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

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 in **Google Cloud Firestore** in the `us-east5` region (Columbus, Ohio). Firestore enforces AES-256 encryption at rest with Google-managed keys (FIPS 140-2 validated).
- Backups (daily PITR + on-demand exports) are retained encrypted in `gs://*-backups` buckets with the same KMS protections.
- Pharmacy / fulfillment files (PDF prescriptions, lab requisitions) are stored in Cloud 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 the SHA-256 hash (`api_key_hash`) is persisted in

`partner_configs`.

- Key comparison uses `crypto.timingSafeEqual` so the auth path is not susceptible to timing oracles.
- Key rotation is supported with a **dual-hash window**: `previous_api_key_hash` 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 § 16.2). 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 the `partner_events` collection 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 in `partner_configs.allowed_ips`).

Audit trail (server-side)

Every PHI-bearing read or mutation produced by a Cloud Function or Next.js route is recorded in the immutable `auditLogs` collection. The schema (versioned, `AUDIT_LOG_SCHEMA_VERSION = 1`) 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`	`cloud_function` / `next_api` / `firestore_trigger` etc. + 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 by a CI guard (`.github/scripts/check-audit-coverage.mjs`) that fails any PR which adds writes to an audited collection without a corresponding `safeExplicitAudit` call. 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 separate Firebase projects with non-overlapping credentials. Sandbox contains only synthetic data — never real PHI.
- Internal staff access to PHI is gated by Google Workspace SSO + MFA + role-based custom claims (`admin` , `pharmacy` , `prescriber` , etc.) and is itself audit-logged under `category: "phi_view"` .
- Engineers do not have direct read access to production Firestore; 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 (Firestore, Cloud Functions, Cloud 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.
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.*`, `fulfillment.*`	Signed; same retention rules as above.

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 § 16.5.
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 in the partner's `partner_configs` record.	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`

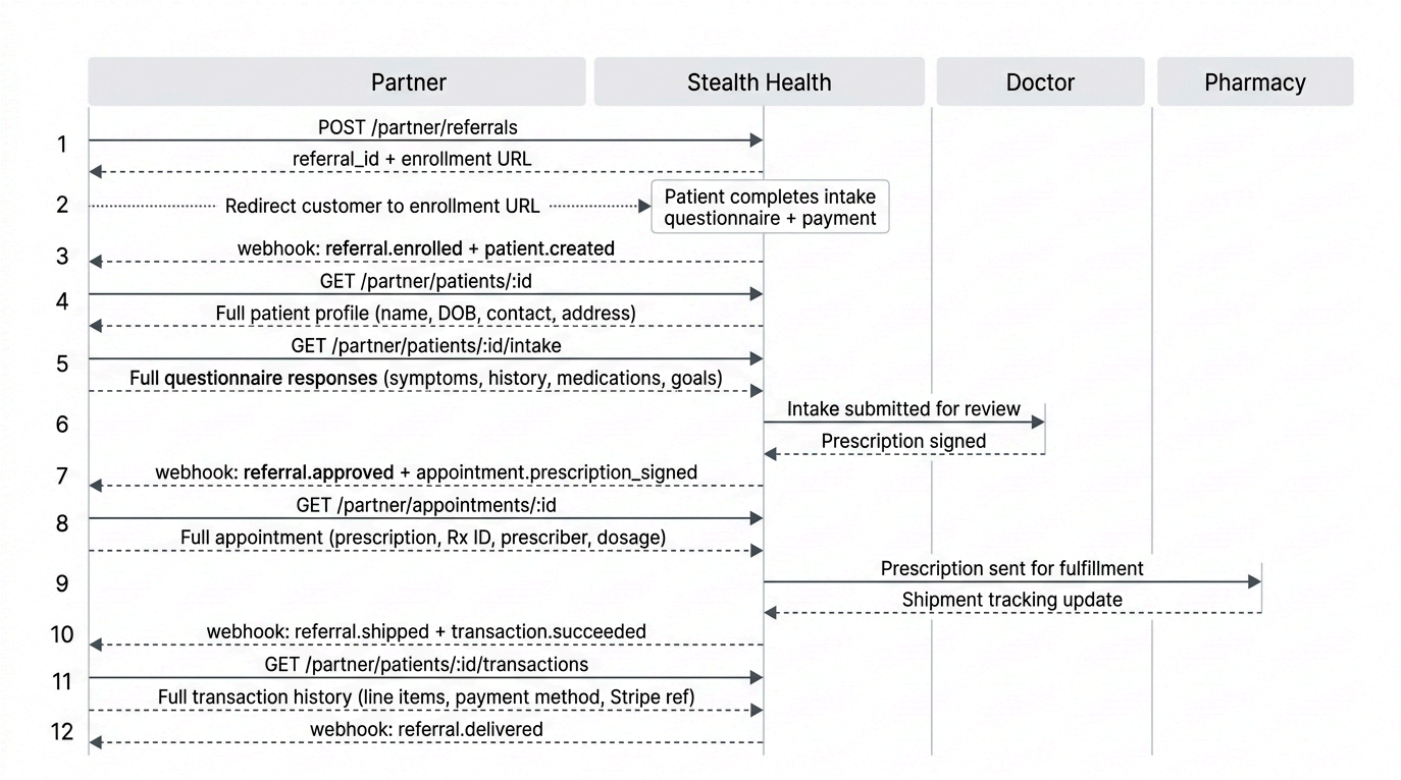
All other security features (key rotation, webhook signatures, TLS, IP allowlisting) are the same as the Referral Tier guide, Section 3.

---

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.*

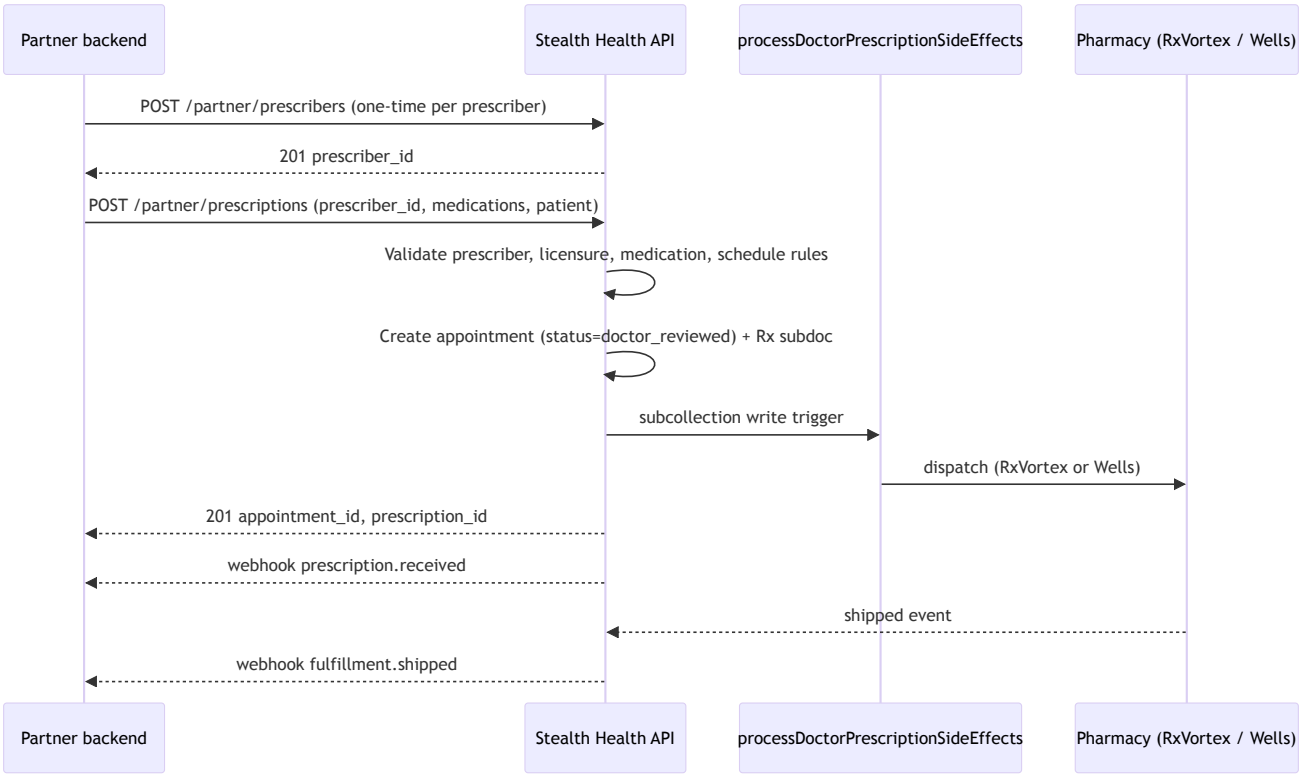
5.2 Step-by-Step

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 `fulfillment.*` webhooks as the order progresses.

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.

---

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.

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)

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 § 14 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. 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.

12.1 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"
}
```


12.2 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 profile)
`patient.updated`	Patient updates their profile	`patient` (changed fields)
`appointment.intake_completed`	Intake questionnaire submitted	`appointment_id`, `intake_summary`
`appointment.prescription_signed`	Prescription signed by doctor	`appointment_id`, `prescription` (full detail)
`transaction.succeeded`	Payment successfully processed	`transaction` (full detail)
`transaction.refunded`	Payment refunded	`transaction`, `refund_amount_cents`, `reason`
`prescription.received`	Partner-submitted prescription accepted by `POST /partner/prescriptions` and queued for fulfillment	`appointment_id`, `prescription_id`, `rx_id`, `prescriber_id`, `medications[]`
`prescription.rejected`	Partner-submitted prescription rejected after async re-validation (reserved; not used in v1's synchronous happy path)	`appointment_id?`, `prescription_id?`, `error_code`, `error_message`

12.3 Signature Verification & Retry Policy

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

---

13. Data Models

13.1 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"
}
```

13.2 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"
}
```

13.3 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"
    }
  ]
}
```

13.4 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"
    }
  ]
}
```

13.5 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"
}
```

13.6 Prescriber Object

Returned by the `/partner/prescribers` endpoints. Stored in the Firestore collection `partner_prescribers/{prescriber_id}`.

```
{
  "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"
}
```

13.7 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"
    }
  ],
  "created_at": "ISO 8601"
}
```

14. 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`	Firestore 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

---

15. 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`

---

16. 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.

16.1 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.

16.2 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");
}
```

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 § 16.3) requests with `401`. Do not answer `2xx` until verification succeeds.

16.3 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.

16.4 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`. 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.

16.5 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.

16.6 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.