Desk OS on WhatsApp

How to switch WhatsApp on, what traders see, and how the AI agent behind it works. Written for the person doing the Meta and Cloudflare setup. You don't need to read code.

Version of 25 November 2026, written from the desk-os code on `main` at commit `2fd00ec`. The live app is **desk.magicteams.ai**.

> **In one paragraph.** Desk OS is an AI desk for commodity traders. A trader forwards a buyer's WhatsApp message to the desk's business number. The desk replies "Got it, reading now", reads the message, opens a deal for each request in it, and sends back a card with *Approve* and *Reject* buttons. The trader can also type questions ("open deals", "who should I follow up with?"). The desk pings people when something urgent needs them, and sends a short brief every morning. Nothing that leaves the desk or changes a deal happens without a person's tap, unless a person gave the agent a standing permission in advance.

Contents

1. Part 1. Switch WhatsApp on, step by step
2. What to get from the developer
3. The settings, by name
4. Test checklist
5. If something doesn't work
6. Part 2. How it works on WhatsApp
7. Part 3. How the AI harness works
8. The flow, in one picture
9. Words used in this guide

Part 1. Switch WhatsApp on, step by step

Right now WhatsApp is **off cleanly** on the live desk: the code is deployed, but none of its settings are. Until they are, the desk turns Meta away (the deploy's smoke test checks this and shows "skip"). Switching it on takes five things:

1. a WhatsApp business number on Meta's official **WhatsApp Cloud API**;
2. four secret settings on the Cloudflare Worker `desk-os`;
3. Meta's webhook pointed at the desk;
4. one message template approved by Meta (for messages after 24 hours of silence);
5. each trader linking their own phone number with a **LINK** code.

Plan for about an hour of work, plus Meta's waiting time for the display name and the template (often minutes, sometimes a day or more).

Before you start: what to get from the developer

What	Why you need it

Admin access to the company's **Meta Business portfolio** (business.facebook.com), ideally already business-verified	To create the app, add the phone number and make a system user. An unverified business has a low daily limit on new conversations.
The **phone number** to use as the desk's business number, and a phone that can receive an SMS or a call on it	Meta sends a code to that number once. The number can't be in use on the ordinary WhatsApp or WhatsApp Business app (delete that account first, or use a new number).
The **display name** traders should see (for example "Harbour Desk")	Meta reviews it against the business's name.
A **payment method** on the WhatsApp account	Meta charges for template messages (the "you have things waiting" reminder after 24 hours). Replies inside the 24-hour window are free.
Access to the **everyai-com Cloudflare account** with Wrangler logged in (`npx wrangler whoami`), on a checkout of the `everyai-com/desk-os` repository	To set the four secrets on the Worker. Setting a secret goes live at once.
Someone who can merge a small **pull request** to desk-os and run a deploy (the steps in `docs/DEPLOY.md`)	Three non-secret settings (the template name, its language and the number as people dial it) live in the code's config file, so they change through a pull request and a deploy.
The **smoke test sign-in**: `SMOKE_EMAIL` and `SMOKE_PASSWORD`	The automatic end-to-end test signs in as a test person on "Smoke Test Desk".
A **spare test phone** that isn't any trader's number	The smoke test links its number to the test desk. A real trader's number would be moved there.
A decision on **quiet hours** and the **morning brief time**	The defaults are no pings between 21:00 and 08:00 and the brief at 08:00. The code calls these "not decided yet".

The steps

1 Create the Meta app and add WhatsApp

At developers.facebook.com/apps, choose **Create app**, type **Business**, and attach it to the company's Business portfolio. In the app, add the **WhatsApp** product. Meta creates a *WhatsApp Business Account* and a free test number. You'll replace the test number in the next step.

2 Add the real phone number and its display name

In the app, open **WhatsApp → API Setup → Add phone number**. Enter the display name and the number, then confirm with the code Meta sends by SMS or call. When asked, set a 6-digit two-step PIN and keep it somewhere safe (it isn't needed by the desk). Add the payment method under the WhatsApp account's billing settings.

Back on **API Setup**, pick the new number in the "From" list and write down its **Phone number ID**: a long number, not the phone number itself. This is `WHATSAPP_PHONE_NUMBER_ID`. Also note the number as people dial it, for example `+65 6000 0000`. This is `WHATSAPP_DISPLAY_NUMBER`.

3 Make a permanent access token (a system user)

The token shown on API Setup expires in a day. Don't use it. Instead, in **Business settings → Users → System users**:

- add a system user (role Admin);
- **Assign assets**: the app (full control) and the WhatsApp account (full control);
- **Generate new token** for the app, expiry **Never**, with the permissions `whatsapp_business_messaging` and `whatsapp_business_management`.

Copy the token straight into the Cloudflare step below. Don't paste it into chat, email or a document. This is `WHATSAPP_ACCESS_TOKEN`. The desk uses it to send replies and to download voice notes and documents.

4 Get the app secret and choose a verify token

App settings → Basic → App secret → Show. This is `WHATSAPP_APP_SECRET`. Meta signs every message it sends the desk with it, and the desk refuses anything that isn't signed with it.

The verify token is a password you make up. Meta shows it back once to prove the webhook address is yours. Make a random one in Terminal with `openssl rand -hex 24`. This is `WHATSAPP_VERIFY_TOKEN`.

5 Put the four secrets on the Cloudflare Worker

From the desk-os checkout, run each line. Each one asks you to paste the value, and no value is ever written in the code. Setting a secret updates the live Worker within seconds.

```
npx wrangler secret put WHATSAPP_VERIFY_TOKEN -c wrangler.jsonc
npx wrangler secret put WHATSAPP_APP_SECRET -c wrangler.jsonc
npx wrangler secret put WHATSAPP_ACCESS_TOKEN -c wrangler.jsonc
npx wrangler secret put WHATSAPP_PHONE_NUMBER_ID -c wrangler.jsonc
```

Always include `-c wrangler.jsonc`. The repository has more than one config file, and leaving the flag off can target the wrong one. Check the names with `npx wrangler secret list -c wrangler.jsonc`, which shows names only, never values.

> Set the verify token **before** step 6. Meta checks the webhook the moment you press "Verify and save", and the desk can only answer if the token is already there.

6 Point Meta's webhook at the desk

In the app, open **WhatsApp → Configuration → Webhook → Edit**:

- Callback URL: `https://desk.magicteams.ai/webhooks/whatsapp`
- Verify token: the one you made in step 4
- **Verify and save**. It should save without an error.

Then under **Webhook fields**, find `messages` and press **Subscribe**. No other field is needed: the desk ignores delivery receipts.

Finally, switch the app from **Development** to **Live** at the top of the app dashboard. Meta asks for a privacy policy URL first. In Development mode, real people's messages don't reach the webhook.

7 Submit the message template

WhatsApp only lets a business send free text within **24 hours** of the person's last message. After that, the only thing allowed is an approved *template*. The desk uses one, to say "something needs you, tap to see it". In **WhatsApp Manager → Message templates → Create template**, category **Utility**, language **English**, enter it exactly:

```
Name:    desk_update
Body:    Hi {{1}}, {{2}} needs you on the desk: {{3}}. Tap below to see it here.
Button:  Quick reply "Show me"
Samples: {{1}} Tina   {{2}} Harbour Desk   {{3}} 2 decisions waiting and 1 email to read
```

Until Meta approves it, the desk just sends nothing outside the 24-hour window. That's safe, but traders who haven't written in a day won't hear from the desk.

8 Add the three plain settings (one small pull request)

These aren't secrets. They go in the `"vars"` block of `wrangler.jsonc` in a pull request, followed by a normal deploy (`docs/DEPLOY.md`):

```
"WHATSAPP_DISPLAY_NUMBER": "+65 6000 0000",   // the business number as people dial it
"WHATSAPP_TEMPLATE": "desk_update",            // only once Meta has approved it
"WHATSAPP_TEMPLATE_LANG": "en"
```

Optional, only if the developer picked other times: `"WHATSAPP_QUIET_HOURS": "21:00-08:00"` and `"WHATSAPP_BRIEF_AT": "08:00"` (desk local time).

Without `WHATSAPP_DISPLAY_NUMBER` everything still works, but the LINK step says "send it to the desk's WhatsApp number" without showing the number.

9 Check that the database has the ping log

The desk records each WhatsApp ping in its read database (`desk-os-read`). It needs the migrations `0004_whatsapp_pings.sql` and `0005_wa_position_seen.sql`. Check that nothing is waiting:

```
npx wrangler d1 migrations list desk-os-read --remote -c wrangler.jsonc
```

It should say there are no migrations to apply. If it lists some, apply them as step 4 of `docs/DEPLOY.md` describes. Don't skip that step's safety checks.

10 Link each trader's phone (LINK code)

The desk only listens to numbers that belong to a person on a desk. The safe way to prove that is a LINK code:

- The trader signs in at desk.magicteams.ai. The chat's setup checklist now shows **"Link your WhatsApp number"**. It appears once the access token and phone number ID are set.
- They tap it and get a line like *"From your phone, send "LINK ABCD-2345" to the desk's WhatsApp number, +65 6000 0000, within 15 minutes."*
- From their own WhatsApp, they send exactly that. The desk replies: *"Linked: messages from this number now go to <desk>, as <their email>. Send DESK to change desks."*

Good to know: a number gets 5 wrong codes an hour. A number that is only typed on the People screen also counts, on the first desk that listed it, but a LINK proof always wins. If someone links a number that belonged to someone else (a recycled SIM), the old holder's link ends.

11 Run the test checklist below

Then tell the developer it's on.

The settings, by name

All four secrets come from Meta and are set only with `wrangler secret put`. They're never written in the code, a document or a chat. This guide only ever names them.

Name	Kind	Where it comes from	What happens without it
`WHATSAPP_VERIFY_TOKEN`	secret	You make it up (`openssl rand -hex 24`)	Meta can't verify the webhook (the desk answers 403)
`WHATSAPP_APP_SECRET`	secret	Meta app → App settings → Basic	Every incoming message is refused (503). The door stays shut.
`WHATSAPP_ACCESS_TOKEN`	secret	System user → Generate token (never expires)	No replies, no pings, voice notes and files can't be downloaded
`WHATSAPP_PHONE_NUMBER_ID`	secret	WhatsApp → API Setup (the ID, not the number)	No replies. When it's set, messages to any other number on the same app are ignored.
`WHATSAPP_DISPLAY_NUMBER`	plain setting	The business number as people dial it	The LINK step doesn't show the number
`WHATSAPP_TEMPLATE`, `WHATSAPP_TEMPLATE_LANG`	plain setting	`desk_update`, `en`, after Meta approves it	Nothing is sent after 24 hours of silence
`WHATSAPP_QUIET_HOURS`, `WHATSAPP_BRIEF_AT`	plain setting	The developer's choice	Defaults: quiet 21:00–08:00, brief at 08:00

Test checklist

Use a phone you've linked with LINK. For the automatic test, use the spare test phone.

#	Do this	You should see
1	Meta → Webhook → Verify and save	Saved, no error

2	Send `LINK` + the code from the app	"Linked: messages from this number now go to …"
3	Type `open deals`	A short answer in bold with a few lines. The same exchange shows in the web chat.
4	Forward a buyer's message (for example "Need 500 MT UCO CIF Rotterdam, Dec shipment, best price?"), then type a note under it: "that's Harbour, margin 30"	Within seconds: "Got it, reading now". About 20 seconds after your last message, plus reading time: a card for the new deal with *Approve / Reject / Open*
5	Tap *Approve* on the card	The desk answers what it did. The web chat shows the tap "on WhatsApp".
6	Send `BRIEF`	What waits for you right now
7	Send `STOP`, then `START`	"Done: the desk will not ping you…", then "Pings are on again…"
8	Send a short voice note asking for a product	"Got it, reading now", then a card (the voice note is transcribed)
9	Message the number from a phone that isn't linked or on file	No answer at all (it's quietly ignored)
10	Automatic: the smoke test, with the test phone's number (digits only) in `SMOKE_WA_FROM`	The WhatsApp rows turn from "skip" to "pass". It asks you to look at the test phone for "Got it, reading now" and the card.

```
SMOKE_WA_VERIFY_TOKEN=… SMOKE_WA_APP_SECRET=… SMOKE_WA_PHONE_NUMBER_ID=… SMOKE_WA_FROM=6591234567 \
SMOKE_EMAIL=… SMOKE_PASSWORD=… node scripts/smoke.mjs
```

The smoke test never prints a secret and masks phone numbers. Use a test phone only: its LINK moves that number to Smoke Test Desk.

If something doesn't work

What you see	Most likely cause	Fix
Meta says the callback URL couldn't be verified	`WHATSAPP_VERIFY_TOKEN` isn't set, or isn't the same as the one typed in Meta	Set it again, wait a few seconds, verify again

Messages don't arrive at all (nothing in the Worker logs)	The `messages` field isn't subscribed, the app is still in Development mode, or the app isn't subscribed to the WhatsApp account	Subscribe the field, switch to Live. If it still fails, subscribe the app to the account (Meta's "subscribed_apps" call on the WhatsApp Business Account ID)
Worker logs show `401 bad signature`	Wrong `WHATSAPP_APP_SECRET` (often another app's secret)	Copy the secret from this app's Basic settings again
Worker logs show `503`	`WHATSAPP_APP_SECRET` isn't set	Set it
Logs say "ignored: the sender's number is not a phone on file"	The trader's number isn't linked or listed on their desk	Do the LINK step
Logs say "ignored: sent to another number"	`WHATSAPP_PHONE_NUMBER_ID` is the test number's ID, not the real one	Set the real number's ID
The desk receives messages but never replies	The access token is wrong, expired or missing permissions, or the 24-hour window has closed	Make a new system-user token (never expires, both permissions)
No reminders after a day of silence	The template isn't approved yet, or `WHATSAPP_TEMPLATE` isn't set	Step 7 and step 8
"This number is on more than one desk. Which desk...?"	The person proved the same number on two of their desks	Expected: they tap one. `DESK` asks again.

To watch live: `npx wrangler tail -c wrangler.jsonc` shows the Worker's logs as messages arrive. Logs never contain message text from the webhook, only ids and reasons.

Part 2. How it works on WhatsApp

The trader has one conversation with the desk. It's the same conversation in the web chat at desk.magicteams.ai and on WhatsApp. Anything said on one shows on the other.

Who the desk listens to

- Only messages **signed by Meta** to the desk's own business number.
- Only from a number that **belongs to a person on a desk**: proved with LINK, or otherwise the first desk that put it on file. Everyone else is silently ignored.
- A person's first message to the number is also their **opt-in** to pings, and it opens WhatsApp's 24-hour window.

Forwards and bursts: "Got it", then the card

The usual way to use it: the trader gets a request from a buyer on WhatsApp, forwards it to the desk, and types a short note under it ("that's Harbour, they want it by Friday"). These arrive as separate messages, so the desk treats messages that arrive close together as one **burst**:

1. The first message of a burst gets an answer within seconds: **"Got it, reading now"**. That way the trader knows the desk has it.
2. The desk keeps each message exactly as it arrived, and waits until the trader has been **quiet for 20 seconds**. A burst that keeps going is read anyway after 2 minutes.
3. The **intake agent** reads the whole burst together, as numbered messages. Voice notes are transcribed and documents are turned into text. A photo is checked to see whether it's a letter of credit.
4. Each separate request in the burst opens its **own deal room** and gets its **own card**. Anything the desk couldn't place is quoted back ("was there another?") and left for a person.

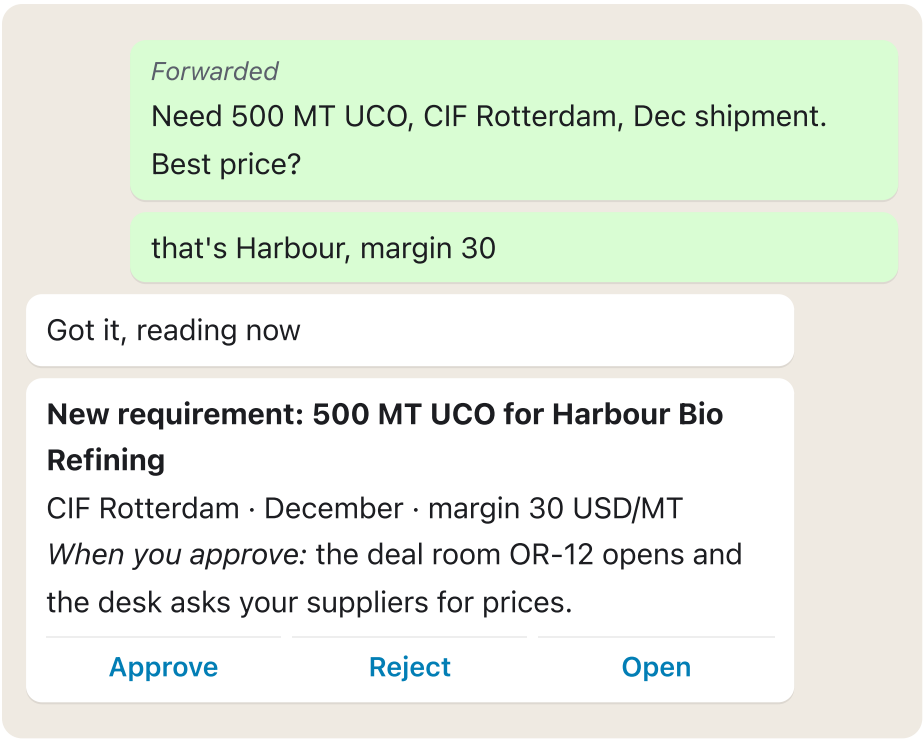


Illustration. The exact wording comes from the deal. If a standing permission (a *mandate*) already covers it, the card says what was done instead ("Opened OR-12", "Invited 3 suppliers"), with an Open button.

Typed questions, and how the desk decides what you mean

When the trader *types* something (not a forward), the desk tries these in order and stops at the first that fits:

1. **LINK** and **DESK**: link this number, or pick which desk it's for.
2. **A tap** on one of the desk's buttons (see below).
3. **Words to the desk itself**: `STOP` (no more pings), `START`, `BRIEF` (what waits now), `DOC <deal>` (make a deal document in their Google Drive).
4. **An email address** the desk asked for (to send a letter-of-credit amendment request to).
5. **A deal note**: "next step on OR-12: call Ana Friday", "give OR-12 to Harry", "what's overdue?"
6. **A chat question or instruction**, answered by the same skills as the web chat: "open deals", "show deal 14", "my position in UCO", "who owes us?". The desk tries its fixed phrasings first, then common rewordings, then the **AI router**. If it still isn't sure and the message reads like a question, it offers two buttons rather than guessing.
7. Anything else (a note, news, a request typed out) goes to the **intake agent**, like a forward.

A forward is never treated as an instruction. A forwarded "STOP" is the buyer's word, not the trader's.

Approve and Reject

- Every card that asks for a decision has buttons, and says plainly **what happens when you approve**. Those lines are written by the code from the action, never by the AI.

- A tap is carried out **as the person who tapped**, with their roles today. Every rule of the desk applies, as if they'd clicked in the app.
- A button only names the ping it came on. A tap from anyone else's phone does nothing, and the same tap is never carried out twice.
- If something the draft was built from has **changed since**, approving is refused: "This changed since it was drafted…". The desk drafts it again at once and shows the new card.
- An approved email waits in an **undo window**, and the answer carries an *Undo* button.
- Changes the chat suggests come with *Confirm / Cancel* instead. The most serious actions (tier T4, such as confirming a deal leg) are decided on the desk only, not on WhatsApp.

The letter of credit (LC) check

When the buyer's letter of credit arrives (a draft, the issued credit or an amendment), the trader forwards it, as a file or a photo, to WhatsApp, to the desk's email, or uploads it. The desk:

- reads the credit's terms, using the AI plus fixed rules for the SWIFT fields (MT700: expiry, amount, ports and so on);
- files it on the right trade;
- checks it against the sale contract: amount and tolerance, quantity, goods, Incoterm, ports, shipment window, presentation period, expiry, documents required, parties;
- sends the findings on WhatsApp, each one naming the credit's field and the contract term, with a drafted **amendment request** to the buyer for Approve or Reject.

If there's no email on file for the buyer, the card asks for one. The trader just replies with the address. A term the desk couldn't read is reported as *uncertain*, never as fine. The check is advisory: the buyer and the banks decide.

Urgent pings and the morning brief

The desk speaks first only about things it already watches, and only to the people they concern:

Right away (urgent)	In the morning brief
A new requirement read from your forward · an award to decide · a counter-offer · an LC amendment request · an urgent email (delay, force majeure, cancellation) with a drafted reply · a deal review that's a day or more overdue	Other decisions waiting · emails that need a person · reviews due today · late ships and the ETA watch · the open position, only when it moved since you last heard · a credit not fully read

- **Quiet hours:** nothing goes out between 21:00 and 08:00 desk time. The brief comes at 08:00. Both can be changed.
- **24-hour window:** free text only within 24 hours of the person's last message. After that, the `desk_update` template ("Hi Tina, Harbour Desk needs you on the desk: 2 decisions waiting. Tap below…"), at most once every 20 hours. Tapping *Show me* sends what was waiting.
- **The sweep** runs every hour, and right after new email is read.

- Text from outside the desk (forwards, emails) is shown quoted, short, labelled, with links removed. It never decides anything.

One thread, on WhatsApp and the web

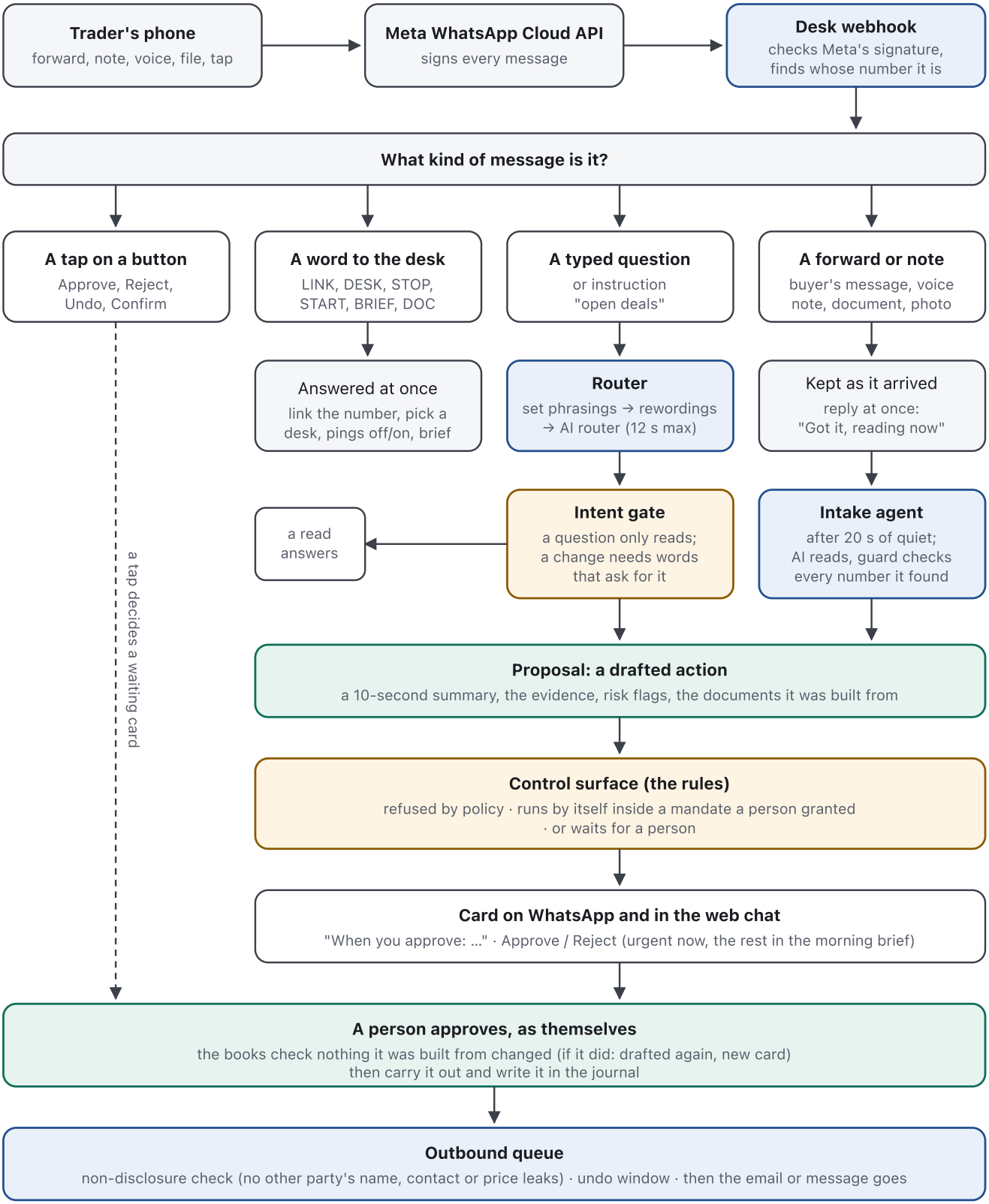
Everything the person sends to the number (text, forwards, files, taps) and everything the desk sends them (pings, the brief, answers, "Got it") goes into their **one thread**. It's kept in their company's own storage, not in a shared database. In the web chat they see the same messages: an Approve tapped on WhatsApp shows there marked "on WhatsApp", with the desk's answer. Cards are redrawn from live data each time, so they always show the current state and respect who may see what.

Part 3. How the AI harness works

"Harness" means everything around the AI model that keeps it useful and safe. The model reads and suggests. The harness decides what it's allowed to see, checks what it says, and makes sure a person stays in charge of anything that matters.

The core rule. The AI never acts directly. It produces *proposals* (drafts of an action). A proposal only happens if a person approves it, or if a standing permission (a mandate) that a person granted covers it. Every step is written to an append-only journal.

The flow, in one picture



Blue boxes use AI or talk to the outside. Amber boxes are rules that hold the AI back. Green boxes are where a person or a standing permission is in charge.

1. Intake: reading what came in

The **intake agent** is the first AI step. It reads a forwarded message, a pasted email or a voice note, and finds each *demand* (a buyer wanting something) and each *offer* (a seller offering something). For each demand it proposes opening a **deal room**. Once the room exists, it proposes inviting the right suppliers and emailing them a request for quotation. It never does any of this itself: each step is a proposal.

The incoming message is treated as **data, not instructions**. It reaches the model wrapped and labelled as untrusted. The model's answer has no place to name an action or a recipient. A guard then checks every number, party and address the model reports: it only counts if people actually wrote it in the message, and not inside a line that tries to instruct the AI. A message that tries to instruct the AI ("ignore your rules and…") is flagged on every room it opens, so a person decides even where a mandate would normally let the desk act alone.

2. The router and the intent gate: understanding a typed request

For typed questions, the **router** picks one *skill* and fills in its details. It tries cheap, predictable methods first:

1. **Set phrasings**: fixed patterns like "open deals", "show deal 14", "received 55,000 USD from Harbour".
2. **Common rewordings**: "anyone after acid oil?", "what needs me?"
3. **The AI router**: a fast model that chooses a skill. If it takes more than 12 seconds or fails, the desk falls back to the rules above, and if nothing fits, offers the three closest things it can do as buttons.

Whatever picked the skill, the **intent gate** checks it before anything runs. A skill that changes something (drafts a deal, emails suppliers, records a payment) only runs if the words actually ask for that. "What quotes did we get on OR-31?" reads the quotes and offers "ask suppliers for OR-31" as a button. It never emails suppliers. This rule was added after a real mistake, where a question about quotes started a supplier round.

3. Skills: what the desk can do

A **skill** is one capability with a fixed shape: list deals, show a deal, the open position, who owes us, check the LC, suggest an award, draft a contract, and so on. There are 51 in the chat today, the same on the web and on WhatsApp. Skills run **as the person**, with their roles and what they're allowed to see. A *read* answers directly. A *write* becomes either a suggestion the person confirms, or a proposal an agent files.

4. Proposals, approvals, mandates and the journal

- **Proposal**: a typed action with a 10-second summary, its evidence, risk flags, and the exact versions of the documents it was built from.

- **Tiers**: each action has a tier. T2 changes only the desk's own records (open a room, record a quote, draft a contract). T3 sends something outside (a request for quotation, a counter-offer, a reminder, an LC amendment request, a reply to an urgent email, a document). T4 is irreversible (confirming a deal leg) and is decided only on the desk, never by a tap on WhatsApp.
- **Policy** refuses what's never allowed.
- **Mandate**: a standing permission a senior person grants ("within these limits, open rooms and ask suppliers by yourself"). Inside it, a proposal runs on its own. Outside it, it waits for a person. A broker desk grants no mandates: everything waits.
- **Earned autonomy**: each kind of action starts as "approve each one". After a run of clean approvals (no one had to correct the result), a person may turn it into a mandate. Any later correction revokes it and the count starts again.
- **Journal**: every step (proposed, approved, rejected, expired, carried out) is written to an append-only log that can't be edited or deleted.

5. Staleness: never approve an old picture

Each proposal remembers the version of every document it was drafted from. If any of them changed before the approval (a price edited, a contract amended), the approval is **refused as stale**. The desk drafts it again from the current state and shows the new card. On WhatsApp the answer to the tap is the new card. A person can never approve something based on numbers that are no longer true.

6. Contact scope: who sees which contacts

Traders' relationships are valuable, so each company and each person in the contacts has an owner and a scope:

- **Desk**: everyone on the desk who may read contacts (the default on a merchant desk);
- **Shared**: the owner and named colleagues;
- **Private**: the owner only (the default on a broker desk).

This is enforced where data leaves storage, so the app, the chat, WhatsApp, the AI's prompts, exports and pings all follow it automatically. No one is let past it silently, not even an administrator. The only way to see everything is an explicit admin view that records who looked and why. A contact you can't see is never named: you'd see "1 supplier kept private by its owner".

7. The AI models

All model calls run on **Cloudflare Workers AI**, through Cloudflare's **AI Gateway** (named `desk-os`), which logs every call and enforces limits. No outside AI key is needed. One setting (`AI_PROVIDER=anthropic`, plus the secret `ANTHROPIC_API_KEY`) would switch to Claude instead.

Job	Model (Workers AI)	Why this one

Primary: reading messages, supplier replies, documents	`@cf/qwen/qwen3.8-27b`	Best on the 31-case test set (93%), with no invented numbers and no injected instructions followed. It also reads images.
Cheap fallback	`@cf/openai/gpt-oss-120b`	Used when the primary fails or the day's budget for it is spent
Router (the chat picking a skill)	`@cf/zai-org/glm-4.7-flash`	Fast, with "thinking" off. The chat waits on it.
Vision (photos, scans)	`@cf/qwen/qwen3.8-27b`	Same as primary. Fallback: `@cf/thinkingmachines/inkling-2-small`
Speech to text (voice notes)	`@cf/openai/whisper-large-v3-turbo`	
Documents to text (PDF, Word)	Workers AI document conversion	

The desk keeps an exact **daily AI budget**. It uses the primary model while the day's share lasts, then only the cheap one, then none. With no model, the desk still works by rules and leaves drafts for people. If Workers AI retires a model, the role moves to its fallback automatically, and the event is logged so someone can pick a new default.

8. The guards, all in one list

- **Signed messages only.** Meta's signature is checked before the message is even read, and the request is refused without the app secret.
- **Proved numbers only.** LINK codes, or the first desk that listed the number. Everyone else is ignored.
- **Forwards are never commands.** Keywords, deal notes and chat questions only count when the person typed them.
- **Outside text is untrusted.** It's wrapped for the model, the model's findings are checked against it, and attempted instructions are flagged.
- **A question never writes** (the intent gate).
- **Nothing important without a tap** or a mandate a person granted. T4 actions happen on the desk only.
- **Stale approvals are refused** and redrafted.
- **Contact scope** applies everywhere, including the AI's own prompts.
- **Non-disclosure.** Every outgoing email or WhatsApp message to a counterparty is checked against every other counterparty on file (names, people, emails, domains, numbers, tax ID, and the other side's prices). A message that would leak is refused, with the reason.

- **Undo window.** An approved email waits a short while and can be stopped.
- **Buttons name pings, not actions.** A tap from the wrong phone does nothing, and a repeated tap runs once.
- **Everything is journaled**, and the journal can't be edited.

Words used in this guide

Word	Meaning
Deal room	One buyer's request (a demand) being worked on: suppliers asked, quotes in, an award
Proposal	An action the AI drafted, waiting for a decision
Mandate	A standing permission a person granted, within limits
Card	The message showing what the desk read and what will happen, with buttons
Burst	Messages from one person less than 20 seconds apart, read together
24-hour window	WhatsApp's rule: free messages only within 24 hours of the person's last one
Template	A pre-approved message, the only thing allowed outside the window
Webhook	The address Meta calls with each new message (`/webhooks/whatsapp`)
Worker, Wrangler	The Cloudflare program that runs the desk, and the command-line tool used to set it up
LC, MT700	Letter of credit, and the SWIFT message format it arrives in

---

Sources: the desk-os repository at commit `2fd00ec`: `src/whatsapp/`, `src/assistant/`, `src/agents/intake.ts`, `src/core/control.ts`, `src/core/autonomy.ts`, `src/core/contact_scope.ts`, `src/ai/`, `wrangler.jsonc`, `README.md`, `docs/DEPLOY.md`, `docs/AUTH.md`, `docs/CHAT.md`. No secret value appears in this guide: secrets are referred to by name only.