Messaging platforms

Telegram

Status: production-ready for bot DMs + groups via grammY. Long polling is the default mode; webhook mode is optional.

Pairing
Default DM policy for Telegram is pairing.

Channel troubleshooting
Cross-channel diagnostics and repair playbooks.

Gateway configuration
Full channel config patterns and examples.

Quick setup

1 **Create the bot token in BotFather**

Open Telegram and chat with **@BotFather** (confirm the handle is exactly `@BotFather`).

Run `/newbot` , follow prompts, and save the token.

2 **Configure token and DM policy**

```
{
  channels: {
    telegram: {
      enabled: true,
      botToken: "123:abc",
      dmPolicy: "pairing",
      groups: { "*": { requireMention: true } },
    },
  },
}
```

Env fallback: `TELEGRAM_BOT_TOKEN=...` (default account only).

3 **Start gateway and approve first DM**

```
openclaw gateway
openclaw pairing list telegram
openclaw pairing approve telegram <CODE>
```

Pairing codes expire after 1 hour.

4 **Add the bot to a group**

Add the bot to your group, then set `channels.telegram.groups` and `groupPolicy` to match your access model.

Token resolution order is account-aware. In practice, config values win over env fallback, and `TELEGRAM_BOT_TOKEN` only applies to the default account.

Telegram side settings

Privacy mode and group visibility

Group permissions

Helpful BotFather toggles

Access control and activation

DM policy **Group policy and allowlists** **Mention behavior**

`channels.telegram.dmPolicy` controls direct message access:

- `pairing` (default)
- `allowlist`
- `open` (requires `allowFrom` to include `"*"`)
- `disabled`

`channels.telegram.allowFrom` accepts numeric Telegram user IDs. `telegram:` / `tg:` prefixes are accepted and normalized. The onboarding wizard accepts `@username` input and resolves it to numeric IDs. If you upgraded and your config contains `@username` allowlist entries, run `openclaw doctor --fix` to resolve them (best-effort; requires a Telegram bot token).

Finding your Telegram user ID

Safer (no third-party bot):

1. DM your bot.
2. Run `openclaw logs --follow` .
3. Read `from.id` .

Official Bot API method:

```
curl "https://api.telegram.org/bot<bot_token>/getUpdates"
```

Third-party method (less private): `@userinfobot` or `@getidsbot`.

Runtime behavior

- Telegram is owned by the gateway process.
- Routing is deterministic: Telegram inbound replies back to Telegram (the model does not pick channels).
- Inbound messages normalize into the shared channel envelope with reply metadata and media placeholders.
- Group sessions are isolated by group ID. Forum topics append `:topic:<threadId>` to keep topics isolated.
- DM messages can carry `message_thread_id`; OpenClaw routes them with thread-aware session keys and preserves thread ID for replies.
- Long polling uses grammY runner with per-chat/per-thread sequencing. Overall runner sink concurrency uses `agents.defaults.maxConcurrent`.
- Telegram Bot API has no read-receipt support (`sendReadReceipts` does not apply).

Feature reference

- Live stream preview (message edits)
- Formatting and HTML fallback
- Native commands and custom commands

Inline buttons

Telegram message actions for agents and automation

Reply threading tags

Forum topics and thread behavior

Audio, video, and stickers

Reaction notifications

Ack reactions

Config writes from Telegram events and commands

Long polling vs webhook

Limits, retry, and CLI targets

Troubleshooting

Bot does not respond to non mention group messages

Bot not seeing group messages at all

Commands work partially or not at all

Polling or network instability

More help: **Channel troubleshooting**.

Telegram config reference pointers

Primary reference:

- `channels.telegram.enabled` : enable/disable channel startup.
- `channels.telegram.botToken` : bot token (BotFather).
- `channels.telegram.tokenFile` : read token from file path.
- `channels.telegram.dmPolicy` : `pairing | allowlist | open | disabled` (default: pairing).
- `channels.telegram.allowFrom` : DM allowlist (numeric Telegram user IDs). `open` requires `"*"` . `openclaw doctor --fix` can resolve legacy `@username` entries to IDs.
- `channels.telegram.groupPolicy` : `open | allowlist | disabled` (default: allowlist).
- `channels.telegram.groupAllowFrom` : group sender allowlist (numeric Telegram user IDs). `openclaw doctor --fix` can resolve legacy `@username` entries to IDs.
- `channels.telegram.groups` : per-group defaults + allowlist (use `"*"` for global defaults).
  - `channels.telegram.groups.<id>.groupPolicy` : per-group override for groupPolicy (`open | allowlist | disabled`).
  - `channels.telegram.groups.<id>.requireMention` : mention gating default.
  - `channels.telegram.groups.<id>.skills` : skill filter (omit = all skills, empty = none).
  - `channels.telegram.groups.<id>.allowFrom` : per-group sender allowlist override.
  - `channels.telegram.groups.<id>.systemPrompt` : extra system prompt for the group.
  - `channels.telegram.groups.<id>.enabled` : disable the group when `false` .
  - `channels.telegram.groups.<id>.topics.<threadId>.*` : per-topic overrides (same fields as group).

- `channels.telegram.groups.<id>.topics.<threadId>.groupPolicy` : per-topic override for groupPolicy (`open | allowlist | disabled`).
- `channels.telegram.groups.<id>.topics.<threadId>.requireMention` : per-topic mention gating override.

- `channels.telegram.capabilities.inlineButtons` : `off | dm | group | all | allowlist` (default: allowlist).
- `channels.telegram.accounts.<account>.capabilities.inlineButtons` : per-account override.
- `channels.telegram.replyToMode` : `off | first | all` (default: `off`).
- `channels.telegram.textChunkLimit` : outbound chunk size (chars).
- `channels.telegram.chunkMode` : `length` (default) or `newline` to split on blank lines (paragraph boundaries) before length chunking.
- `channels.telegram.linkPreview` : toggle link previews for outbound messages (default: true).
- `channels.telegram.streamMode` : `off | partial | block` (live stream preview).
- `channels.telegram.mediaMaxMb` : inbound/outbound media cap (MB).
- `channels.telegram.retry` : retry policy for outbound Telegram API calls (attempts, minDelayMs, maxDelayMs, jitter).
- `channels.telegram.network.autoSelectFamily` : override Node autoSelectFamily (true=enable, false=disable). Defaults to disabled on Node 22 to avoid Happy Eyeballs timeouts.
- `channels.telegram.proxy` : proxy URL for Bot API calls (SOCKS/HTTP).
- `channels.telegram.webhookUrl` : enable webhook mode (requires `channels.telegram.webhookSecret`).
- `channels.telegram.webhookSecret` : webhook secret (required when webhookUrl is set).
- `channels.telegram.webhookPath` : local webhook path (default `/telegram-webhook`).

- `channels.telegram.webhookHost` : local webhook bind host (default `127.0.0.1`).
- `channels.telegram.actions.reactions` : gate Telegram tool reactions.
- `channels.telegram.actions.sendMessage` : gate Telegram tool message sends.
- `channels.telegram.actions.deleteMessage` : gate Telegram tool message deletes.
- `channels.telegram.actions.sticker` : gate Telegram sticker actions — send and search (default: false).
- `channels.telegram.reactionNotifications` : `off | own | all` — control which reactions trigger system events (default: `own` when not set).
- `channels.telegram.reactionLevel` : `off | ack | minimal | extensive` — control agent's reaction capability (default: `minimal` when not set).

[Configuration reference - Telegram]

Telegram-specific high-signal fields:

- startup/auth: `enabled` , `botToken` , `tokenFile` , `accounts.*`
- access control: `dmPolicy` , `allowFrom` , `groupPolicy` , `groupAllowFrom` , `groups` , `groups.*.topics.*`
- command/menu: `commands.native` , `customCommands`
- threading/replies: `replyToMode`
- streaming: `streamMode` (preview), `draftChunk` , `blockStreaming`
- formatting/delivery: `textChunkLimit` , `chunkMode` , `linkPreview` , `responsePrefix`
- media/network: `mediaMaxMb` , `timeoutSeconds` , `retry` , `network.autoSelectFamily` , `proxy`
- webhook: `webhookUrl` , `webhookSecret` , `webhookPath` , `webhookHost`
- actions/capabilities: `capabilities.inlineButtons` , `actions.sendMessage|editMessage|deleteMessage|reactions|sticker`

reactions: `reactionNotifications` , `reactionLevel`

writes/history: `configWrites` , `historyLimit` , `dmHistoryLimit` , `dms.*.historyLimit`

Related

- Pairing
- Channel routing
- Multi-agent routing
- Troubleshooting

‹ WhatsApp Discord ›