

Amtaar Partner API — Integration Guide

How a bank, broker or fintech app connects to Amtaar: send us investors, let them buy property shares from inside your product, and follow every order, payout and commission.

Audience: product and engineering teams at banks, brokers and fintech apps that partner with Amtaar · API version **v1 (2026-10-18)** · 18 Oct 2026

The complete endpoint reference (every field, every error) is at amtaar.app/developers/reference, generated from the OpenAPI 3.1 contract your developers can download and generate a client from (openapi.json). Partner access: hi@amtaar.dev.

Contents

1. What you can build
2. How a partnership goes live
3. Key concepts
4. Quick start (sandbox)
5. The flows in detail
6. Webhooks
7. Errors, retries and limits
8. Security and data protection
9. Testing in the sandbox
10. Go-live checklist
11. Endpoint summary
12. FAQ

1. What you can build

Amtaar sells fractional ownership of rented commercial property in Egypt: an investor buys **shares** of a unit (one share = one square metre), receives a share of the rent every quarter, and owns a registered stake. The Partner API lets your product offer this to your customers in one of three ways. You can use one or several.

Model	Your customer...	Money	What you build
A. Referral	taps a link in your app, installs Amtaar, invests there	paid to Amtaar in the Amtaar app	one API call to register the customer, then webhooks to show progress and commissions
B. Embedded, partner-settled	browses and buys Amtaar units inside <i>your</i> app	you debit your customer's account and transfer to Amtaar	catalogue, quote, order and settlement calls; status screens fed by webhooks

Model	Your customer...	Money	What you build
C. Embedded, card checkout	buys inside your app and pays by card on an Amtaar (Paymob) page	paid to Amtaar by card	catalogue and order calls; open the checkout page in a web view

In every model you can also show your customers their **holdings, rental income and wallet** (with their consent and the matching permission), and track **your commissions**.

2. How a partnership goes live

1. **Agreement.** Amtaar and the partner agree on the model(s), the commission terms and, for model B, the settlement account and KYC reliance.
2. **Sandbox access.** Amtaar enables your partner account on the sandbox server and issues sandbox API keys (`amtk_sandbox_...`). The sandbox is a full copy of the platform with test data. It never moves real money and has extra endpoints that simulate what Amtaar staff and customers do.
3. **Build and test.** Your developers integrate against the sandbox base URL we send with your keys, and register a webhook endpoint.
4. **Certification call.** We walk through the go-live checklist (section 10) together.
5. **Live access.** Amtaar enables the same capabilities on the live server and issues live keys (`amtk_live_...`), optionally limited to your server IP addresses.

Capabilities

What your keys can do is bounded by **capabilities** that Amtaar switches on according to your agreement. `GET /me` always shows the current set.

Capability	Allows	Typical for
<code>createCustomers</code>	open an Amtaar account for a person new to Amtaar	all models
<code>linkExistingCustomers</code>	link someone who already has an Amtaar account, after they confirm an SMS code	all models
<code>partnerKycReliance</code>	your own regulated KYC counts for orders placed through the API	B, C (needs a reliance agreement)
<code>partnerSettlementOrders</code>	place orders that you pay for and settle to Amtaar	B
<code>cardCheckoutOrders</code>	place orders paid on an Amtaar card page	C
<code>portfolioAccess</code>	holdings, rental payouts, contracts of linked customers	showing portfolios
<code>walletAccess</code>	Amtaar wallet balance of linked customers	showing portfolios

Each **API key** also has **scopes** (for example `orders:write`), so you can give a reporting job a read-only key and your order service a write key.

3. Key concepts

Customer. A person you brought or linked, identified by `cus_...` (and by your own `externalId` if you send one). You only ever see customers linked to you.

Attribution. When you open a new Amtaar account through the API, that investor is attributed to you. Every purchase they make, in your app or in the Amtaar app, earns you the commission in your agreement. Existing Amtaar investors keep their existing attribution when you link them.

Consent. If the mobile number already belongs to an Amtaar investor, nothing about them is shared until they confirm. Amtaar texts them a 6-digit code, and you collect it in your app and send it to us (`/consent` , then `/consent/verify`).

KYC. Before anyone can invest they must be verified. Amtaar KYC happens in the Amtaar app: mobile number, e-mail, PIN, then national ID and face match. If your account has `partnerKycReliance` , you may instead attest your own KYC when you create the customer, and orders through the API will accept it. `customer.kyc.missingSteps` and `customer.canInvest` tell you where each customer stands.

Order. A purchase of N shares of one property at the current price, plus the management fee. Placing an order **reserves** the shares. An unpaid API order releases them at `expiresAt` (your hold time, 24 h by default; 60 minutes for card checkout).

Order statuses (stable; your screens can map them one to one):

Status	Meaning	Next
<code>awaiting_payment</code>	shares reserved, waiting for the money	<code>payment_under_review</code> , <code>paid</code> , <code>cancelled</code>
<code>payment_under_review</code>	your settlement or the customer's transfer proof arrived; Amtaar Finance is checking it (normally within one working day)	<code>paid</code> , <code>payment_rejected</code>
<code>paid</code>	money confirmed. Shares are issued when the unit is fully funded	<code>completed</code> , <code>cancellation_requested</code>
<code>completed</code>	the customer owns the shares; the contract follows	—
<code>cancelled</code>	cancelled by you or the customer, or expired unpaid; shares released	—
<code>payment_rejected</code>	Finance could not match the payment	<code>awaiting_payment</code> (retry), <code>cancelled</code>
<code>cancellation_requested</code>	the customer asked to cancel a paid order; Amtaar reviews it (fees may apply)	<code>cancelled</code> , <code>refunded</code> , back to <code>paid</code>
<code>refunded</code>	money returned	—

Environments. Sandbox and live are separate servers with separate keys and data. A sandbox key is refused by the live server, and the other way round.

4. Quick start (sandbox)

Every request carries your key:

```
export AMTAAR=<sandbox base URL sent with your keys> # e.g. https://.../open/v1
export KEY=amtk_sandbox_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
curl -s $AMTAAR/me -H "Authorization: Bearer $KEY"
```

1 — Show the catalogue and price a purchase.

```
curl -s "$AMTAAR/properties?status=available" -H "Authorization: Bearer $KEY"
curl -s -X POST $AMTAAR/quotes -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
  -d '{"property":"AMT-26-017","shares":10}'
```

2 — Register your customer. Always send an `Idempotency-Key`. If the call times out, retry with the same key.

```
curl -s -X POST $AMTAAR/customers -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
  -H "Idempotency-Key: 4f0c6a5e-2a3b-4f7e-9f1e-0c1d2e3f4a5b" \
  -d '{"phone":"+201001234567","externalId":"your-user-88213","name":{"en":"Mona Ahmed","ar":"منى أحمد"},"language":"ar"}'
```

The answer is a `customer` with `status: "onboarding"` (new account), `"pending_consent"` (existing investor, see section 5.1) or `"active"` (can invest).

3 — In sandbox, pretend the customer finished KYC.

```
curl -s -X POST $AMTAAR/sandbox/customers/cus_.../complete-kyc -H "Authorization: Bearer $KEY"
```

4 — Place a partner-settled order with the settlement in the same call. `amount` must equal the quoted `total`.

```
curl -s -X POST $AMTAAR/orders -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
  -H "Idempotency-Key: 8d2b1c0e-..." \
  -d '{"customer":"cus_...", "property":"AMT-26-017", "shares":10, "externalReference":"your-order-55120",
    "funding":{"method":"partner_settlement", "settlement":{"reference":"TRX-99812", "amount":102667.12}}}'
```

5 — Move it along as Amtaar would (sandbox only), and watch your webhooks arrive:

```
curl -s -X POST $AMTAAR/sandbox/orders/ord_.../simulate -H "Authorization: Bearer $KEY" \
  -H "Content-Type: application/json" -d '{"to":"completed"}'
curl -s "$AMTAAR/customers/cus_.../holdings" -H "Authorization: Bearer $KEY"
```

5. The flows in detail

5.1 Registering and linking customers

1. `POST /customers` with the customer's Egyptian mobile number (any common format; `+2010...`, `010...`).
2. **New to Amtaar:** an account is opened and attributed to you (event `customer.created`). The customer later signs in to the Amtaar app with **the same mobile number**. There is nothing to sign up for again.

3. **Already an Amtaar investor:** status `pending_consent` , phone shown masked. Call `POST /customers/{id}/consent` , ask the customer for the code we text them, then `POST /customers/{id}/consent/verify` (event `customer.linked`). Codes last 10 minutes; five wrong tries cancel a code.
4. Optional `nationalId` is used only to stop a second account for the same person. It is not stored.
5. `customer.kyc_verified` arrives when they finish Amtaar KYC.

5.2 Model A — referral

Register the customer (5.1), then send them to `customer.links.app` (your tracked Amtaar download link). They invest in the Amtaar app, and you receive `order.created` , `order.paid` , `order.completed` for their orders (`source: "app"`), plus `commission.created` when your commission is booked.

5.3 Model B — embedded, partner-settled

1. Show units (`GET /properties`) and the price (`POST /quotes`).
2. Make sure the customer can invest (`customer.canInvest`).
3. Debit your customer's account for the quoted `total` .
4. `POST /orders` with `funding.method = "partner_settlement"` and `funding.settlement` (your transfer reference and the exact amount). If you debit later, omit `settlement` and call `POST /orders/{id}/settlement` before `expiresAt` .
5. Transfer the money to Amtaar's settlement account using the same reference.
6. Amtaar Finance matches the transfer and the order becomes `paid` (`order.paid`). If it cannot be matched it becomes `payment_rejected` ; contact Amtaar with the reference.
7. When the unit is fully funded the shares are issued: `order.completed` .

If an order is rejected or cancelled after you debited your customer, refund your customer from your side. Amtaar returns the settled amount to you according to the agreement.

5.4 Model C — embedded, card checkout

`POST /orders` with `funding.method = "card_checkout"` . Open `funding.checkoutUrl` (Paymob) in a web view within 60 minutes. The order becomes `paid` when the card payment is confirmed, then `completed` as above.

5.5 Showing the portfolio

With `portfolioAccess` : `GET /customers/{id}/holdings` (shares per property, invested amount, total rental income), `/payouts` (each quarterly rental payment) and `/contracts` (contract number and signing status). With `walletAccess` : `/wallet` . The `payout.paid` event tells you when new rent arrived.

5.6 Commissions

`GET /commissions` lists every row of your ledger (signup fees, purchase commissions, bonuses, payouts to you, adjustments, reversals). `GET /commissions/summary` shows totals and the balance Amtaar owes you. Rows marked `testMode` were recorded while your account was in test mode and are not payable. A cancelled or refunded order reverses its commission.

6. Webhooks

Register an `https` endpoint (`POST /webhook-endpoints` , or ask Amtaar). Copy the signing secret from the answer; it is shown only once. Every event is POSTed as JSON:

```
{
  "object": "event",
  "id": "evt_6712c0ffee0123456789abcd",
  "type": "order.paid",
  "environment": "live",
  "apiVersion": "2026-10-18",
  "createdAt": "2026-10-18T09:41:07.000Z",
  "data": { "order": { "object": "order", "id": "ord_...", "status": "paid", "...": "..." } }
}
```

Verify every delivery. The header is `Amtaar-Signature: t=<unix seconds>,v1=<hex>` , where `v1 = HMAC-SHA256(secret, t + "." + raw body)` .

```
// Node.js (Express): use the RAW body, not re-serialised JSON
const crypto = require('crypto');
app.post('/amtaar/webhooks', express.raw({ type: 'application/json' })), (req, res) => {
  const [, t, v1] = /^t=(\d+),v1=([0-9a-f]{64})$/ .exec(req.get('Amtaar-Signature')) || '' || [];
  const expected = t && crypto.createHmac('sha256',
process.env.AMTAAR_WEBHOOK_SECRET).update(`${t}.${req.body}`).digest();
  const fresh = t && Math.abs(Date.now() / 1000 - Number(t)) < 300;
  if (!v1 || !fresh || !crypto.timingSafeEqual(Buffer.from(v1, 'hex'), expected)) return
res.sendStatus(400);
  const event = JSON.parse(req.body);
  queue.add(event);          // do the work asynchronously
  res.sendStatus(200);        // answer within 10 seconds
});
```

```
# Python (Flask)
import hmac, hashlib, time, re
@app.post("/amtaar/webhooks")
def amtaar_webhook():
    m = re.fullmatch(r"t=(\d+),v1=([0-9a-f]{64})", request.headers.get("Amtaar-Signature", ""))
    raw = request.get_data()
    if not m or abs(time.time() - int(m[1])) > 300:
        abort(400)
    expected = hmac.new(SECRET.encode(), m[1].encode() + b"." + raw, hashlib.sha256).hexdigest()
    if not hmac.compare_digest(expected, m[2]):
        abort(400)
    enqueue(request.get_json())
    return "", 200
```

Delivery rules.

- Any `2xx` answer within 10 seconds is a success. Otherwise we retry after 1 m, 5 m, 15 m, 1 h, 3 h, 6 h, 12 h and 24 h (9 attempts over about two days).
- An endpoint that fails 100 times in a row is disabled. Re-enable it with `PATCH /webhook-endpoints/{id} ( "status": "enabled" )`.
- Events can arrive **more than once** and **out of order**. Deduplicate on `Amtaar-Event-Id`. When the order of events matters, re-read the object (`GET /orders/{id}`).
- Events are produced within about a minute of the change.
- After downtime, catch up with `GET /events?after=<last event id you processed>` until `hasMore` is false. Events are kept 180 days.
- `POST /webhook-endpoints/{id}/test` sends a signed test event and shows you what your server answered.

Event	When
<code>customer.created</code>	you opened a new Amtaar account
<code>customer.linked</code>	an existing investor confirmed the link
<code>customer.kyc_verified</code>	a linked customer finished Amtaar KYC
<code>customer.unlinked</code>	a customer was unlinked
<code>order.created</code>	an order exists (placed by you, or by your customer in the app)
<code>order.payment_under_review</code>	payment evidence received; Finance is checking
<code>order.paid</code>	money confirmed
<code>order.completed</code>	shares issued to the customer
<code>order.cancelled</code>	cancelled or expired unpaid; shares released
<code>order.payment_rejected</code>	Finance rejected the payment
<code>order.cancellation_requested</code>	the customer asked to cancel a paid order
<code>order.refunded</code>	order refunded
<code>payout.paid</code>	rental income credited to the customer (needs <code>portfolioAccess</code>)
<code>commission.created</code>	a row was added to your commission ledger

7. Errors, retries and limits

Every error has the same shape. Branch on `code`, show `message` only in logs, and quote `requestId` when you contact us.

```
{ "error": { "type": "permission_denied", "code": "customer_not_eligible",
  "message": "The customer must finish verification before investing",
  "details": { "missingSteps": ["identity_verification"] }, "requestId": "1f0c..." } }
```

HTTP	type	Common code s	What to do
400	invalid_request	invalid_phone , invalid_shares , idempotency_key_required	fix the request
401	authentication_error	invalid_api_key , api_key_revoked , wrong_environment	check the key and the base URL
403	permission_denied	missing_scope , capability_not_enabled , customer_not_eligible , consent_required , ip_not_allowed	permission or customer state — do not retry unchanged
404	not_found	resource_not_found	wrong id, or not linked to you
409	conflict	external_id_in_use , already_settled , order_not_cancellable , state_conflict	re-read the object; the state has moved on
422	unprocessable / idempotency_error	amount_mismatch , order_limit_exceeded , daily_limit_exceeded , property_not_available , idempotency_key_reused	valid request that cannot be done
429	rate_limited	rate_limited	wait Retry-After seconds
500/503	api_error / service_unavailable	internal_error , checkout_unavailable , sms_unavailable	retry with backoff, same Idempotency-Key

- **Idempotency.** POST /customers , POST /orders and POST /orders/{id}/settlement require an Idempotency-Key (a UUID per operation). A retry with the same key and body returns the original answer (Idempotent-Replayed: true) and never creates a second customer or order. Keys are remembered for 24 hours.
- **Rate limit.** Per key; 300 requests per minute unless agreed otherwise. Every answer carries RateLimit-* headers.
- **Order limits.** Amtaar may set a maximum per order and per day for API orders (GET /me → limits).
- **Versioning.** v1 only changes in backward-compatible ways: new endpoints, optional fields, response fields, event types and enum values. Ignore what you do not recognise. A breaking change would come as v2, with v1 kept for at least six months.

8. Security and data protection

- Keep API keys and webhook secrets in a secret manager, on servers only. Never put them in a mobile app or a browser.
- Ask Amtaar to restrict live keys to your egress IP addresses, and to rotate a key at once if you suspect a leak. Several keys can be active, so you can rotate without downtime.
- Always verify webhook signatures and timestamps.
- Use the customer data only for the purposes in the agreement. DELETE /customers/{id} ends your access to a customer. Their Amtaar account and investments are unaffected.
- Linking an existing investor always needs their SMS confirmation. Never ask a customer for their Amtaar PIN or OTP for any other reason.

9. Testing in the sandbox

You want to test	How
A customer finishing KYC	<code>POST /sandbox/customers/{id}/complete-kyc</code>
Finance accepting a settlement	<code>POST /sandbox/orders/{id}/simulate {"to":"paid"}</code>
Shares being issued	<code>{"to":"completed"}</code>
A rejected payment	<code>{"to":"payment_rejected"}</code>
Consent of an existing investor	<code>POST /customers/{id}/consent</code> returns <code>sandboxCode</code> instead of <code>texting</code>
Your webhook handler	<code>POST /webhook-endpoints/{id}/test</code> ; sandbox also accepts <code>http://</code> URLs (e.g. a tunnel to your laptop)
Expiry	place an order and leave it unpaid past <code>expiresAt</code>

10. Go-live checklist

- ☐ Keys stored in a secret manager; the live key is IP-restricted.
- ☐ Every `POST` that creates something sends an `Idempotency-Key` and retries with the same key.
- ☐ Webhook signatures and timestamps are verified; events are de-duplicated by id; the handler answers within 10 seconds.
- ☐ A catch-up job reads `GET /events?after=...` after downtime.
- ☐ Every order status in section 3 has a screen or message.
- ☐ `customer_not_eligible` sends the customer to finish KYC (`missingSteps`).
- ☐ Model B: the debit, the settlement call and the bank transfer use the same reference; rejected or cancelled orders are refunded to your customer.
- ☐ Amounts are shown in EGP with two decimals, from `quote.total` / `order.amounts.payable` .
- ☐ Projections (`forecast`) are labelled as projections, not promises.
- ☐ A contact for incidents is shared with Amtaar.

11. Endpoint summary

Area	Endpoints
Account	<code>GET /me</code>
Properties	<code>GET /properties</code> , <code>GET /properties/{id or code}</code> , <code>POST /quotes</code>
Customers	<code>POST /customers</code> , <code>GET /customers</code> , <code>GET/PATCH/DELETE /customers/{id}</code> , <code>POST /customers/{id}/consent</code> , <code>POST /customers/{id}/consent/verify</code>

Area	Endpoints
Portfolio	GET /customers/{id}/holdings , /payouts , /contracts , /wallet
Orders	POST /orders , GET /orders , GET /orders/{id} , POST /orders/{id}/settlement , POST /orders/{id}/cancel
Commissions	GET /commissions , GET /commissions/summary
Events	GET /events , GET /events/{id}
Webhooks	GET/POST /webhook-endpoints , GET/PATCH/DELETE /webhook-endpoints/{id} , POST .../rotate-secret , POST .../test , GET /webhook-deliveries , POST /webhook-deliveries/{id}/retry
Sandbox	POST /sandbox/customers/{id}/complete-kyc , POST /sandbox/orders/{id}/simulate

12. FAQ

Does my customer need to download the Amtaar app? In model A, yes. In models B and C they can invest entirely inside your app, *if* their KYC is done (Amtaar KYC in the app, or your KYC under a reliance agreement). They can always open the Amtaar app later with the same mobile number to see their investment and contracts.

Why is an order paid but not completed ? Shares are issued when the whole unit is funded. Until then the money is held against the order. If a unit does not reach its funding, Amtaar contacts investors under its funding terms.

Can I cancel a paid order? Not through the API. The customer requests a cancellation in the Amtaar app, and Amtaar reviews it (`order.cancellation_requested`). Cancellation fees may apply.

Do I see orders my customers place in the Amtaar app? Yes, for customers linked to you. They have `source: "app"` and `funding.method: "in_app"`.

What about international mobile numbers? v1 accepts Egyptian mobile numbers. Customers abroad with an Egyptian number are supported; other numbers will come later.

Can I sell shares on behalf of a customer? Not yet. The resale market is planned and will be added to v1 as new endpoints when it launches.

Contact: Amtaar partnerships — hi@amtaar.dev · amtaar.app/developers. Quote the `requestId` of any failing call.