Linkitecture

Version 1.0 · November 2026

INTEGRATION GUIDE

Connect a business to Linkitecture.

How a website, a checkout, your own server or an AI assistant sends orders and enquiries to the Linkitecture app, and how the app sends them back out. Use cases, copy-paste code and ready-made prompts.

FOR
Developers and studios setting up a client

TIME
About 15 minutes per website

WEB
app.linkitecture.com/guide

01

What it does

Linkitecture is an iPhone and iPad app for the people who run a business and the developer who looks after its website. Once a site is connected, everything a customer does on it arrives in the app as it happens.

Restaurant or café

Online orders land in **Orders** with a push alert. Staff tap Accept, Preparing, Ready. The customer follows each step on their tracking page and can message the shop.

Shop

Paid orders arrive from the checkout. When the parcel ships, the carrier's tracking link appears for the customer. Pause ordering when you're out of stock or closed.

Service business

Contact and booking forms become **Enquiries** with the person's name, email, phone and message, so nothing sits unread in an inbox.

The developer

Uptime and SSL alerts, deploy notices from GitHub, Vercel, Netlify or Cloudflare, client messages, invoices and quotes, for every client in one app.

Your own software

The public API and signed webhooks let a POS, a spreadsheet tool or Zapier read orders, change their status and message customers.

An AI assistant

Claude, ChatGPT and other MCP apps can check orders and reply to customers through a key with only the permissions you choose.

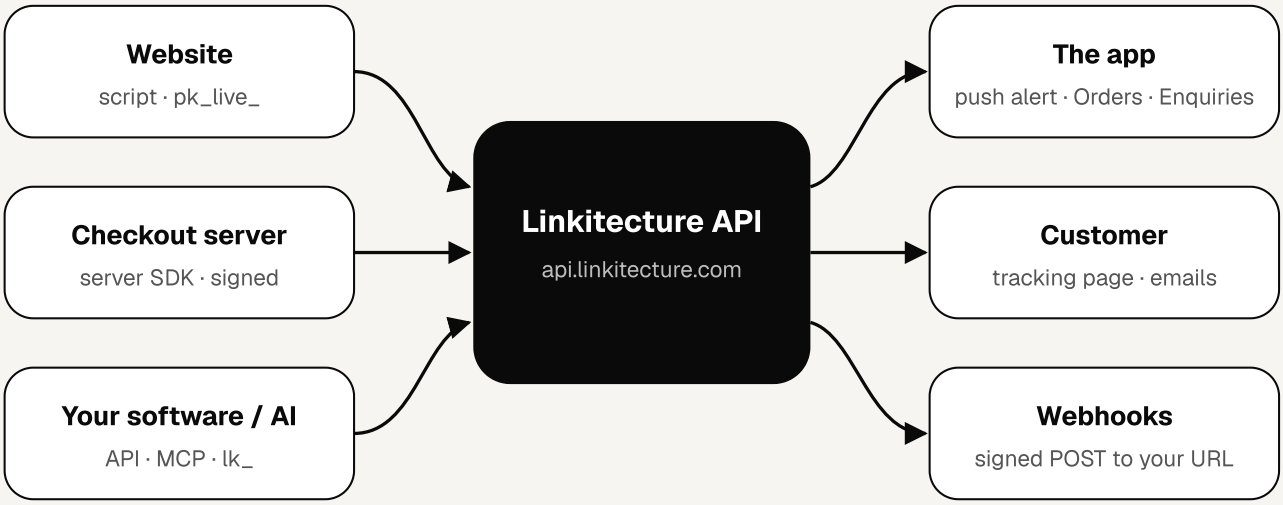
Five ways in

Piece	Use it for	Key
Website script	Forms and simple orders from any site, no build step	`pk_live_…` public
Server SDK	Paid orders from your checkout's server	key + secret
Public API	Reading orders, changing status, messaging customers	`lk_live_…` / `lk_test_…`
Webhooks	Being told the moment something happens	`whsec_…` signing secret
MCP	AI assistants	`lk_…` key

02

How it connects

One API sits in the middle. Websites and servers send events in; the app, webhooks and AI assistants read them out.



In seconds

An event from the website reaches the owner's phone within a few seconds as a push notification.

Never twice

Every event carries its own id. Retries and double-clicks are recognised and recorded once.

Walled off

Each key belongs to one business. A key for one client can never see another client's data.

Addresses

Website script	`https://app.linkitecture.com/sdk/linkitecture.js`
Event ingest	`https://api.linkitecture.com/v1/ingest`
Public API	`https://api.linkitecture.com/v1/public` · OpenAPI at `/v1/public/openapi.json`
MCP server	`https://api.linkitecture.com/v1/mcp`

03

Before you start

You need the app (TestFlight beta), a developer account in it, and the business added as a client.

1. Open Linkitecture → **Clients** → add the business (name, type, website address).
2. Invite the owner from **Tools → Invite**. They sign in on their own phone and allow notifications.
3. Open the client → **Connect**. The app walks you through nine steps; this guide follows the same order.

The keys, and where each one lives

Key	Where you get it	Where it goes	Secret?
`pk_live_…`	Connect → 1. Website key	Website code	No. It only works from the website address you gave it.
Secret	Shown once with the website key	Your server's environment variables	**Yes.** Never in browser code or git.
`lk_test_…` / `lk_live_…`	More → Connect your AI (or the client's "Connect an AI assistant")	Your server, Zapier, an AI app	**Yes.** Test keys only touch test data.
`whsec_…`	More → Webhooks → add an endpoint	Your webhook receiver	**Yes.**

Rule of thumb. Anything starting `pk_` is safe in a web page. Everything else stays on a server or in a password manager. If one leaks, delete it in the app and make a new one; the old one stops working at once.

API key permissions

`orders:read`	See orders and whether ordering is open
`orders:write`	Change an order's status; create sandbox test orders
`messages:read`	Read the conversation on an order (never staff notes)
`messages:write`	Message the customer on their tracking page
`events:read`	Catch up on everything that happened

04

Connect the website

Steps 1 and 2 in the app. Contact and booking forms start arriving as Enquiries.

1. Create the website key

1. Client → **Connect** → **1. Website key**.
2. Type the website address exactly as visitors see it: `https://`, no trailing slash. Add `www` as its own key if the site uses both.
3. Tap **Create website key**. Copy the public key. If the site has a server, store the secret there now: it is shown once.

2. Add the script (any website)

Paste before `</body>` on every page, then add `data-linkitecture-form` to each form you want captured. The value names the form in the app.

index.html

```
<script src="https://app.linkitecture.com/sdk/linkitecture.js"
        data-key="pk_live_YOUR_KEY" data-auto-capture defer></script>

<form data-linkitecture-form="contact" action="/thanks">
  <input name="name" autocomplete="name">
  <input name="email" type="email">
  <input name="phone" type="tel">
  <textarea name="message"></textarea>
  <button>Send</button>
</form>
```

The form still submits as before. Linkitecture reads it on the way through; nothing else on the page changes.

Or call it yourself

any script on the page

```
Linkitecture.enquiry({ name: "Jo", email: "jo@example.com", phone: "0400 000 000",
  message: "Table for 4 on Friday?", form: "booking" });
```

05

React and Next.js

For sites with a build step, install the SDK and start it once in the layout.

app/linkitecture-provider.tsx

```
"use client";
import { useEffect } from "react";
import { Linkitecture } from "@linkitecture/sdk";

export default function LinkitectureProvider() {
  useEffect(() => {
    Linkitecture.init({
      key: process.env.NEXT_PUBLIC_LINKITECTURE_KEY!,
      autoCapture: true,
    });
  }, []);
  return null;
}
// Render <LinkitectureProvider /> in app/layout.tsx and add
// data-linkitecture-form="contact" to your forms.
```

.env.local

```
NEXT_PUBLIC_LINKITECTURE_KEY=pk_live_YOUR_KEY
```

Not on npm yet. The `@linkitecture/sdk` package isn't published. Until it is, use the script tag from page 4 (it works in React too), or copy the SDK's source from the developer who set you up.

06

Send online orders

Step 3 in the app. Send the order from your server once payment succeeds, so prices can't be changed in the browser.

app/api/order-paid/route.ts

```
import { createServerClient } from "@linkitecture/sdk/server";

const linkitecture = createServerClient({
  key: process.env.LINKITECTURE_KEY!,       // pk_live_…
  secret: process.env.LINKITECTURE_SECRET!, // server only
});

export async function POST(req: Request) {
  const o = await req.json(); // your paid order
  await linkitecture.order({
    customer_name: o.name,
    customer_email: o.email,
    external_ref: o.id,                 // your order number
    fulfilment: "pickup",               // "delivery", "dine_in" or "other"
    items: o.items.map((i) => ({ name: i.name, quantity: i.qty, unit_price_cents: i.cents })),
    total_cents: o.totalCents,
    currency: "AUD",
  });
  return Response.json({ ok: true });
}
```

Works on Node 18+, Cloudflare Workers, Vercel Edge and anything with `fetch` and WebCrypto. Each request is signed with the secret and re-signed on every retry.

Other languages:
`signature = hex(HMAC_SHA256(key = hex(SHA256(secret)), timestampMs + "." + rawBody))`, sent as `X-Linkitecture-Key`, `X-Linkitecture-Timestamp` and `X-Linkitecture-Signature` to `https://api.linkitecture.com/v1/ingest`.

What the owner sees

A push alert, then the order in **Orders** with Accept and Decline. Each status change updates the customer's tracking page and sends the confirmation emails you set up in Connect → 8.

07

Tracking, alerts and deploys

Order tracking on the website (step 9)

Customers paste the tracking link from their confirmation email and watch live progress without leaving the site. It holds no secrets and can't change anything.

the tracking page

```
<div id="linkitecture-track"></div>
<script src="https://api.linkitecture.com/widget/track.js" data-title="Track your order"
        data-color="#2563eb" async></script>
```

Set `data-color` to the brand colour.

Push notifications (4)

Nothing to code per client. The owner allows notifications and picks which alerts they want in More → Notifications. Staff only get what their role allows.

Website host (5)

Connect GitHub, Vercel, Netlify or Cloudflare and the app shows each deploy, and tells you when one fails.

Uptime and SSL (6)

Checked every few minutes. You (and the owner, if you choose) are alerted when the site goes down or the certificate is close to expiring. Nothing to install.

Emails (8)

Choose which confirmation emails customers receive and how they read.

08

Public API

For your own software: a POS, a kitchen screen, a spreadsheet, Zapier. Server to server only; the key goes in the header, never the URL.

```
list new orders

curl https://api.linkitecture.com/v1/public/orders?status=new \
  -H "Authorization: Bearer lk_test_YOUR_KEY"
```

Call	Does	Needs
`GET /me`	Check a key works and what it may do	any key
`GET /ordering`	Is online ordering open right now?	orders:read
`GET /orders`	Newest first; pass `next_cursor` back as `cursor`	orders:read
`GET /orders/{id}`	One order with its tracking reference	orders:read
`POST /orders/{id}/transition`	`{ "to": "preparing" }`. Illegal moves return 409	orders:write
`GET /orders/{id}/messages`	The conversation with the customer	messages:read
`POST /orders/{id}/messages`	`{ "body": "On its way" }` to the tracking page	messages:write
`GET /events`	Everything that happened, to catch up	events:read
`POST /sandbox/orders`	Make a fake order to practise with	orders:write, test key
`DELETE /sandbox/orders`	Clear all test orders	orders:write, test key

Test first

`lk_test_` keys see only sandbox orders, notify nobody and never touch real customers.

Safe retries

Send an `Idempotency-Key` header on POST and DELETE. A repeat returns the first answer.

Limits

300 requests a minute per key. Errors come back as `{ "error", "message" }`.

09

Webhooks

Linkitecture POSTs each event to your URL the moment it happens. Add the endpoint in More → Webhooks, choose the events, and copy its `whsec_` secret.

verify.js

```
import crypto from "node:crypto";

// Use the raw request body, before any JSON parsing.
export function verify(raw, headers, secret) {
  const id = headers["webhook-id"], ts = headers["webhook-timestamp"];
  if (Math.abs(Date.now() / 1000 - Number(ts)) > 300) return false; // older than 5 min
  const key = Buffer.from(secret.replace(/^whsec_/, ""), "base64");
  const expected = crypto.createHmac("sha256",
key).update(`${id}.${ts}.${raw}`).digest("base64");
  return headers["webhook-signature"].split(" ").some((s) => s === `v1,${expected}`);
}
```

Answer with any 2xx within 10 seconds. A failed delivery is retried after 1 min, 5 min, 30 min, 2 h, 6 h and 12 h, then marked failed; you can resend it from the app. Use `webhook-id` to ignore duplicates.

every event looks like this

```
{
  "id": "evt_…", "type": "order.created", "created_at": "2026-11-25T09:30:00Z",
  "livemode": true, "workspace_id": "…",
  "subject": { "type": "order", "id": "…" },
  "data": { … }
}
```

Events: contact.created · enquiry.received · order.created · order.updated · order.status_changed · payment.succeeded · payment.failed · payment.recorded · deployment.started · deployment.succeeded · deployment.failed · site.status_changed · message.received · request.submitted · request.status_changed · integration.connected · integration.disconnected · ordering.paused · ordering.resumed

10

AI assistants (MCP)

Give Claude, ChatGPT or any MCP app safe, limited access to one business. It can only do what the key's permissions allow.

1. More → **Connect your AI** → pick the business → **Create a key for your AI**.
2. Name it ("Claude on my Mac"), tick only the permissions it needs, start with a **sandbox** key.
3. Tap **Copy full setup** and paste it into your AI app's MCP settings.

MCP settings

```
{
  "mcpServers": {
    "linkitecture": {
      "type": "http",
      "url": "https://api.linkitecture.com/v1/mcp",
      "headers": { "Authorization": "Bearer lk_test_YOUR_KEY" }
    }
  }
}
```

Things to ask it

"Which orders are still waiting to be accepted?"

"Mark order 1042 as ready and tell the customer it's at the counter."

"Summarise today's enquiries and who still needs a reply."

"Is online ordering open right now?"

11

Copy-paste prompts

For Claude Code, Cursor, Copilot or any AI coding tool. Replace the parts in capitals. The app has the same prompts with the client's real key filled in (Connect → "AI" tab).

Connect a plain website

```
You are working on a website. Connect its contact and booking forms to Linkitecture.
1. Add this tag before </body> on every page:
<script src="https://app.linkitecture.com/sdk/linkitecture.js" data-key="pk_live_YOUR_KEY"
data-auto-capture defer></script>
2. Add data-linkitecture-form="<form name>" to every contact, booking and quote form (e.g.
"contact", "booking").
3. Make sure the inputs are named name, email, phone and message where they exist. Don't rename
fields the site's own code relies on; add hidden inputs instead if needed.
4. Don't change how the forms submit or look. Don't add any other tracking.
5. Tell me which forms you changed.
```

Send paid orders from the checkout

```
Add Linkitecture order notifications to this project's checkout.
- After payment succeeds on the SERVER (webhook or success handler), send the order to
Linkitecture with @linkitecture/sdk/server createServerClient({ key:
process.env.LINKITECTURE_KEY, secret: process.env.LINKITECTURE_SECRET }).
- Call linkitecture.order({ external_ref, customer_name, customer_email, fulfilment: "pickup" |
"delivery" | "dine_in", items: [{ name, quantity, unit_price_cents }], total_cents, currency
}).
- Prices come from our own server data, never from the browser.
- Read the key and secret from environment variables; never commit them and never expose the
secret to the browser.
- If the package isn't installable, sign the request yourself: signature = hex(HMAC_SHA256(key
= hex(SHA256(secret)), timestampMs + "." + rawBody)), headers X-Linkitecture-Key, X-
Linkitecture-Timestamp, X-Linkitecture-Signature, POST to
https://api.linkitecture.com/v1/ingest.
- A failure to notify must never fail the customer's payment. Log it and carry on.
```

12

More prompts

Build a webhook receiver

```
Build an HTTPS endpoint that receives Linkitecture webhooks.
- Read the RAW body. Verify: base64(HMAC-SHA256(base64decode(secret without "whsec_"), webhook-id + "." + webhook-timestamp + "." + rawBody)) must equal one of the "v1,<sig>" values in webhook-signature. Reject timestamps older than 5 minutes.
- The secret comes from the LINKITECTURE_WEBHOOK_SECRET environment variable.
- Answer 200 within 10 seconds; do slow work afterwards. Ignore an event whose webhook-id you've already processed.
- Handle order.created and order.status_changed: <describe what should happen>.
```

Use the Public API

```
Write a small script that uses the Linkitecture Public API (https://api.linkitecture.com/v1/public) with the key in LINKITECTURE_API_KEY (Authorization: Bearer).
- GET /orders?status=new, paging with next_cursor → cursor.
- For each order, POST /orders/{id}/transition with { "to": "preparing" } and an Idempotency-Key header of "prep-<id>".
- Then POST /orders/{id}/messages with { "body": "We're on it." }.
- Use an lk_test_ key and POST /sandbox/orders to create practice orders first. Handle 409 (illegal move) and 429 (slow down) politely.
```

13

Test it, then go live

1. Submit the website's contact form yourself. The enquiry should appear in the client's timeline within a few seconds, with a push alert.
2. Place a test order (or `POST /sandbox/orders` with a test key) and move it through Accept, Preparing and Ready. Watch the tracking page change.
3. Send a test webhook from More → Webhooks and check your receiver answered 2xx.
4. Swap test keys for live ones. Delete any key you no longer use.

If nothing arrives

Form sends but no enquiry	The website address on the key must match exactly: `https://`, same `www` or not, no trailing slash. Make a second key for the other version.
401 from the API	Key missing, mistyped or deleted. `GET /me` tells you which key you're using.
403 from the API	The key lacks the permission for that call. Make a new key with it ticked.
409 invalid_state	That status change isn't allowed from where the order is now (e.g. Ready straight to New).
429	More than 300 calls a minute. Wait and retry.
Webhook marked failed	Your URL answered something other than 2xx, took over 10 seconds, or redirected. Fix it, then resend from the app.
No push alerts	On the owner's phone: Settings → Notifications → Linkitecture → Allow, then More → Notifications in the app.

Not available yet

The npm package, signing in with Linkitecture (OAuth) for other apps, writing menus or bookings through the API, and card payments inside the app. They're planned; this guide will be updated when they ship.

Linkitecture

Questions: support@bookinntime.com · app.linkitecture.com · linkitecture.com