Messaging platforms

# Google Chat

Status: ready for DMs + spaces via Google Chat API webhooks (HTTP only).

## Quick setup (beginner)

1. Create a Google Cloud project and enable the **Google Chat API**.

   Go to: **Google Chat API Credentials**

   Enable the API if it is not already enabled.

2. Create a **Service Account**:

   Press **Create Credentials** > **Service Account**.

   Name it whatever you want (e.g., `openclaw-chat`).

   Leave permissions blank (press **Continue**).

   Leave principals with access blank (press **Done**).

3. Create and download the **JSON Key**:

   In the list of service accounts, click on the one you just created.

   Go to the **Keys** tab.

   Click **Add Key** > **Create new key**.

   Select **JSON** and press **Create**.

4. Store the downloaded JSON file on your gateway host (e.g., `~/.openclaw/googlechat-service-account.json`).
5. Create a Google Chat app in the **Google Cloud Console Chat Configuration**:
   - Fill in the **Application info**:
     - **App name**: (e.g. `OpenClaw`)
     - **Avatar URL**: (e.g. `https://openclaw.ai/logo.png`)
     - **Description**: (e.g. `Personal AI Assistant`)
   - Enable **Interactive features**.
   - Under **Functionality**, check **Join spaces and group conversations**.
   - Under **Connection settings**, select **HTTP endpoint URL**.
   - Under **Triggers**, select **Use a common HTTP endpoint URL for all triggers** and set it to your gateway's public URL followed by `/googlechat`.
     - *Tip: Run `openclaw status` to find your gateway's public URL.*
   - Under **Visibility**, check **Make this Chat app available to specific people and groups in <Your Domain>**.
   - Enter your email address (e.g. `user@example.com`) in the text box.
   - Click **Save** at the bottom.
6. **Enable the app status**:
   - After saving, **refresh the page**.
   - Look for the **App status** section (usually near the top or bottom after saving).
   - Change the status to **Live - available to users**.
   - Click **Save** again.
7. Configure OpenClaw with the service account path + webhook audience:

Env: `GOOGLE_CHAT_SERVICE_ACCOUNT_FILE=/path/to/service-account.json`

Or config: `channels.googlechat.serviceAccountFile: "/path/to/service-account.json"` .

8. Set the webhook audience type + value (matches your Chat app config).
9. Start the gateway. Google Chat will POST to your webhook path.

## Add to Google Chat

Once the gateway is running and your email is added to the visibility list:

1. Go to **Google Chat**.
2. Click the **+** (plus) icon next to **Direct Messages**.
3. In the search bar (where you usually add people), type the **App name** you configured in the Google Cloud Console.

   **Note**: The bot will *not* appear in the “Marketplace” browse list because it is a private app. You must search for it by name.
4. Select your bot from the results.
5. Click **Add** or **Chat** to start a 1:1 conversation.
6. Send “Hello” to trigger the assistant!

## Public URL (Webhook-only)

Google Chat webhooks require a public HTTPS endpoint. For security, **only expose the `/googlechat` path** to the internet. Keep the OpenClaw dashboard and other sensitive endpoints on your private network.

### Option A: Tailscale Funnel (Recommended)

Use Tailscale Serve for the private dashboard and Funnel for the public webhook path. This keeps `/` private while exposing only `/googlechat`.

1. **Check what address your gateway is bound to:**

```
ss -tlnp | grep 18789
```

Note the IP address (e.g., `127.0.0.1`, `0.0.0.0`, or your Tailscale IP like `100.x.x.x`).

2. **Expose the dashboard to the tailnet only (port 8443):**

```
# If bound to localhost (127.0.0.1 or 0.0.0.0):
tailscale serve --bg --https 8443 http://127.0.0.1:18789

# If bound to Tailscale IP only (e.g., 100.106.161.80):
tailscale serve --bg --https 8443 http://100.106.161.80:18789
```

3. **Expose only the webhook path publicly:**

```
# If bound to localhost (127.0.0.1 or 0.0.0.0):
tailscale funnel --bg --set-path /googlechat http://127.0.0.1:18789/googlecha

# If bound to Tailscale IP only (e.g., 100.106.161.80):
tailscale funnel --bg --set-path /googlechat http://100.106.161.80:18789/goog
```

4. **Authorize the node for Funnel access:** If prompted, visit the authorization URL shown in the output to enable Funnel for this node in your tailnet policy.
5. **Verify the configuration:**

```
tailscale serve status
tailscale funnel status
```

Your public webhook URL will be: `https://<node-name>.<tailnet>.ts.net/googlechat`

Your private dashboard stays tailnet-only: `https://<node-name>.<tailnet>.ts.net:8443/`

Use the public URL (without `:8443`) in the Google Chat app config.

> Note: This configuration persists across reboots. To remove it later, run `tailscale funnel reset` and `tailscale serve reset`.

### Option B: Reverse Proxy (Caddy)

If you use a reverse proxy like Caddy, only proxy the specific path:

```
your-domain.com {
    reverse_proxy /googlechat* localhost:18789
}
```

With this config, any request to `your-domain.com/` will be ignored or returned as 404, while `your-domain.com/googlechat` is safely routed to OpenClaw.

### Option C: Cloudflare Tunnel

Configure your tunnel's ingress rules to only route the webhook path:

- **Path**: `/googlechat` → `http://localhost:18789/googlechat`
- **Default Rule**: HTTP 404 (Not Found)

## How it works

1. Google Chat sends webhook POSTs to the gateway. Each request includes an `Authorization: Bearer <token>` header.
2. OpenClaw verifies the token against the configured `audienceType` + `audience`:

   `audienceType: "app-url"` → audience is your HTTPS webhook URL.

   `audienceType: "project-number"` → audience is the Cloud project number.
3. Messages are routed by space:

   DMs use session key `agent:<agentId>:googlechat:dm:<spaceId>`.

   Spaces use session key `agent:<agentId>:googlechat:group:<spaceId>`.
4. DM access is pairing by default. Unknown senders receive a pairing code; approve with:

   ```
   openclaw pairing approve googlechat <code>
   ```
5. Group spaces require @-mention by default. Use `botUser` if mention detection needs the app's user name.

## Targets

Use these identifiers for delivery and allowlists:

Direct messages: `users/<userId>` (recommended) or raw email `name@example.com` (mutable principal).

Deprecated: `users/<email>` is treated as a user id, not an email allowlist.

Spaces: `spaces/<spaceId>`.

## Config highlights

```
{
  channels: {
    googlechat: {
      enabled: true,
      serviceAccountFile: "/path/to/service-account.json",
      audienceType: "app-url",
      audience: "https://gateway.example.com/googlechat",
      webhookPath: "/googlechat",
      botUser: "users/1234567890", // optional; helps mention detection
      dm: {
        policy: "pairing",
        allowFrom: ["users/1234567890", "name@example.com"],
      },
      groupPolicy: "allowlist",
      groups: {
        "spaces/AAAA": {
          allow: true,
          requireMention: true,
          users: ["users/1234567890"],
          systemPrompt: "Short answers only.",
        },
      },
      actions: { reactions: true },
      typingIndicator: "message",
      mediaMaxMb: 20,
    },
  },
}
```

Notes:

- Service account credentials can also be passed inline with `serviceAccount` (JSON string).
- Default webhook path is `/googlechat` if `webhookPath` isn't set.
- Reactions are available via the `reactions` tool and `channels action` when `actions.reactions` is enabled.

`typingIndicator` supports `none`, `message` (default), and `reaction` (reaction requires user OAuth).

Attachments are downloaded through the Chat API and stored in the media pipeline (size capped by `mediaMaxMb`).

## Troubleshooting

### 405 Method Not Allowed

If Google Cloud Logs Explorer shows errors like:

```
status code: 405, reason phrase: HTTP error response: HTTP/1.1 405 Met
```

This means the webhook handler isn't registered. Common causes:

1. **Channel not configured**: The `channels.googlechat` section is missing from your config. Verify with:

   ```
   openclaw config get channels.googlechat
   ```

   If it returns "Config path not found", add the configuration (see ).

2. **Plugin not enabled**: Check plugin status:

   ```
   openclaw plugins list | grep googlechat
   ```

   If it shows "disabled", add `plugins.entries.googlechat.enabled: true` to your config.

3. **Gateway not restarted**: After adding config, restart the gateway:

```
openclaw gateway restart
```

Verify the channel is running:

```
openclaw channels status
# Should show: Google Chat default: enabled, configured, ...
```

## Other issues

- Check `openclaw channels status --probe` for auth errors or missing audience config.
- If no messages arrive, confirm the Chat app's webhook URL + event subscriptions.
- If mention gating blocks replies, set `botUser` to the app's user resource name and verify `requireMention`.
- Use `openclaw logs --follow` while sending a test message to see if requests reach the gateway.

Related docs:

‹ Feishu Mattermost ›