Messaging platforms

Slack

Status: production-ready for DMs + channels via Slack app integrations. Default mode is Socket Mode; HTTP Events API mode is also supported.

Pairing
Slack DMs default to pairing mode.

Slash commands
Native command behavior and command catalog.

Channel troubleshooting
Cross-channel diagnostics and repair playbooks.

Quick setup

Socket Mode (default) | HTTP Events API mode

1 **Create Slack app and tokens**

In Slack app settings:

enable **Socket Mode**

create **App Token** (`xapp-...`) with `connections:write`

install app and copy **Bot Token** (`xoxb-...`)

2 **Configure OpenClaw**

›

```
{
  channels: {
    slack: {
      enabled: true,
      mode: "socket",
      appToken: "xapp-...",
      botToken: "xoxb-...",
    },
  },
}
```

Env fallback (default account only):

```
SLACK_APP_TOKEN=xapp-...
SLACK_BOT_TOKEN=xoxb-...
```

3 **Subscribe app events**

Subscribe bot events for:

`app_mention`

`message.channels` , `message.groups` , `message.im` , `message.mpim`

`reaction_added` , `reaction_removed`

`member_joined_channel` , `member_left_channel`

`channel_rename`

`pin_added` , `pin_removed`

Also enable App Home **Messages Tab** for DMs.

4 **Start gateway**

```
openclaw gateway
```

Token model

`botToken` + `appToken` are required for Socket Mode.

HTTP mode requires `botToken` + `signingSecret`.

Config tokens override env fallback.

`SLACK_BOT_TOKEN` / `SLACK_APP_TOKEN` env fallback applies only to the default account.

`userToken` (`xoxp-...`) is config-only (no env fallback) and defaults to read-only behavior (`userTokenReadOnly: true`).

Optional: add `chat:write.customize` if you want outgoing messages to use the active agent identity (custom `username` and icon). `icon_emoji` uses `:emoji_name:` syntax.

> For actions/directory reads, user token can be preferred when configured. For writes, bot token remains preferred; user-token writes are only allowed when `userTokenReadOnly: false` and bot token is unavailable.

Access control and routing

DM policy | Channel policy | Mentions and channel users

`channels.slack.dmPolicy` controls DM access (legacy: `channels.slack.dm.policy`):

- `pairing` (default)
- `allowlist`
- `open` (requires `channels.slack.allowFrom` to include `"*"`; legacy: `channels.slack.dm.allowFrom`)
- `disabled`

DM flags:

- `dm.enabled` (default true)
- `channels.slack.allowFrom` (preferred)
- `dm.allowFrom` (legacy)
- `dm.groupEnabled` (group DMs default false)
- `dm.groupChannels` (optional MPIM allowlist)

Pairing in DMs uses `openclaw pairing approve slack <code>`.

Commands and slash behavior

Native command auto-mode is **off** for Slack (`commands.native: "auto"` does not enable Slack native commands).

Enable native Slack command handlers with `channels.slack.commands.native: true` (or global `commands.native: true`).

When native commands are enabled, register matching slash commands in Slack (`/<command>` names).

If native commands are not enabled, you can run a single configured slash command via `channels.slack.slashCommand`.

Native arg menus now adapt their rendering strategy:

- up to 5 options: button blocks
- 6-100 options: static select menu
- more than 100 options: external select with async option filtering when interactivity options handlers are available

if encoded option values exceed Slack limits, the flow falls back to buttons

For long option payloads, Slash command argument menus use a confirm dialog before dispatching a selected value.

Default slash command settings:

```
enabled: false
name: "openclaw"
sessionPrefix: "slack:slash"
ephemeral: true
```

Slash sessions use isolated keys:

```
agent:<agentId>:slack:slash:<userId>
```

and still route command execution against the target conversation session (`CommandTargetSessionKey`).

Threading, sessions, and reply tags

- DMs route as `direct` ; channels as `channel` ; MPIMs as `group` .
- With default `session.dmScope=main` , Slack DMs collapse to agent main session.
- Channel sessions: `agent:<agentId>:slack:channel:<channelId>` .
- Thread replies can create thread session suffixes (`:thread:<threadTs>`) when applicable.
- `channels.slack.thread.historyScope` default is `thread` ; `thread.inheritParent` default is `false` .
- `channels.slack.thread.initialHistoryLimit` controls how many existing thread messages are fetched when a new thread session starts (default `20` ; set `0` to disable).

Reply threading controls:

- `channels.slack.replyToMode` : `off|first|all` (default `off`)
- `channels.slack.replyToModeByChatType` : per `direct|group|channel`
  - legacy fallback for direct chats: `channels.slack.dm.replyToMode`

Manual reply tags are supported:

- `[[reply_to_current]]`
- `[[reply_to:<id>]]`

Note: `replyToMode="off"` disables implicit reply threading. Explicit `[[reply_to_*]]` tags are still honored.

Media, chunking, and delivery

Inbound attachments
Outbound text and files
Delivery targets

Actions and gates

Slack actions are controlled by `channels.slack.actions.*` .

Available action groups in current Slack tooling:

Group	Default
messages	enabled
reactions	enabled
pins	enabled

Group	Default
memberInfo	enabled
emojiList	enabled

Events and operational behavior

Message edits/deletes/thread broadcasts are mapped into system events.

Reaction add/remove events are mapped into system events.

Member join/leave, channel created/renamed, and pin add/remove events are mapped into system events.

Assistant thread status updates (for “is typing…” indicators in threads) use `assistant.threads.setStatus` and require bot scope `assistant:write`.

`channel_id_changed` can migrate channel config keys when `configWrites` is enabled.

Channel topic/purpose metadata is treated as untrusted context and can be injected into routing context.

Block actions and modal interactions emit structured `Slack interaction: ...` system events with rich payload fields:

- block actions: selected values, labels, picker values, and `workflow_*` metadata
- modal `view_submission` and `view_closed` events with routed channel metadata and form inputs

Ack reactions

`ackReaction` sends an acknowledgement emoji while OpenClaw is processing an inbound message.

Resolution order:

1. `channels.slack.accounts.<accountId>.ackReaction`
2. `channels.slack.ackReaction`
3. `messages.ackReaction`
4. agent identity emoji fallback (`agents.list[].identity.emoji` , else ”👀”)

Notes:

- Slack expects shortcodes (for example `"eyes"`).
- Use `""` to disable the reaction for a channel or account.

Manifest and scope checklist

- Slack app manifest example
- Optional user-token scopes (read operations)

Troubleshooting

- No replies in channels
- DM messages ignored
- Socket mode not connecting
- HTTP mode not receiving events
- Native/slash commands not firing

Text streaming

OpenClaw supports Slack native text streaming via the Agents and AI Apps API.

By default, streaming is enabled. Disable it per account:

```
channels:
  slack:
    streaming: false
```

Requirements

1. Enable **Agents and AI Apps** in your Slack app settings.
2. Ensure the app has the `assistant:write` scope.
3. A reply thread must be available for that message. Thread selection still follows `replyToMode`.

Behavior

First text chunk starts a stream (`chat.startStream`).

Later text chunks append to the same stream (`chat.appendStream`).

End of reply finalizes stream (`chat.stopStream`).

Media and non-text payloads fall back to normal delivery.

If streaming fails mid-reply, OpenClaw falls back to normal delivery for remaining payloads.

Configuration reference pointers

Primary reference:

High-signal Slack fields:

- mode/auth: `mode`, `botToken`, `appToken`, `signingSecret`, `webhookPath`, `accounts.*`
- DM access: `dm.enabled`, `dmPolicy`, `allowFrom` (legacy: `dm.policy`, `dm.allowFrom`), `dm.groupEnabled`, `dm.groupChannels`
- channel access: `groupPolicy`, `channels.*`, `channels.*.users`, `channels.*.requireMention`
- threading/history: `replyToMode`, `replyToModeByChatType`, `thread.*`, `historyLimit`, `dmHistoryLimit`, `dms.*.historyLimit`
- delivery: `textChunkLimit`, `chunkMode`, `mediaMaxMb`
- ops/features: `configWrites`, `commands.native`, `slashCommand.*`, `actions.*`, `userToken`, `userTokenReadOnly`

Related

- Pairing
- Channel routing
- Troubleshooting
- Configuration
- Slash commands

‹ IRC Feishu ›

Powered by mintlify