | null",
  "type": "string — enrollment_payment | subscription_payment | refund | adjustment",
  "status": "string — succeeded | pending | failed | refunded",
  "amount_cents": "integer",
  "currency": "string — USD | CAD",
  "description": "string",
  "line_items": [
    {
      "description": "string",
      "amount_cents": "integer",
      "quantity": "integer"
    }
  ],
  "payment_method": {
    "type": "string — card | bank",
    "brand": "string | null",
    "last4": "string"
  },
  "stripe_payment_intent_id": "string | null",
  "refund": {
    "amount_cents": "integer",
    "reason": "string",
    "refunded_at": "ISO 8601"
  },
  "created_at": "ISO 8601"
}
```

14.6 Prescriber Object

Returned by the `/partner/prescribers` endpoints.

```
{
  "prescriber_id": "string — pres_xxx",
  "status": "string — active | inactive",
  "first_name": "string",
  "last_name": "string",
  "email": "string",
  "npi": "string | null",
  "dea_number": "string | null — required for US Schedule II–V",
  "licenses": [
    {
      "country": "string — US | CA",
      "jurisdiction": "string — state or province code",
      "license_number": "string",
      "expires_at": "string — ISO date (YYYY-MM-DD)"
    }
  ],
  "created_at": "ISO 8601",
  "updated_at": "ISO 8601"
}
```

14.7 Partner-Submitted Prescription Object

Returned by `POST /partner/prescriptions`. The full prescription record is also persisted to `appointments/{appointment_id}/prescriptions/{prescription_id}`.

```
{
  "appointment_id": "string",
  "prescription_id": "string — submission ID, also the subcollection doc id",
  "rx_id": "string — PRX-xxx human-readable Rx number",
  "referral_id": "string",
  "patient_id": "string",
  "prescriber_id": "string",
  "fulfillment": { "status": "string — queued | dispatched | shipped | delivered" },
  "medications": [
    {
      "code": "string",
      "product_name": "string",
      "quantity": "integer",
      "dosage_instructions": "string",
      "repeats": "integer",
      "schedule": "string | null — II | III | IV | V | null",
      "pharmacy_location_id": "string | null"
    }
  ],
  "lab_orders": [
    {
      "panel_id": "string",
      "collection_method": "string — walk_in_test | at_home_phlebotomy | testkit |
on_site_collection",
      "payment_model": "string — patient_pays | doctor_pays | included"
    }
  ],
  "created_at": "ISO 8601"
}
```

14.8 Lab Order Object

Returned by `GET /partner/patients/:id/lab-orders` and `GET /partner/lab-orders/:id`. Mirrors the internal `appointments/{appointment_id}/labOrders/{junctionOrderId}` document.

```
{
  "lab_order_id": "string — lab_xxx",
  "patient_id": "string",
  "appointment_id": "string",
  "referral_id": "string",
  "partner_reference": "string | null",
  "status": "string — ordered | awaiting_collection | requisition_ready | sample_collected |
in_transit_to_lab | at_lab | partial_results | completed | cancelled | exception",
  "detailed_status": "string — dotted-path refinement of status",
  "collection_method": "string — walk_in_test | at_home_phlebotomy | testkit |
on_site_collection",
  "provider": "string — quest | labcorp | sonora_quest | bioreference | crl | ...",
  "lab_test_id": "string — opaque Junction identifier",
  "tests": [
    {
      "panel_id": "string",
      "name": "string",
      "method": "string",
      "sample_type": "string",
      "price_cents": "integer",
      "markers": [
        {
          "name": "string",
          "code": "string",
          "loinc": "string | null"
        }
      ]
    }
  ],
  "events": [
    {
      "status": "string",
      "occurred_at": "ISO 8601"
    }
  ],
  "psc_appointment": {
    "appointment_id": "string | null",
    "provider": "string | null",
    "status": "string — confirmed | pending | reserved | in_progress | completed | cancelled",
    "start_at": "ISO 8601 | null",
    "end_at": "ISO 8601 | null",
    "iana_timezone": "string | null",
    "location_name": "string | null",
    "location_address": "string | null"
  },
  "payment": {
    "model": "string — patient_pays | doctor_pays | included",
    "status": "string — pending | paid | no-charge",
    "amount_cents": "integer | null",
    "currency": "string — USD | CAD",
    "stripe_payment_intent_id": "string | null"
  },
  "ordered_at": "ISO 8601",
```

```
  "collected_at": "ISO 8601 | null",
  "resulted_at": "ISO 8601 | null",
  "requisition_available": "boolean",
  "results_available": "boolean",
  "created_at": "ISO 8601",
  "updated_at": "ISO 8601"
}
```

14.9 Lab Order Results Object

Returned by `GET /partner/lab-orders/:id/results`. See § 12.4 for the full example.

```
{
  "lab_order_id": "string",
  "patient_id": "string",
  "result_status": "string — partial | final",
  "resulted_at": "ISO 8601",
  "results": [
    {
      "name": "string",
      "slug": "string",
      "value": "number | null",
      "result": "string",
      "unit": "string | null",
      "type": "string — numeric | range | comment | coded_value",
      "loinc": "string | null",
      "reference_range": "string | null",
      "min_range_value": "number | null",
      "max_range_value": "number | null",
      "is_above_max_range": "boolean | null",
      "is_below_min_range": "boolean | null",
      "interpretation": "string — normal | abnormal | critical",
      "notes": "string | null",
      "timestamp": "ISO 8601 | null"
    }
  ],
  "missing_results": [
    {
      "name": "string",
      "slug": "string",
      "loinc": "string | null",
      "inferred_failure_type": "string — sample_quality | sample_volume | lab_assay_failure |
partial_release | other",
      "note": "string | null"
    }
  ],
  "results_pdf_url": "string — short-lived signed URL, 15-min TTL"
}
```

15. Error Handling

Error handling is identical to the Referral Tier (see Referral Tier, Section 9), with the following additional error codes:

Code	Description
`PATIENT_NOT_FOUND`	Patient doesn't exist or doesn't belong to this partner
`APPOINTMENT_NOT_FOUND`	Appointment doesn't exist or doesn't belong to a partner patient
`TRANSACTION_NOT_FOUND`	Transaction doesn't exist or doesn't belong to a partner patient
`INTAKE_NOT_AVAILABLE`	Intake responses not yet submitted for this appointment
`QUERY_ERROR`	Internal query failed (usually transient — retry)
`CLINICAL_ACCESS_REQUIRED`	Endpoint requires `X-Access-Tier: clinical` header
`COMPLIANCE_REVIEW_PENDING`	Partner's annual compliance review is overdue — API access suspended
`MISSING_PRESCRIBER_ID`	`POST /partner/prescriptions` body did not include `prescriber_id`
`PRESCRIBER_NOT_FOUND`	Prescriber does not exist or does not belong to this partner
`PRESCRIBER_INACTIVE`	Prescriber has been deactivated; re-activate or register a new one
`PRESCRIBER_NAME_REQUIRED`	`POST /partner/prescribers` body missing `first_name` / `last_name`
`PRESCRIBER_EMAIL_INVALID`	`POST /partner/prescribers` body has malformed `email`
`PRESCRIBER_LICENSE_REQUIRED`	`POST /partner/prescribers` body has empty `licenses[]`
`PRESCRIBER_LICENSE_INVALID`	Invalid `country`, `jurisdiction`, or `license_number` on a license entry
`PRESCRIBER_LICENSE_EXPIRED`	License `expires_at` is in the past at registration time
`PRESCRIBER_LICENSE_MISMATCH`	Prescriber holds no unexpired license matching the patient's `country` + `jurisdiction`
`MEDICATIONS_REQUIRED`	`POST /partner/prescriptions` body has empty `medications[]`
`MEDICATION_INVALID`	A medication entry is missing `code`, `quantity`, or `dosage_instructions`
`MEDICATION_NOT_FOUND`	Medication `code` not found in Stealth Health's catalog (or marked inactive)

`MEDICATION_NOT_AVAILABLE_IN_JURISDICTION`	Medication is not approved for the patient's `country` / `jurisdiction`
`CONTROLLED_SUBSTANCE_REQUIRES_DEA`	US Schedule II–V prescription submitted by a prescriber without a `dea_number`
`PATIENT_REFERRAL_REQUIRED`	`POST /partner/prescriptions` body must include exactly one of `patient_id`, `referral_id`, or `inline_patient`
`INLINE_PATIENT_INVALID`	`inline_patient` block is missing required fields or has malformed `email` / `dob` / `address`
`REFERRAL_NOT_FOUND`	`referral_id` does not exist or does not belong to this partner
`LAB_ORDER_NOT_FOUND`	Lab order does not exist or does not belong to a partner-owned patient
`REQUISITION_NOT_READY`	The lab has not yet published the requisition PDF. Retry after the `lab_order.requisition_ready` webhook fires.
`REQUISITION_EXPIRED`	Requisition PDFs are retained for 90 days after `completed`; older orders return `410`.
`RESULTS_NOT_AVAILABLE`	Lab has not released results yet (still in `at_lab`). Retry after `lab_order.results_updated`.
`LAB_ORDER_ACCESS_REQUIRED`	Partner has not signed the lab-orders BAA addendum. Email `partners@stealth.health` to enable.
`DOCUMENT_ACCESS_REQUIRED`	Partner is in production without the documents BAA addendum (`features.documents`). Sandbox bypasses this check. See § 12A.
`UNSUPPORTED_CONTENT_TYPE`	Document upload `content_type` is not in the allow-list (PDF / PNG / JPEG / HEIC / TIFF).
`DOCUMENT_TOO_LARGE`	Document upload exceeds the 10 MiB decoded-size limit.
`DOCUMENT_INVALID`	Document upload body is missing/invalid (`content_base64`, `document_type`, or an `appointment_id` the partner doesn't own).
`DOCUMENT_NOT_FOUND`	Document does not exist or does not belong to this partner.
`WEARABLES_ACCESS_REQUIRED`	Partner is in production without the wearables BAA addendum (`features.wearables`). Sandbox bypasses this check. See § 12B.

`WEARABLE_DATE_INVALID`	Wearables `start_date` / `end_date` missing, malformed, inverted, or the window exceeds the ceiling (90 days for summaries, 31 for timeseries).
`WEARABLE_METRIC_INVALID`	Wearables timeseries `metric` is missing or not a lowercase snake_case slug.

16. Rate Limits & Environments

Same as the Referral Tier — see Referral Tier, Sections 10–11.

Environment	Base URL
Sandbox	`https://sandbox.stealth.health`
Production	`https://api.stealth.health`

17. Partner Implementation Best Practices

These are the patterns Stealth Health expects to see when reviewing a partner's clinical-tier integration. They are not contractually required (the BAA + § 3.4 cover the contractual minimums), but every successful partner audit has implemented them.

17.1 Credential Storage & Rotation

- **Never check API keys or webhook secrets into source control.** Store them in your platform's secret manager (AWS Secrets Manager, GCP Secret Manager, HashiCorp Vault, Doppler, etc.) and inject at runtime.
- Treat `X-Api-Key` as a **password**: log only the first 8 characters (`sk_live_7f3a…`) when you must reference it in operational tooling.
- **Rotate annually**, or immediately if a workforce member with access leaves. Stealth Health supports zero-downtime rotation: request a new key from `partners@stealth.health` , deploy it everywhere, then call us to invalidate the old key. The old key remains valid until `previous_key_expires_at` (default 7 days).
- Use **separate keys for sandbox and production**. The sandbox key prefix is `sk_test_…` ; never let a sandbox key reach production deployment.
- If you operate multiple internal services, prefer issuing **service-scoped sub-keys** (request from Stealth Health) rather than sharing the root key.

17.2 Webhook Signature Verification

Every webhook delivery includes a `X-Stealth-Signature: sha256=<hex>` header computed as `HMAC-SHA256(webhook_secret, raw_request_body)`. **Verify on every request.** Skipping this lets an attacker spoof prescription / fulfillment events into your system.

```
import crypto from "node:crypto";
import type { Request, Response } from "express";

const WEBHOOK_SECRET = process.env.STEALTH_WEBHOOK_SECRET!;

function verifySignature(rawBody: Buffer, headerValue: string | undefined) {
  if (!headerValue?.startsWith("sha256=")) return false;
  const provided = headerValue.slice("sha256=".length);
  const expected = crypto
    .createHmac("sha256", WEBHOOK_SECRET)
    .update(rawBody)
    .digest("hex");
  const a = Buffer.from(provided, "hex");
  const b = Buffer.from(expected, "hex");
  return a.length === b.length && crypto.timingSafeEqual(a, b);
}

export function stealthWebhookHandler(req: Request, res: Response) {
  if (!verifySignature(req.rawBody, req.get("x-stealth-signature"))) {
    return res.status(401).send("invalid signature");
  }
  const event = JSON.parse(req.rawBody.toString("utf8"));
  // Idempotency: dedupe on event.event_id before processing
  // ...
  return res.status(200).send("ok");
}
```

Zero-downtime webhook-secret rotation (dual-verify window). When we rotate your `webhook_secret`, Stealth signs every delivery with **both** the new and old secret for a 7-day overlap window:

- `X-Stealth-Signature` — HMAC with the **new** secret (always present).
- `X-Stealth-Signature-Previous` — HMAC with the **old** secret (present only during the overlap window).

To cut over without dropping a single event, accept a delivery if **either** header validates against the secret you currently hold:

```
// verifySignature() reuses your single configured WEBHOOK_SECRET; accept the
// delivery if either signature header validates against it.
function verifyWithRotation(rawBody: Buffer, req: Request) {
  return (
    verifySignature(rawBody, req.get("x-stealth-signature")) ||
    verifySignature(rawBody, req.get("x-stealth-signature-previous"))
  );
}
```

Rotation steps: (1) we rotate and the old secret stays valid for 7 days; (2) deploy the new secret everywhere on your side; (3) once your fleet runs the new secret, the `X-Stealth-Signature-Previous` header simply stops appearing after the window expires. No coordinated cutover call required.

Notes:

- Verify against the **raw request body bytes** before any JSON parsing or middleware mutation. Express's `bodyParser.json` mutates the body — capture `rawBody` via the `verify` hook.
- Use `crypto.timingSafeEqual` — naïve `===` leaks timing information.
- Reject unsigned, malformed, or expired (see § 17.3) requests with `401`. Do not answer `2xx` until verification succeeds.

17.3 Idempotency & Retry Handling

Stealth Health retries failed webhooks up to 6 times (`30s, 5m, 30m, 2h, 12h` backoff). Your endpoint **will see duplicates** during transient outages.

- **Dedupe on `event.event_id`.** Store the most recent N event IDs (e.g. last 10,000 in Redis with a 7-day TTL, or a `partner_webhook_events` table with a UNIQUE constraint).
- **Process within 10 seconds.** Stealth Health times out the webhook delivery at 10s; longer work must be enqueued (SQS, Pub/Sub, BullMQ) before responding `2xx`.
- **Respond `2xx` only after persisting** the event reference. If you crash between persist and ack, the next retry will re-process — so processing must be idempotent on `event_id`.
- For inbound API calls you make to Stealth Health, the platform **is** idempotent on `partner_reference` for `POST /partner/referrals` and `POST /partner/prescriptions`. Always set `partner_reference` to your own primary key, not a generated UUID per attempt.
- If you receive `5xx` from `api.stealth.health`, retry with exponential backoff (start 1s, cap 60s, max 5 attempts). `4xx` errors should not be retried; surface them as integration alerts.

17.4 Minimum Necessary & Local Redaction

The HIPAA "Minimum Necessary" rule applies to your downstream use of PHI received from this API.

- **Do not request what you do not display.** If your partner UI only shows appointment status, do not call `GET /partner/patients/:id/intake`. Each request is audit-logged on our side and counts against the partner's annual review.
- **Redact in your own logs.** Wrap any logger you use with a serializer that strips fields tagged as PHI: `first_name`, `last_name`, `dob`, `email`, `phone`, `address.*`, `intake_responses[*].answer`, `messages[*].body`, `results[*].value`, `results[*].result`, `results[*].interpretation`, `missing_results[*].note`. Most successful partners maintain a single `safeLog()` helper that applies this list before any `console.log` / `logger.info` call.
- **Never include PHI in URL paths or query strings** when forwarding to internal services — those frequently land in unredacted load-balancer logs.
- **Do not store webhook payloads verbatim.** Persist only the fields you need; drop the rest.
- For analytics, use **`partner_reference`** **and** **`referral_id`** **as the join key**. They are de-identified surrogate IDs and are safe to send to third-party analytics tools.

17.5 Logging, Monitoring & SIEM

For HIPAA § 164.312(b) (audit controls), you must be able to answer the question: *"Show me everyone in our system who viewed PHI for patient X between dates Y and Z."*

- Emit a structured audit log entry from any internal handler that **reads** PHI received from this API. Recommended schema:

```
{
  "ts": "2026-04-23T14:02:11Z",
  "actor": "alice@partner.com",
  "actor_role": "support_agent",
  "action": "phi.view",
  "stealth_patient_id": "ptn_abc123",
  "fields_viewed": ["intake_responses", "appointment_status"],
  "request_id": "req_…",
  "ip": "203.0.113.10"
}
```

- Ship those logs to an immutable, retention-controlled store (Datadog Audit Trail, Splunk, AWS CloudTrail Lake, GCP Cloud Logging with retention buckets). **6-year retention** is the contractual minimum.

- Set alerts on:
  - Sustained `401` / `403` from `api.stealth.health` (credential or tier misconfiguration).
  - `429` rate-limit responses (you're approaching capacity).
  - Webhook signature verification failures (potential attack).
  - `4xx` rate from inbound webhooks (your handler is rejecting valid traffic).
- Surface incident-response contact info to your on-call team: a partner-side incident that involves PHI received from this API is a Stealth Health incident too.

17.6 Sandbox-to-Production Cutover

Before flipping production traffic, complete this checklist:

- [ ] BAA executed and signed by both parties.
- [ ] Production API key and webhook secret stored in your secrets manager (separate from sandbox).
- [ ] Webhook receiver is publicly reachable, returns `2xx` within 10s, and verifies `X-Stealth-Signature`.
- [ ] Idempotency layer in place (dedupe on `event_id`).
- [ ] Inbound retries on `5xx` from `api.stealth.health` configured.
- [ ] PHI-redacting logger in use across all services that handle responses.
- [ ] Audit log shipping to long-term retention store, with a 6-year retention policy.
- [ ] On-call paging set up for `401` / `429` / signature-verification spikes.
- [ ] Sub-processor list shared with `partners@stealth.health`.
- [ ] Workforce HIPAA training records on file for everyone with PHI access.
- [ ] DR / backup procedure tested for any datastore that holds Stealth Health PHI.

Once the checklist is complete, email `partners@stealth.health` to request production credentials and the production webhook URL allowlist update.

Appendix: Status Lifecycle

Same as the Referral Tier — see Referral Tier, Appendix.

Questions for Partner Discussion

1. **Data scope** — Does the partner need all intake responses, or only specific categories (e.g. symptoms + medications but not demographics)?
2. **Prescription visibility** — Should the partner see pending prescriptions or only signed ones?
3. **Tracking numbers** — Confirm the partner accepts liability for securing tracking numbers (PHI correlation risk).
4. **Data retention** — How long will the partner store patient data? Needs to align with BAA terms.
5. **Patient consent** — Will patients be informed that their data is shared with the partner? Consent flow design.
6. **Webhook vs. polling** — Does the partner prefer real-time webhooks, periodic polling, or both?
7. **Subscription/refill visibility** — Should the partner see recurring subscription status and upcoming refill dates?
8. **Revenue share** — How will partner compensation be structured for this tier?
9. **White-label branding** — Full white-label (custom domain, partner-branded emails) or co-branded?
10. **Patient support** — Which party handles first-line patient inquiries?

This document is a proposal for discussion purposes. Endpoint paths, field names, and behaviors are subject to change during implementation.