Sophea Nous

Powered by Sophea AI

Nous Platform Pre-release Administrator Guide

Table of contents

Chapter 1

Welcome

Welcome

This guide explains how workspace admins and IT staff operate **Sophea Nous** as it behaves today. It focuses on the real setup path for connectors, document sets, search configuration, and chat-side verification.

1.1 Who This Guide Is For

- **Workspace administrators** responsible for content onboarding and search validation
- **IT staff** operating Sophea Nous for internal company use
- **Platform owners** maintaining search quality, AI configuration, and connector health

1.2 How Nous Works Today

The most important workflow concepts for admins:

- Connectors index content into the workspace search index.
- File connectors do **not** automatically create starter document sets. Admins create document sets explicitly on the **Document Sets** page to group indexed sources.
- Searchable connector content is available through **All sources** or its connector source. Admins create document sets when users or workspace-created agents need a reusable curated scope.
- **Projects** group chats with shared instructions and files, giving teams a persistent context space separate from one-off sessions.

Admins may also enable published Sophea agents for a workspace from the Portal. Those published agents are different from the workspace-created agents that admins build inside Nous.

A connector and its document set are related, but they are not the same thing. A connector brings content into Nous. A document set is an optional saved scope for search and workspace-created agents.

1.3 Platform Architecture

Sophea Nous is one service in a multi-service platform. The key services relevant to workspace admins are:

- **Portal**: Organizations, workspace creation, authentication, people and Team access, plans and billing, Digital Employee control, LLM routing, and managed connector credentials.
- **Nous**: the AI search and chat interface, connector management, document processing, agents, and actions.

- **RAG service**: the retrieval pipeline that powers search and agent knowledge lookups. Managed by Sophea; no direct admin configuration required.
- **Agent Service**: hosts published Sophea agents (Deep Research and others). Enabled per workspace from the Portal.

Language model configuration (which models are available, rate limits, and billing) is managed entirely in the Portal. The **Nous** admin panel does not expose a dedicated LLM configuration page.

1.4 Navigation Overview

The main rail is entitlement-aware. It can show:

- **Chats** for chat history and new conversations.
- **Projects** for chats with shared instructions and files.
- **Agents** for the agent gallery and member-created agents when the Agents product is enabled.
- **Knowledge** for connectors and document sets.
- **Integrations** for service accounts, Slack Integration, and related tools.
- **Digital Employees** for employee chat and status when that product is enabled.
- **Automations** for the workspace automation builder when that product is enabled.
- **Settings** for workspace administration. Personal settings are available from the avatar menu.

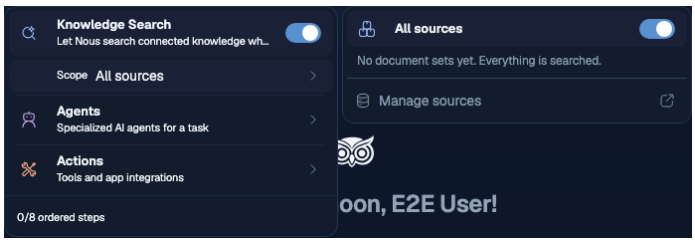

Figure 1.1: Knowledge Search controls open in the chat Tools menu

Workspace admins see the sections they are authorized to manage. Product-specific entries appear only when Portal enables that product. For example, an enabled Agents product adds **Agents** and exposes **MCP Actions** and **OpenAPI Actions** in the Knowledge sidebar.

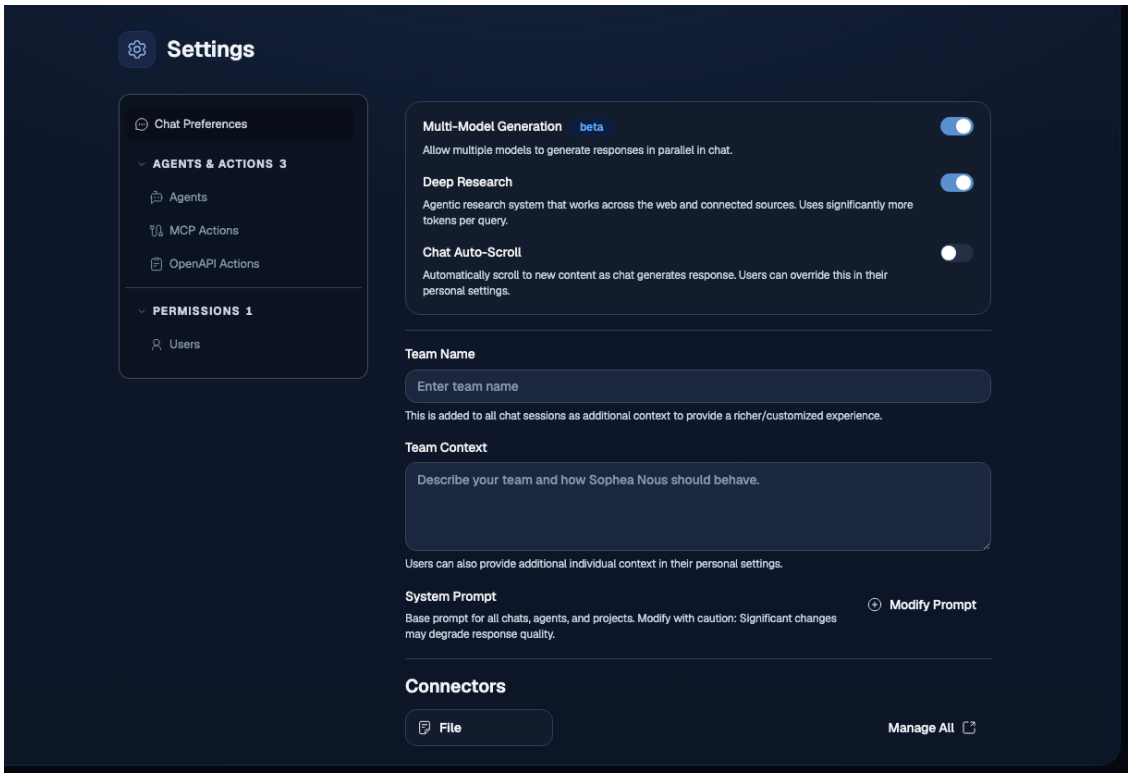

Figure 1.2: Settings panel showing Chat Preferences admin page

> **Missing navigation item**
>
> If an Agents or Digital Employees entry is missing, first check the workspace product settings in Portal. A hidden product is not restored by entering its URL directly.

1.5 Recommended Reading Order

> **Tip**
>
> For a first-time setup, read the guide in this order:
>
> 1. Getting Started
> 2. Organizations and workspace access
> 3. Plans and billing
> 4. Chat Model Governance
> 5. Users and service accounts
> 6. Chat Preferences
> 7. Connectors
> 8. SharePoint Connector
> 9. Google Drive Connector
> 10. Microsoft Teams Connector

11. Slack Connector
12. Confluence Connector
13. Notion Connector
14. Dropbox Connector
15. Gmail Connector
16. Microsoft Outlook Connector
17. ClickUp Connector
18. Jira Connector
19. Linear Connector
20. HubSpot Connector
21. Salesforce Connector
22. Amazon S3 Connector
23. Google Cloud Storage Connector
24. Web Connector
25. Document Sets
26. Knowledge Search
27. Agents
28. Actions and Tools
29. Search Verification and Troubleshooting
30. Digital Employees
31. Automations admin guide

    Discord Bot deprecated
    The Discord Bot integration is no longer available as a managed bot. Discord appears in the connector catalog only as a source for indexing message history into the knowledge base.

1.6 Coming Soon

These features are tracked for a future release and are not part of the current admin workflow:

- **Standard Answers**: pinned Q&A pairs (UI in development).
- **Appearance and Branding**: tenant-side theming (currently managed in Portal).
- **SCIM**: managed via Portal; no tenant-side configuration today.

1.7 Conventions Used

Throughout this guide:

- **Bold text** indicates UI elements (buttons, menus, fields).
- `Code formatting` indicates API endpoints, environment variables, or terminal commands.
- Screenshots show the interface in the language of the guide you are reading. Diagrams are still labelled in English in both languages.

Scope

This is an administrator guide. It covers workspace setup and validation. It does not cover end-user prompting or day-to-day chat usage.

Part I

Part I: Getting Started

Chapter 2

Get started with Sophea Nous

The hosted Sophea Nous platform is invite-only. You start with one of two invitations:

- A **customer setup invitation** lets you create an Organization and its first Workspace.
- A **member invitation** grants a role in an existing Organization, Team, or Workspace.

Use the email address shown in the invitation throughout sign-in. Invitations are tied to that address and cannot be accepted from a different account.

> Important
>
> Opening a member invitation link does not grant access by itself. You must sign in with the invited email address and click **Accept invitation**.

2.1 If you received a customer setup invitation

A customer setup invitation is for the person who will create and initially own the Organization.

1. Open the invitation email and click its **Sign in** link.
2. Sign in with the email address named in the invitation. Complete any identity setup or email verification requested by the sign-in service.
3. Portal opens the **Organizations** page. Click **Create organization**.
4. Enter the company or business-unit name that people should see in Portal, then create the Organization.

Customer setup invitations are valid for 14 days. If the invitation has expired or the invited email is wrong, contact the Sophea team for a replacement.

The person who creates the Organization becomes its Organization owner. In the hosted invite-only platform, only an Organization owner can create a Workspace.

2.2 Create the first Workspace

A Workspace is the boundary for people, data, connectors, settings, and Digital Employees in Nous.

1. Open the Organization in Portal.

2. Select **Create Nous Workspace**.
3. Enter a short, recognizable Workspace name, such as `Acme Legal Operations`.
4. Click **Create**.

Portal lists the Workspace as a row that reads **Setting up** while setup is in progress. You can leave the page and return later without creating another Workspace.

> Warning
>
> Do not create a second Workspace because setup appears slow. If the Workspace remains unavailable, contact Sophea support and provide the Organization and Workspace names.

Selecting a Workspace opens its **Overview**, which shows its Members, Connectors, Digital Employees, and Pending approvals at a glance. To enter Nous directly, select **Open** on the Workspace row, or **Open workspace** on its Overview.

2.3 Workspace allowance

Each Organization has a Workspace allowance. Deleted Workspaces release their allowance, and a failed setup attempt does not consume it.

When the allowance is used, Portal replaces the **Create Nous Workspace** control with a **Workspace limit reached** notice on the Organization and Workspaces pages. Delete a Workspace you no longer need, or contact `billing@sophea.ai` to raise the allowance.

2.4 If you received a member invitation

A member invitation can grant Organization membership, Team membership, or direct access to one Workspace. The invitation email names the scope and role you are being offered.

1. Open the invitation link.
2. On **Accept invitation**, click **Sign in**.
3. Sign in with the exact email address shown on the invitation.
4. Review the invitation and click **Accept invitation**.
5. When Portal confirms acceptance, click **Continue**.

If you are signed in with another account, Portal shows **Switch account**. Switch to the invited address instead of asking the sender to add the wrong account.

Portal may briefly finish applying the access change before **Continue** becomes available. Accepted workspace-only access appears under **Shared with me**.

If an invitation is expired, revoked, or was accepted by another user, Portal shows that it is unavailable. Ask an Organization owner, Organization admin, or Workspace admin with the relevant authority for a replacement. An expired invitation is resent from the list it still appears on; a revoked one needs a new invitation.

For the differences between Organization membership, Teams, and direct Workspace access, see Organizations and Workspace access.

2.5 Reset or add a password

On the Portal sign-in page, select **Forgot password** and enter the email address for your account. Portal shows a neutral confirmation whether or not an account exists for that address. This confirmation does not verify that the account exists.

If the account uses only a passkey, the link in the email lets you add a password without deleting the passkey. If the link has expired or has already been used, return to **Forgot password** and request a new one.

2.6 Open Nous and send a first message

In Portal, select **Open** on the Workspace row (or **Open workspace** on its Overview). Portal signs you into the selected Workspace and opens Nous.

The first visit may ask for your display name in a small card. Answer it or ignore it (it never blocks chat), then send a harmless test message such as:

```
Reply with the single word "ready" so I can confirm chat works.
```

A new Workspace has no connected company data yet. The first response confirms that sign-in, Workspace access, and chat are working. It does not confirm connector or search setup.

If Nous does not open:

1. Return to Portal and confirm the Workspace no longer reads **Setting up** (setup finished).
2. Open the Workspace again from Portal.
3. Confirm that your invitation was accepted and that the Workspace appears in the expected Organization or under **Shared with me**.
4. If access still fails, ask a Workspace admin to check the Workspace **Access** page.

2.7 Know where to manage each setting

Portal and Nous have different responsibilities:

Task	Where to do it
Create Organizations and Workspaces	Portal
Manage Organization people and Teams	Portal
Invite direct Workspace collaborators and review effective access	Portal
Review plans, usage, and billing details	Portal
Create and manage Digital Employees	Portal
Chat with Digital Employees	Nous
Configure connectors, document access, and search	Nous
Chat with Workspace knowledge	Nous

> **Note**
>
> Hosted Workspace invitations are managed in Portal. Do not use the legacy Nous user page to invite or revoke Workspace collaborators.

2.8 Next steps

After the Workspace opens successfully:

1. Review Organizations and Workspace access and add another administrator if needed.
2. Check Plans and billing for the Workspace plan and current usage.
3. Configure a small data source in Connectors.
4. Test the indexed content in Knowledge Search.
5. If the product is enabled for the Workspace, configure a Digital Employee.

Chapter 3

Organizations and Workspace access

Portal is the source of truth for Organization membership, Teams, direct Workspace collaborators, and invitations. Use the **Access** page on each Workspace to understand who can open it and why.

3.1 Understand the scopes

An **Organization** contains its people, Teams, Workspaces, and billing. A **Workspace** contains the Nous experience, data, products, and Workspace-specific settings.

Organization and Workspace roles are separate:

Role	What it grants
Organization owner	Manages Organization people and Teams, receives owner access to every Organization Workspace, and can create Workspaces in the hosted invite-only platform.
Organization admin	Manages Organization people and Teams and receives admin access to every Organization Workspace. It cannot create a Workspace in the hosted invite-only platform.
Organization member	Belongs to the Organization but receives no Workspace access automatically. A direct grant or Team grant is required.
Workspace admin	Manages one Workspace. This role does not grant authority over Organization people or Teams.
Workspace collaborator	Uses one Workspace without Workspace administration rights.

Always include the scope when assigning an admin role. An Organization admin and a Workspace admin have different authority.

3.2 Manage Organization people

Organization people and Teams share one page, **People & Teams**, with two segments, **Teams** and **People**. It is available to Organization owners and Organization admins only. Every other member gets no entry for it and no page: Portal is the administration console, and a member's Organizations list shows their member and Team counts as plain text rather than links.

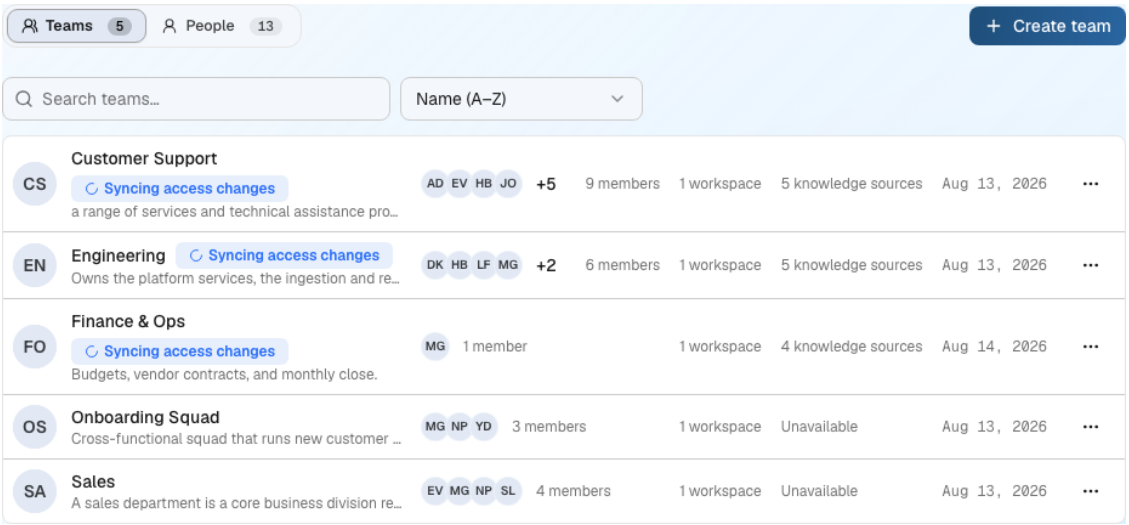

Figure 3.1: People & Teams on the Teams segment, showing the Teams and People counts, the search and sort controls, and one row per Team with its members, Workspaces, and knowledge sources

On a narrow screen the same page keeps each Team's name and member count and drops the columns that do not fit; the Workspace and knowledge-source counts stay on the Team's own page.

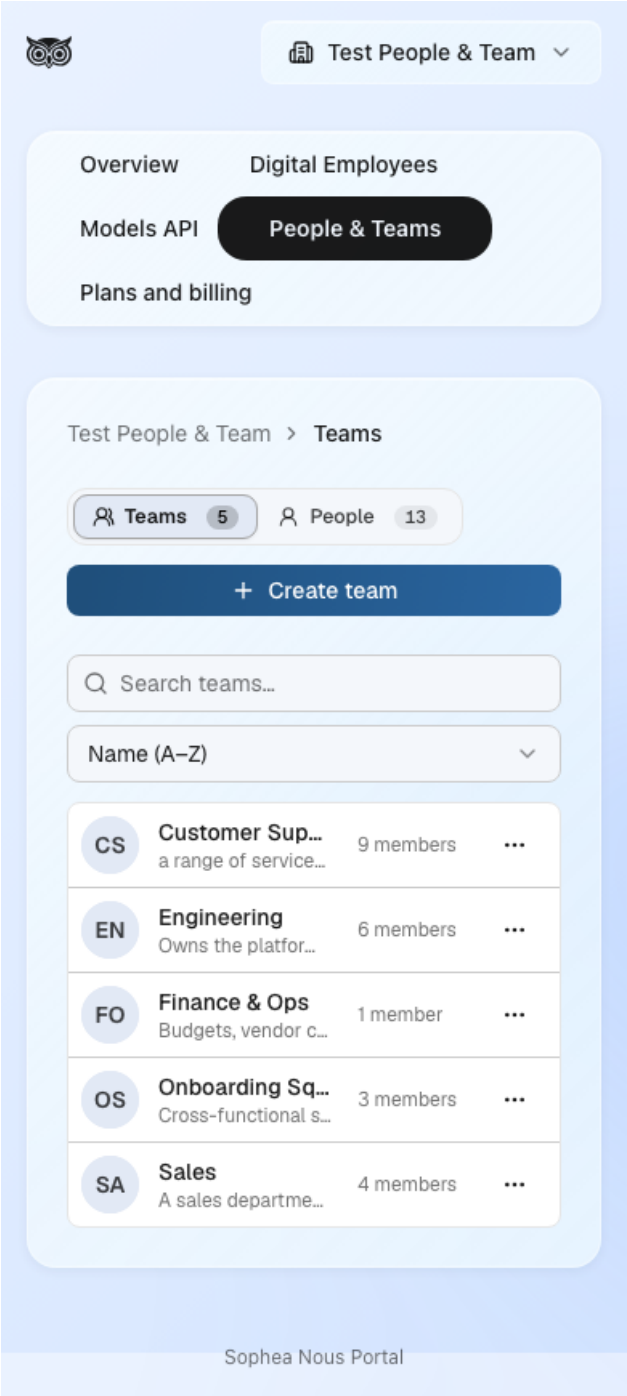

Figure 3.2: The same Teams list on a phone-width screen, each row reduced to the Team name, its description, and its member count

1. Open **Organizations** in Portal and select the Organization.
2. Open **People & Teams** and select the **People** segment.
3. Click **Add people**.

4. On **Invite by email**, type a name or an email address. People already in the Organization are offered from the list; any other address becomes a new invitation.
5. Choose **Member** or **Admin** on each selected person's own role control. Someone who is already in the Organization is marked **Already in organization** and has no role control here; change their role from the roster instead.
6. Click the button that names the outcome, such as **Invite 3 people** or **Add 3 people**.

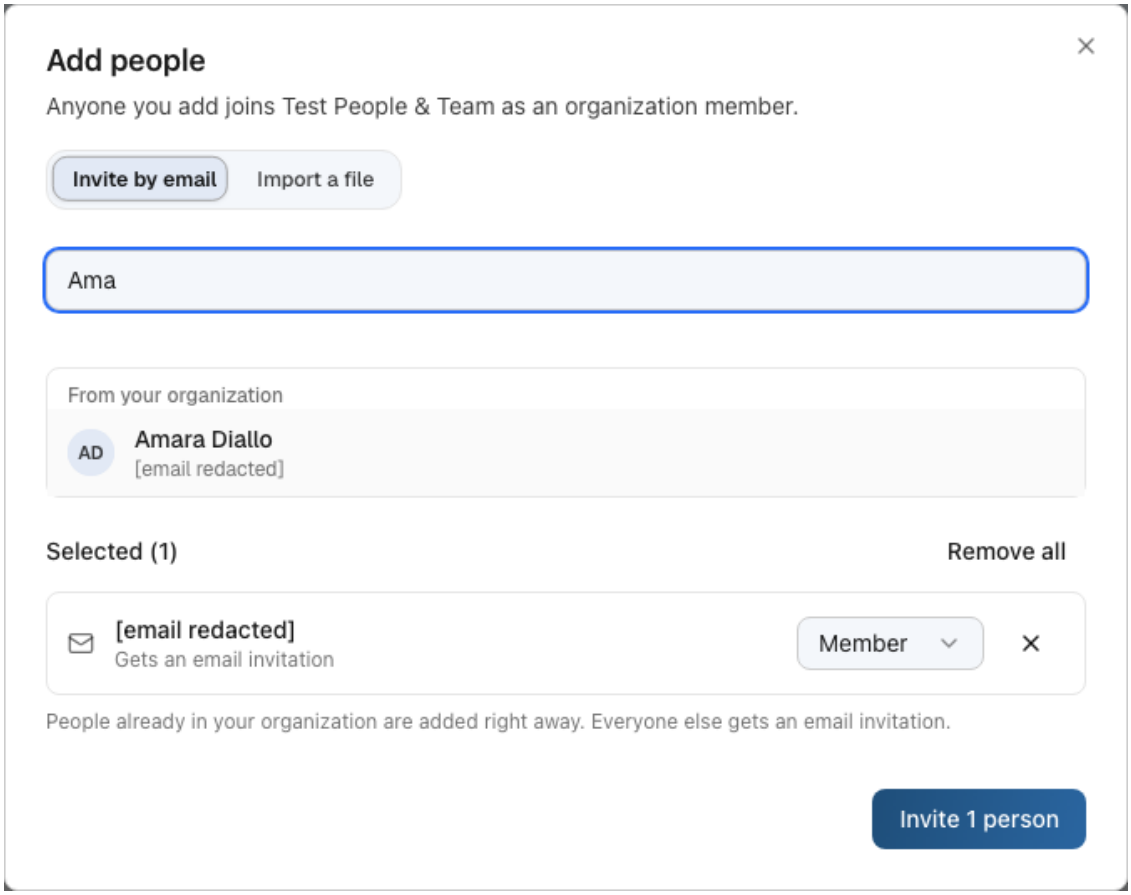

Figure 3.3: The Add people dialog on the Invite by email tab, with a selected person carrying their own Member or Admin role control

To add many people at once, use the **Import a file** tab. Choose a `.csv` or `.xlsx` file with one email address per row in the first column, or click **Download template** to start from the expected shape. Portal accepts up to 100 addresses in one batch and skips blank, repeated, and unusable rows instead of rejecting the file, reporting how many it skipped. Every imported address still gets its own role control before you submit.

The **People** segment lists the active members, with a **Pending invitations** section under the roster. An Organization invitation can also carry Workspace access, and such an entry carries an **Also grants** line naming each Workspace and Team it carries.

The roster also shows an **Added** column, the date the person was added to the Organization, and a **Workspaces** column, listing which Workspaces the person can reach. An Organization owner or admin reaches every Workspace, so that row reads **All workspaces** instead of listing them one by

one. A member's row lists the Workspaces reached through a Team or a direct grant, and is empty when the member has no Workspace access yet.

A Workspace filter sits above the roster next to the existing search box and Team filter, narrowing the list to people who can reach a chosen Workspace. **Clear filters** resets all three at once.

Export to Excel downloads the roster as an `.xlsx` file. The download includes everyone who matches the filters currently applied, not only the rows visible on the page, and the export is recorded in the Organization's audit trail.

An invitation stays in **Pending invitations** until it is accepted or revoked, including after it expires. An expired invitation shows the **Expired** state instead of **Pending**, and active invitations are listed first.

Portal also closes an invitation on its own, in one case: when the person it names already holds everything it would grant. That happens when access arrives by another route, such as making them an Organization owner or admin, adding them to a Team that already has the Workspace, or granting the Workspace directly. There is then nothing left for the invitation to give, so Portal revokes it, stops offering **Resend**, and sends no further email about it. The entry leaves **Pending invitations** without anyone having accepted it, and the closure is recorded in the Organization's audit trail. An invitation that carries more than the person already holds is untouched and stays listed, resendable and revocable.

An Organization owner or Organization admin can use **Resend** and **Revoke** on any listed invitation, expired ones included, and can change an active member's Organization role. The owner role is the exception: only an Organization owner can make somebody an owner, change an owner's role, or remove an owner. An Organization admin manages admins and members, and on an owner's row the role is shown as plain text with no **Remove**, with the reason on hover. Nobody can change or remove themselves, so an Organization always retains at least one owner. To hand over ownership, make the other person an owner first, and they can then remove you. **Revoke** asks for confirmation first and names the Workspace access and Team membership the invitation carries, because revoking withdraws all of it and not only the Organization membership.

Bulk actions build on the same roster. Each row carries a checkbox, and a header checkbox above the table selects every row on the page at once. Some rows are left out of that selection because they cannot be acted on, and the checkbox says why on hover: you cannot select yourself, and an Organization admin cannot select an owner's row, only an Organization owner can.

Selecting one or more people opens a bar above the table naming how many are selected, with a **Change role to…** control and **Remove**. **Change role to…** offers **Member** and **Admin**, and **Owner** only when the signed-in person is an Organization owner, the same rule the roster already applies row by row. **Remove** asks for confirmation first, naming how many people are about to leave the Organization and listing them.

A bulk action on the People roster is all or nothing: if anything in the selection cannot go through, nothing at all changes, and the message names the people to unselect before trying again. There is no partial result to untangle. The selection itself clears whenever the underlying list changes, on a page turn, a filter change, or once an action completes, so it can never end up naming somebody the roster no longer shows.

Resend gives the invitation a new expiry date and sends a fresh email. Inviting the same address again reaches that same invitation instead of creating a second one. Portal sends another email when something about the invitation has to change, such as an expired invitation or a different role, and sends nothing when the invitation is already current. The confirmation reads the same either way, so use **Resend** when you need to be certain that an email went out. After a send, **Resend** is unavailable for a short period and states when it becomes available. That period is longer when Portal could not

confirm what became of the email: the invitation is saved, and the confirmation reports the delivery as unconfirmed rather than as a success or a failure, because an email that may already have arrived must not be sent a second time. Inviting the address again or using **Resend** while that longer period is still running names the same wait.

Bulk actions are available on **Pending invitations** as well. Each entry carries a checkbox, the checkbox above the list selects every listed invitation at once, and selecting one or more opens a bar naming how many are selected, with **Resend** and **Revoke**. **Revoke** applies to the whole selection or to none of it, the same way the People roster's bulk actions do: if one selected invitation cannot be revoked, nothing is revoked, and the message names the invitations to unselect before trying again.

Resend reports each invitation on its own, because it sends an email and an email that has already gone out cannot be taken back. One action can therefore report some invitations sent, some still inside the wait that follows their last send, some whose email Portal could not confirm, and some changed since the page was loaded, such as one that somebody else has already resent or revoked. Read the result rather than assuming that every selected invitation went out; an invitation still inside its wait can be resent once that wait elapses. **Resend** covers at most 25 invitations in one action, because each one is an email to a person. Select more than that and **Resend** turns unavailable and says so, rather than letting you confirm a send that would be refused; **Revoke** stays available, up to 100 at a time.

A search box above **Pending invitations** narrows the list by email address, the same way the search box above the People roster does. It also narrows what the checkbox above the list selects, so **select all** means the invitations you can see and nothing else, and changing what you search for clears the selection rather than leaving it holding invitations the list no longer shows.

> Important
>
> Giving someone the Organization member role does not let that person open every Workspace. Add a direct Workspace grant or include the person in a Team that has a Workspace grant.

3.3 Invite direct Workspace collaborators

Use a direct grant when one person needs access to a specific Workspace without broader Organization membership.

1. Open the Organization and find the Workspace card.
2. Click **Access**.
3. Click **Add people** on Workspace **Access**.
4. Enter or paste one or more email addresses, up to 20 in a single batch.
5. Choose **Member** for a Workspace collaborator or **Admin** for a Workspace admin.
6. Click **Send Invites**.

This Workspace **Add people** dialog grants access only to the current Workspace. The **Add people** dialog on Organization **People & Teams**, illustrated above, manages Organization membership and has separate import controls.

Pending Workspace invitations appear under **Pending invitations** on the same **Access** page and stay listed after they expire, with the **Expired** state. A Workspace admin can revoke them, expired ones included, and can change their role the same way. **Resend** is available on most entries; the two entries further down that withhold it say so and explain why. **Resend** issues a new email and a new expiry date, and is unavailable for a short period after each send. As on the Organization **People & Teams**

page, that period is longer when Portal could not confirm what became of the email. Such a send still gives the invitation its new expiry date, and is reported as unconfirmed rather than as a success or a failure; using **Resend** again before the period elapses names the wait instead.

An address that cannot receive email is refused in the **Add people** dialog before anything is sent, with the message **Enter a valid email address**; this covers reserved domains such as `.local` or `.test`. When the email for an accepted address still cannot be delivered, the confirmation message names that address and the entry is listed as **Not confirmed** or **Not delivered** instead of **Pending**, with a note under it. **Not confirmed** means no email is known to have left for the address, most often because the address itself was rejected; **Resend** would meet the same answer, so revoke the entry and invite the address again, corrected if it was mistyped. **Not delivered** means an email was attempted and did not arrive; Portal tries again on its own, **Resend** is available as soon as no send is under way, and the entry returns to **Pending** when the email goes out.

Each kind of email keeps its own wait. The Organization email and the Workspace email are sent separately, so a wait shown on one entry does not silence the other. Someone you add to an Organization and to a Workspace within the same few minutes therefore receives both emails. Both carry the same invitation, so accepting either one is enough.

To change the role on a listed Workspace invitation without inviting the address again, open its **Invitation actions** menu and choose **Set role to Member** or **Set role to Admin**. Choose **Revoke** in the same menu to open the confirmation dialog. The new role applies when the invitee accepts. Changing it sends no email and does not change the invitation's expiry date.

Workspace invitations differ from Organization invitations in one way. Inviting an address that already holds a listed Workspace invitation never sends a new email, not even when that invitation has expired; Portal reports the address as already invited instead. If the role you choose differs from the role on the listed invitation, the invitation takes the role from the new batch, whether that raises it to **Admin** or lowers it to **Member**, and the confirmation message names the change. Use **Resend** for those, or revoke the invitation and then invite the address again. Revoking and inviting again produces a new invitation only when the invitation carries nothing beyond this Workspace: one that still carries active access to another Workspace, or that also carries Organization membership, stays pending after you revoke its Workspace access. Inviting the address again then restores that access on the same invitation instead of sending a new email.

One entry behaves differently. A single invitation can grant both Organization membership and Workspace access, which happens when someone holding a pending Workspace invitation is then invited to the Organization. That entry is marked **Organization invitation**.

Resend on such an entry resends the Organization invitation, exactly as it would on the Organization **People & Teams** page: it applies the Organization expiry period and sends the Organization email, because that is what the recipient accepts. It is therefore offered only to Organization owners and admins. A Workspace admin who does not hold one of those roles sees no **Resend** on that entry, and a note under the list says so; managing a Workspace does not confer authority over Organization membership.

Everything else on the entry is Workspace work and is available to any Workspace admin. Changing the role is managed here, because the role belongs to the Workspace access rather than to the Organization membership. Revoking here withdraws only the Workspace access; the Organization invitation stays pending on the Organization **People & Teams** page, and the entry is removed from this **Access** page. Inviting the same address to this Workspace again restores the Workspace access on that same invitation, and the confirmation message names the restoration and the role the address now holds. Both the revoke and the restoration are recorded in the Organization's audit trail.

A second entry withholds **Resend**, and it is marked **Sent from another account**. It appears when a

Workspace has changed hands: invitations sent before the move still carry access to the Workspace, but they belong to the account that sent them, and only that account can resend one. A note under the list says so. Changing the role and revoking remain available, because both act on this Workspace's own access. Revoking such an entry withdraws that access and nothing else; when the invitation carried no other access, it is closed as well, and that closure is recorded in the audit trail of the account that sent it rather than in yours.

The invitee must open the invitation, sign in with the invited email, and click **Accept invitation**. Opening the link without accepting it does not grant access.

3.4 Use Teams for repeatable access

A Team groups active Organization members. Team membership has no role by itself. The role is assigned when the Team receives access to a Workspace.

To create and grant a Team:

1. Open the Organization and open **People & Teams**, which opens on the **Teams** segment.
2. Click **Create team**, give it a recognizable name, and add a description if it helps. The description is optional. Team names are unique inside an Organization regardless of capitalization, so a second `Finance` is refused rather than merged into the first.
3. Open the Team, and on **Members** click **Add members**. This uses the same dialog as the Organization, with the same **Invite by email** and **Import a file** tabs. Someone added to a Team who is not yet in the Organization joins it as an ordinary Organization member; a Team never grants an Organization role.
4. Open **Access** and select a Workspace under **Workspace access**.
5. Choose the **Member** or **Admin** role and click **Grant**. To change an existing grant, choose the new role on its entry, or select the Workspace again and click **Update**.

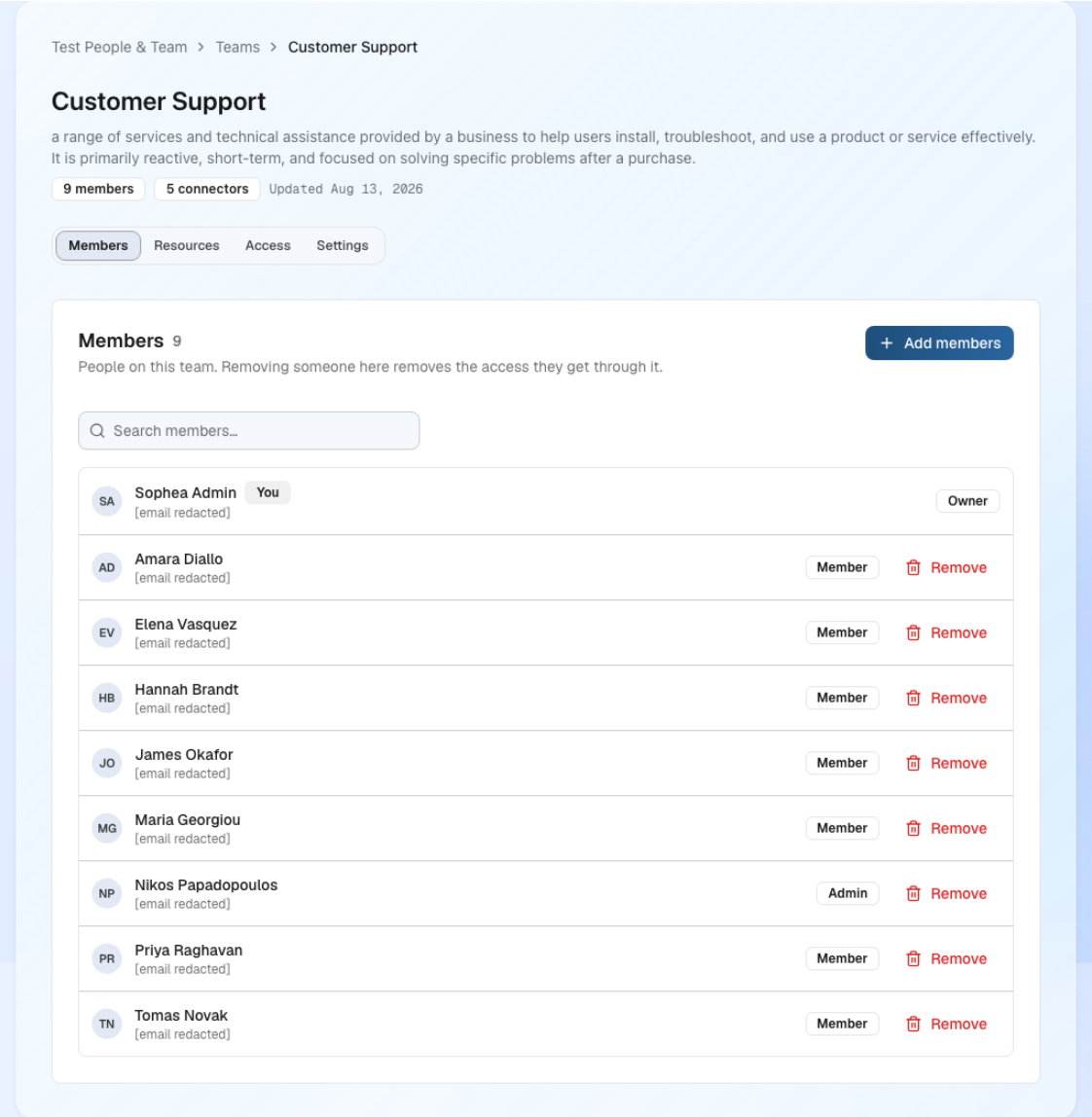

Figure 3.4: A Team on its Members tab, showing the four tabs, the Team description, and the member list with a role column

A Team has four tabs:

- **Members** holds the people on the Team and the Team's own invitations.
- **Resources** is read-only. For each granted Workspace it lists the Connectors, Projects and Document Sets that Workspace shares with the Team, with an **Open connectors** link into the Nous screen where they are managed, and the Digital employees in the Workspaces the Team can reach. A kind with nothing in it says so rather than disappearing, so an empty list and a missing one never look the same.
- **Access** is the only place where the Team's Workspace grants are made, changed, and revoked.
- **Settings** holds the Team name and description, and **Delete team**.

Deleting a Team removes it for everyone and revokes the access it grants. Members keep their ac-

counts, and access they hold through another Team or a direct grant is unaffected. The confirmation names the Team and counts the memberships and Workspace grants it is about to withdraw. There is no restore, so re-create the Team and grant it again if you need it back.

Digital Employees attached to a Team inherit that Team's Workspace knowledge access. The Team's **Resources** view shows this inherited knowledge as read-only. Revoking the Team's Workspace grant denies access to attached employees, but does not change their lifecycle. Regranting the Team's Workspace access restores their access. Portal blocks **Delete team** until every attached Digital Employee is fully deleted.

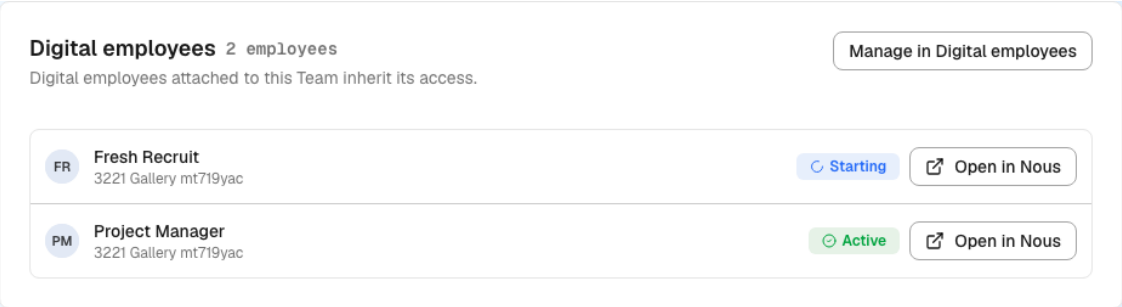

Figure 3.5: Digital Employees attached to a Team on its Resources tab

Every active Team member receives the Team's role for that Workspace. A Team can have different roles in different Workspaces. When someone must accept an invitation before becoming an active Team member, the **Members** tab shows a **Pending invitations** control with the number waiting, and opens that list in a dialog. The control is absent when nothing is pending.

Team invitations behave like Organization invitations. An expired one keeps its place in the list with the **Expired** state, stays resendable and revocable, and is refreshed and sent again if you add the same address to the Team a second time.

Use Teams when access follows a stable business group, such as Legal, Finance, or Customer Support. Use a direct grant for an exception that applies to one person.

> Note
>
> Portal owns Team identity, membership, and Workspace grants. In Nous, share supported resources with a named Portal Team. Do not create a second, parallel group for the same people.

3.5 Review effective access

The Workspace **Access** page separates active access from invitations:

- **Effective access** lists everyone who can currently open the Workspace and every source of that access.
- **Teams with access** lists Teams that grant a Workspace role. It appears for Organization Workspaces.
- **People with direct access** lists direct grants.
- **Pending invitations** lists invitations that have not been accepted, including expired ones. It appears for Workspace managers.

Each area has its own search and count. Searching one area leaves the others unchanged. A filtered count shows how many entries match out of the total; **Clear search** restores the list. Invitations do not count as active access, and matching people keep all their source badges visible.

A person can have several active sources at once:

- Organization authority
- A direct Workspace grant
- One or more Team grants

Portal uses the highest active role across those sources. The order is owner, then admin, then member. A direct Member grant does not reduce an Admin role received from the Organization or a Team.

Before removing access, review the person's source badges in **Effective access**. Removing a direct grant does not remove a Team or Organization grant. Removing someone from one Team does not remove access granted by another Team.

3.6 Change or remove access safely

For a direct grant, open the person's **Direct access actions** menu and choose **Set role to Member**, **Set role to Admin**, or **Remove access**. Removal asks for confirmation. Owner and self-change protections still apply.

Choose the action that matches the source:

Access source	Where to change it
Organization role	Organization **People & Teams**, **People** segment
Direct Workspace grant	Workspace **Access**
Team membership	Organization **People & Teams**, then open the Team and use **Members**
Team-to-Workspace role	Organization **People & Teams**, then open the Team and use **Access**
Pending Organization invitation	Organization **People & Teams**, **People** segment
Pending Workspace invitation	Workspace **Access**
Pending invitation that also grants Organization membership	Workspace **Access** for the Workspace role and to withdraw Workspace access. Resending is available on either Workspace **Access** or Organization **People & Teams**, to Organization owners and admins only

After a change, return to **Effective access** to confirm the result. Access changes can show a short synchronization state before they are available in Nous.

3.7 Management access and document access are different

Workspace roles control who can enter and administer a Workspace. They do not automatically make every private connector, document set, Project, or other shared resource searchable.

Use the sharing controls for each resource to choose **Everyone in this Workspace**, a named Team, or another supported private scope. This separation lets a Workspace admin manage the Workspace without silently receiving access to every private document.

See Connectors for connector document access and Team sharing.

3.8 Switch Workspace inside Nous

Open the user menu from the avatar at the bottom of the left rail in Nous. The Workspace block there names your Organization, the Workspace you are in, and your role in it. If you can reach more than one Workspace, the block opens a list grouped by Organization.

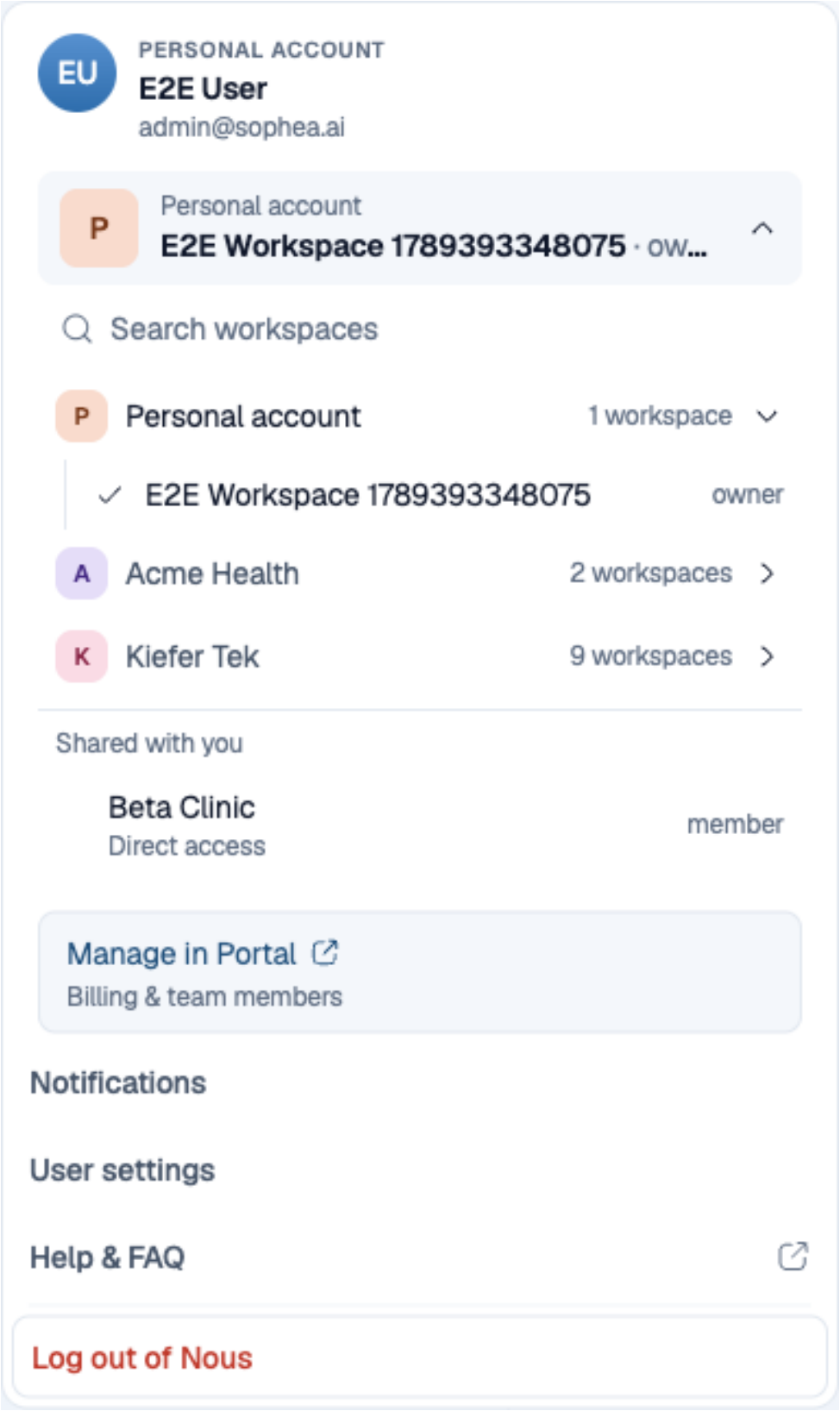

Figure 3.6: The Nous user menu with the Workspace block expanded into an Organization and Workspace list

Selecting an Organization only opens or closes its list. Only selecting a Workspace switches: Nous reloads chats, Projects, agents, and connectors for that Workspace and lands you on a new chat. A Workspace that is still being created or whose setup failed stays in the list with the reason shown and cannot be opened.

If you have typed a message you have not sent, Nous asks you to confirm before switching, because the message is not carried across.

3.9 Find Workspaces shared with you

Direct and Team-derived access to a Workspace outside your Organization membership appears under **Shared with you** in the Nous workspace switcher, and under **Shared with me** in Portal for administrators who still use it.

If an accepted invitation does not appear:

1. Confirm you accepted it while signed in with the invited email.
2. Refresh Portal after the access change finishes.
3. Ask a Workspace admin to verify your entry in **Effective access**.
4. If you rely on a Team, ask an Organization owner or Organization admin to confirm both Team membership and the Team's Workspace grant.

Chapter 4

Plans and billing

Portal brings each Organization's Workspace plans, usage, billing details, invoice records, and plan requests together on **Plans and billing**.

Organization owners and Organization admins can open this page. A Workspace admin who is not also an Organization owner or Organization admin cannot manage Organization billing.

4.1 Open Plans and billing

1. Open **Organizations** in Portal.
2. Select the Organization.
3. Select **Plans and billing** in the top navigation.

The page begins with the current plan state and a summary of:

- Tokens used during active usage periods
- Current seats
- Active Digital Employees
- Pending plan requests

If the Organization has more than one Workspace, the totals combine the current Workspace periods shown below them.

4.2 Review Workspace plans

Each Workspace has its own subscription and plan-change flow. In **Workspace plans**, review:

- The current plan
- Whether the Workspace has an active subscription
- Whether a plan change can be requested

The Organization summary can show that no subscription is active, that setup is incomplete for some Workspaces, or that Workspaces use different plans. Treat those states as account-level summaries and check the individual Workspace rows before requesting a change.

4.3 Review usage

The **Usage** table shows each Workspace's current period, tokens used, and tokens remaining. Use this view for a quick allowance check across the Organization.

The table reflects current platform usage counters. Contact Sophea support if an expected period or Workspace is missing instead of estimating usage from chat counts.

Document ingestion usage follows successful model calls. A call still counts if the document later fails to finish indexing, is cancelled, or its model result cannot be used. Deleting a document does not reverse model work already performed. Reusing completed work does not add another call; repeating the model work does.

Ingestion usage normally appears after the next hourly update. Visual image inputs are reported separately from tokens. Billing remains under development; these usage records do not activate customer charges or usage blocking.

Document ingestion token usage is priced for the Workspace separately from individual members' included allowances. Visual image inputs and visual search queries are counted separately from tokens. Chat, search, memory learning, and agent usage appear after the usage update; a later application failure does not reverse model work that already completed successfully. Usage from Nous and the Models API shares the same member allowance when the user can be matched to a verified active Workspace member.

4.4 Review platform usage reporting

Platform admins can open the separate Portal **Billing > Usage** report. It is an internal reporting surface, not part of an Organization's **Plans and billing** page. Organization owners, Organization admins, and Workspace admins cannot open it unless they also have Platform admin access.

The report has a global view by Workspace and a Workspace drill-down. It keeps LiteLLM usage and direct provider usage as separate source classes. Compatible quantities are added only when their unit is `token`; `visual_input` quantities remain separate. A direct usage event appears once in the report and is not reclassified as a LiteLLM row.

For a direct usage row, the **Provider model** is the actual model or deployment name reported by the provider. The **Price alias** is the stable Sophea service identity used to select the configured price. Different provider model names can therefore use the same price alias and receive the same configured charge. Older history may have no verified provider model; Portal shows **Unknown** and keeps the recorded price evidence. It does not invent a provider name or provider cost.

The report shows the latest event time and source checkpoint. Use these fields to judge whether the data is current before comparing workspaces or periods.

4.4.1 Understand provider cost and modeled customer charge

- **Provider cost** is the cost recorded by the provider source, with its currency and source. It describes provider-side usage and is not the amount charged to a customer.
- **Modeled customer charge** is a read-only estimate based on the usage, the applicable plan and model-rate rules, and the pricing basis shown in the report. It is not a finalized invoice or a payment request.

4.4.2 Understand report states

Display	Meaning
Known zero	The provider cost was verified and its numeric value is `0`. It is not unknown.
`UNKNOWN`	The provider cost is missing or invalid. The amount is left blank instead of being treated as zero.
`PARTIAL`	Some usage or cost details are incomplete, so the aggregate is not a complete fact.
`STALE`	Data is available, but the latest event or source checkpoint is older than the freshness target.
`UNAVAILABLE`	The usage reader could not provide a result. Do not infer zero usage or zero cost.

> Important
>
> The usage report is read-only. It does not charge customers, collect payments, issue legal invoices, deny requests, suspend Workspaces, or enforce usage limits. Use it for operational visibility and reconciliation only. A change to payments or enforcement requires a separate product decision.

4.5 Maintain billing details

The **Billing details** card contains the information used for billing communication and invoice records:

- Legal name
- Trade name
- Billing email
- Country
- VAT number
- Preferred language
- Billing address

Click **Edit**, update the fields, and select **Save profile**. Keep the legal name, country, VAT number, and address aligned with the entity that will receive future invoices.

4.6 Request a plan change

Plan changes are reviewed by the Sophea billing team. Submitting a request does not change the Workspace immediately.

1. Find the Workspace under **Workspace plans**.
2. Click **Request change**.
3. Select the requested plan.
4. Add any timing, procurement, or plan details in **Note**.
5. Click **Submit request**.

The request appears under **Plan requests** with its current status. You can cancel a request while it is **Pending**. Approved, rejected, and canceled requests remain visible as closed history.

> Note
>
> Self-service upgrades are not available yet. For a trial, Team plan, Enterprise agreement, or a request that cannot be submitted from Portal, contact `billing@sophea.ai`.

4.7 Review invoice records

The **Invoices** section lists available billing records by Workspace and period. Select **View** for the detailed record or **Print** for a printer-friendly version.

> Important
>
> Billing is not production-live yet. The current records are for review and do not charge a customer, collect payment, or serve as legally issued PDF invoices. Automated payment and PDF invoice delivery will be documented when those services are available.

If you need a commercial document or have a question about a displayed amount, contact the Sophea billing team. Include the Organization, Workspace, billing period, and record number shown in Portal.

4.8 Troubleshoot billing access

If **Plans and billing** is unavailable:

1. Confirm that you opened the correct Organization.
2. Confirm that your Organization role is owner or admin. A direct Workspace Admin role is not sufficient.
3. Ask an Organization owner to update your Organization role if you need billing authority.

If a Workspace has no active plan or **Request change** is unavailable, contact `billing@sophea.ai`. Do not create another Workspace to work around a plan state.

Part II

Part II: Workspace Setup

Chapter 5

Chat Model Governance

This chapter covers which chat models are visible to your workspace and which one is selected by default.

LLM provider keys (OpenAI, Anthropic, Bedrock, Vertex, and so on) are managed in Sophea Portal, not in Nous. The Nous admin panel never asks you for an API key for chat models.

> Note
>
> Your Portal administrator manages LLM provider keys. To add, rotate, or change provider keys, open the **Models** settings in Sophea Portal. Nous reads the active provider list from Portal automatically.

5.1 Workspace chat model

The visible CHAT models and the default selected for this Workspace are the only model authority for Nous chat. A Workspace administrator can change the existing Workspace default or hide a model in Portal. Account-level defaults affect newly provisioned Workspaces only unless an administrator explicitly applies the change to existing Workspaces.

Nous loads the accessible model list and Workspace default together. A new chat stays disabled until both are current, so it never briefly selects the first model in the list. If a saved chat, agent, or personal preference names a model that is no longer visible, Nous uses the current Workspace default. If no accessible default exists, chat remains disabled and shows a safe configuration message.

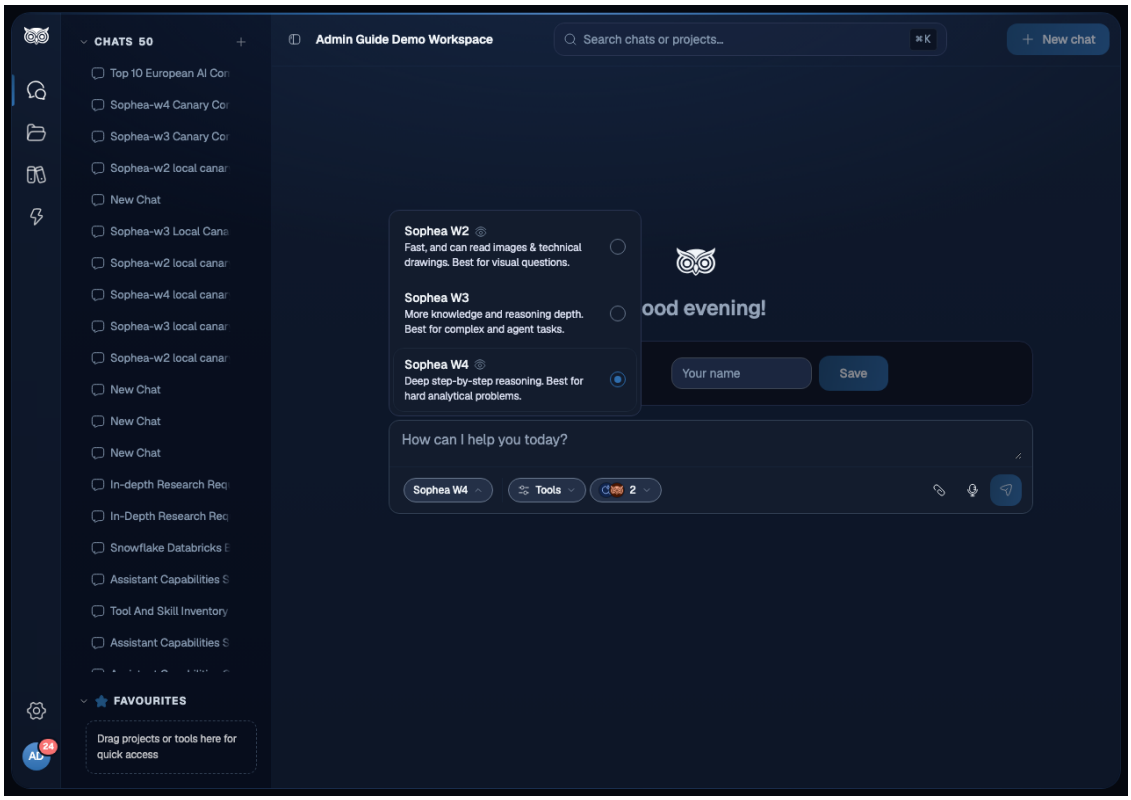

Figure 5.1: Nous model selector showing Sophea W4 as the live Workspace default

Important

Hiding a model is reversible. It removes the model from new selections; it does not delete the provider catalog row. Physically missing or Workspace-foreign model IDs are rejected rather than redirected.

5.2 What this page does not control

Workspace chat model visibility and defaults control which model answers a chat message. They do not control:

- how documents become vectors, how the search index is built, or which reranking model or advanced retrieval options apply. Embedding and reranking configuration is platform-owned and is not exposed in the Nous admin UI.
- which provider keys are used (Portal-managed)
- which users can see which documents (see Users and access and the permissions model in Knowledge Search)
- which document sets exist (see Document Sets)

5.3 What to check next

After the chat model list looks right:

1. Start a new chat and confirm the expected default model loads.
2. If an expected model does not appear, confirm your Portal administrator has enabled it and applied it to this Workspace.
3. Add one small **File** connector through the **Knowledge** flyout > **Add Connector**, then confirm it reaches **Indexed** and returns results in **Knowledge Search**.

See Connectors for the connector workflow and Document Sets for how to group indexed content for chat.

Chapter 6

Users and service accounts

Sophea Portal owns people, invitations, Teams, and workspace access. Nous receives the resulting workspace membership and role so it can authorize chat and administration. Use this page to verify that Portal access has reached Nous and to manage service-account API keys.

For instructions on inviting people, assigning direct workspace access, or granting access through a Team, see Organizations and workspace access.

6.1 Verify synchronized users

Open **Settings** and select **Users**, or navigate to `/app/admin/users`.

The user table shows the people currently known to this Nous workspace, including their email, synchronized role, and account state. Treat it as a verification surface, not the source of truth for access.

> Important
>
> In Portal-managed deployments, Nous does not send workspace invitations, approve join requests, cancel invitations, or own Team membership. Perform those operations on the workspace **Access** page or the Organization **People & Teams** page in Portal.

If Portal access changed but Nous still shows the old state:

1. Confirm the person still has an active access source in Portal.
2. Check direct grants and every Team the person belongs to. Removing one source does not remove access when another source remains.
3. Ask the person to reopen the workspace from Portal so the current account mapping is synchronized.
4. If the state still differs, report the workspace name, user email, expected role, and the access source shown in Portal to Sophea support.

Do not repeatedly invite the person or create a second account. Invitations are bound to the invited email address, and duplicate identities make access harder to audit.

6.2 Understand Nous roles

Portal resolves the effective workspace role before synchronizing the account into Nous.

Nous role	Typical workspace meaning	Capabilities
Admin	Workspace owner or workspace admin	Manage workspace configuration, connectors, document sets, agents, actions, integrations, and synchronized users.
Basic	Workspace member	Use chat, projects, available knowledge, and shared agents. Members can also create their own projects and agents where the product is enabled.
Limited	Restricted service account	Read supported resources and send chat messages without general administration.

Organization roles and effective workspace roles are different. An Organization admin receives workspace-admin authority across that Organization's active workspaces. A direct workspace admin manages only that workspace and does not gain Organization people or Team management.

6.3 Service accounts

Service accounts represent scripts, scheduled jobs, and integrations that call the Nous API without a browser session. They are workspace-scoped and use bearer tokens.

Open **Settings** > **Service Accounts**, or navigate to `/app/admin/service-accounts`.

6.3.1 Create a service account

1. Click **New Service Account**.
2. Enter a name that identifies the system and purpose, for example `Nightly policy check`.
3. Choose **Basic** or **Limited** permissions. Admin service accounts are provisioned through Portal rather than this Nous page.
4. Click **Create Account**.
5. Copy or download the token immediately and store it in a secret manager.

> Warning
>
> The token value is shown once. If you close the dialog before storing it, regenerate the token. Never put service-account tokens in source control, screenshots, chat messages, or plain configuration files.

Send the token as a bearer credential:

```
curl -H "Authorization: Bearer $SOPHEA_SERVICE_TOKEN" \
  https://nous.example.com/api/me
```

6.3.2 Choose the minimum role

- **Admin** can call administrative endpoints. Use it only when the integration must change workspace configuration.
- **Basic** is the default for automation that needs ordinary user-level APIs.
- **Limited** is appropriate for integrations that only need supported read and chat operations.

Create separate service accounts for separate systems. This gives every integration an independent audit identity and lets you rotate or revoke one token without interrupting others.

6.3.3 Rotate or revoke a token

- Use **Regenerate** when a token may be exposed or reaches your rotation date. The previous token stops working immediately.
- Use **Delete Account** when the integration is retired. Existing audit entries remain attributable to that service account.

After every rotation, confirm the calling system uses the new token before considering the operation complete.

Chapter 7

Chat Preferences

This chapter covers the **Chat Preferences** admin page. Use it to customize how the workspace is presented to users in chat, set an org-level system prompt, control which tools are available during chat sessions, and toggle beta features such as Multi-Model Generation and Deep Research.

7.1 Navigate to Chat Preferences

Click the **Settings** gear in the left sidebar, then select **Chat Preferences** under **Settings**. The page is at `/app/admin/configuration/chat-preferences`.

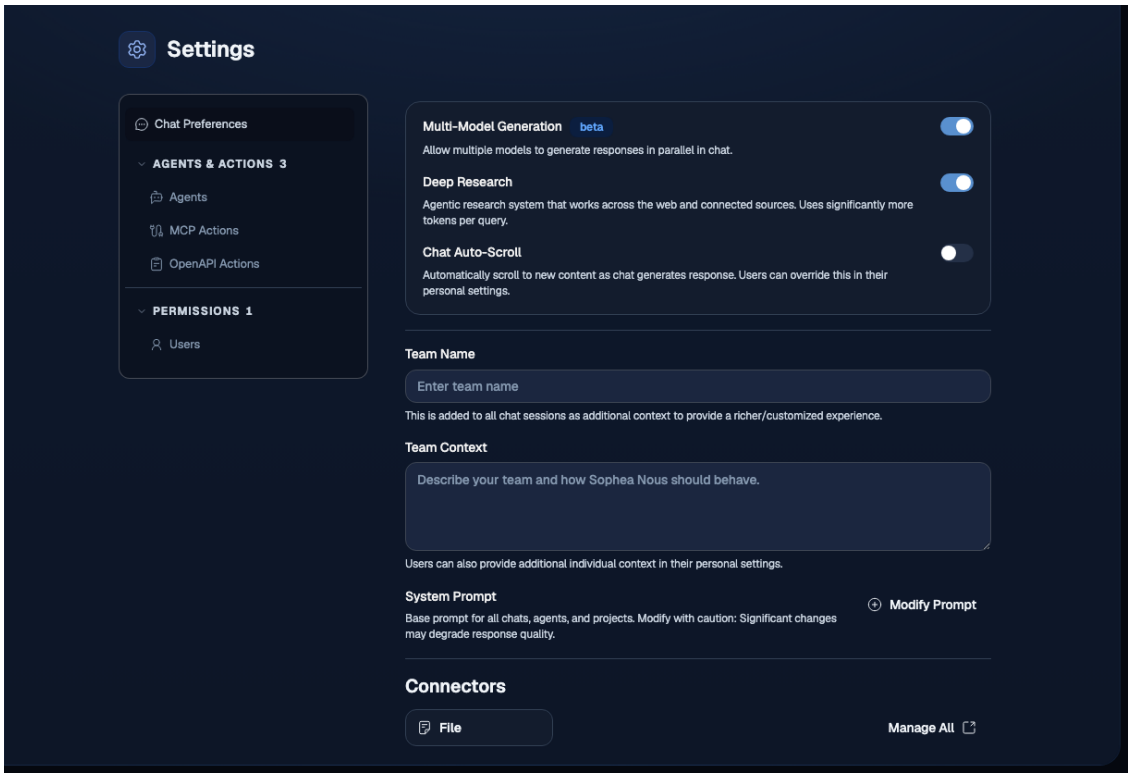

Figure 7.1: Chat Preferences settings page showing toggles for Multi-Model Generation, Deep Research, and Chat Auto-Scroll, plus the Actions & Tools section with Internal Search, Image Generation, and Web Search controls

7.2 Team Name and Team Context

Two fields near the top of the page let you tailor how the workspace is identified inside chat conversations.

Team Name A short label for your organization or team. Nous surfaces this name in the chat welcome experience and in onboarding prompts shown to new users. Keep it to a recognizable name your team actually uses.

Team Context A brief description of what your team does. This text is passed to the LLM as workspace context so responses can be grounded in your organization's domain without you having to repeat it in every chat. Example: `Customer success team at Acme Corp, focused on enterprise SaaS support.`

After updating either field, click **Save** to apply the change. The new values take effect on the next chat session each user opens.

> Tip
>
> A precise **Team Context** reduces the need for users to repeat background in every prompt. Aim for two to four sentences that would help a new colleague understand what your team does and

what kind of questions it asks.

7.3 System Prompt

The **System Prompt** field sets an org-level instruction that Nous prepends to every chat conversation across the workspace. Use it to enforce a tone, restrict topic scope, require citation formats, or set any standing instruction that should apply to all users by default.

Click **Modify Prompt** to open the editor, write or paste your instructions, and save. Users cannot see or override the system prompt from the chat interface.

Warning

The system prompt is prepended to every conversation in the workspace. Keep it focused on standing instructions that genuinely apply to all users. Over-long or overly restrictive prompts can degrade response quality and frustrate users whose tasks fall outside the prompt's assumptions.

Note

Individual users may have their own personal instructions set in their profile settings. The org-level system prompt and a user's personal instructions are both passed to the model. If they conflict, behavior depends on the model in use. Write the system prompt so it complements rather than contradicts common personal instructions.

7.4 End-user personalization and memory

Each user has separate stored information in each workspace. **Saved by you** contains up to 10 memories that the user adds or explicitly asks Sophea to save, correct, or forget. **Learned from chats** contains at most 100 active facts that Sophea can learn automatically from the user's own eligible messages.

Learned facts are copied from one exact user-message evidence span. Sophea keeps only clear personal preferences, profile details, goals, and temporary context. Uncertain statements are dropped. Passwords, tokens, identity and sensitive details, assistant or tool output, operational data, and information about other people are excluded. Workspace and document facts remain in **Knowledge Search**. Raw prior user and assistant message pairs never enter a new prompt as remembered context.

Saved memories can personalize future answers only while **Use saved memories** is on. A memory enters the saved list when the user adds it on this page or explicitly asks Sophea to save or remember it. Sophea corrects or forgets a memory only when the user explicitly asks.

7.5 Published agent artifacts

Chat Preferences does not control the availability of published agents. Portal administrators manage that policy under `external_agents`. When a published agent creates a chart or presentation, the chat

displays a separate artifact card: charts provide a PNG preview plus JSON, SVG, and PNG downloads, while presentations provide available slide thumbnails and PPTX or PDF downloads.

Artifacts are retained for 30 days. An expired artifact link cannot be restored from Chat Preferences; ask Sophea to generate the artifact again. **Chart** and **Presentation** are enabled by default for workspaces that inherit the platform agent policy. They run automatically for clear chart and presentation requests and stay out of the agent picker, so users do not select either agent in the composer. A Portal administrator can still disable either agent globally or for a workspace.

Immediately after Sophea creates a chart, users can ask for a presentation about “it” or “that chart.” Presentation reuses the exact chart data and visual from the preceding turn. If the chart is no longer available or cannot be used safely in the current workspace, Sophea asks the user to create or provide the chart again instead of producing a deck from incomplete data.

Two independent controls create four states:

Use stored information	Learn from my chats	Result
Off	Off	No retrieval and no automatic learning
Off	On	No retrieval and no automatic learning
On	Off	Use saved memories and learned facts, but do not learn new facts
On	On	Use stored information and learn clear facts automatically

Turning either control off does not delete stored information. Users can open **Manage** in every state, including when both controls are off and both lists are empty.

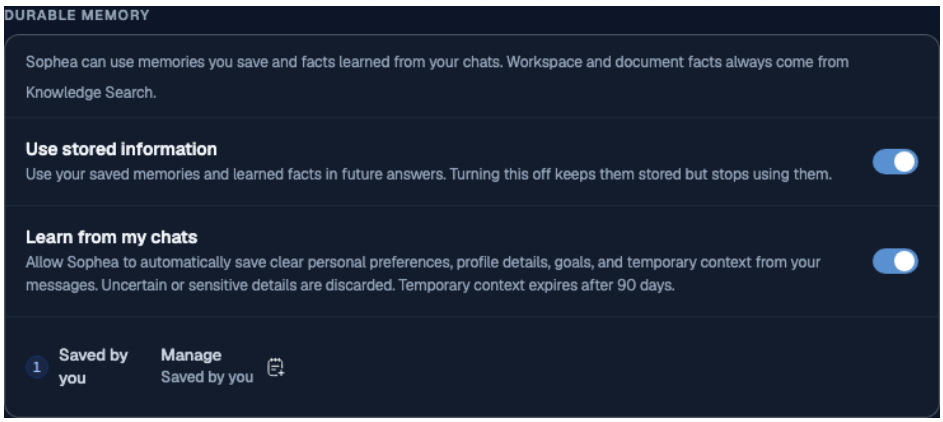

Figure 7.2: Personal Chat Preferences with independent controls for using stored information and learning from chats

The management dialog keeps the two lists separate. Every learned fact shows its category, source, date, and expiry when applicable. A source link appears only while the current user can still access that chat. Deleting a source chat removes an unedited learned fact; a fact the user edited is preserved

without the source link. Users can edit or permanently delete every learned fact even while retrieval or learning is off.

Temporary context expires 90 days after its source message. Editing temporary context confirms it and restarts the 90-day period. The one-time historical backfill covers only the fixed 30-day window and uses the same eligibility, safety checks and processing as live learning.

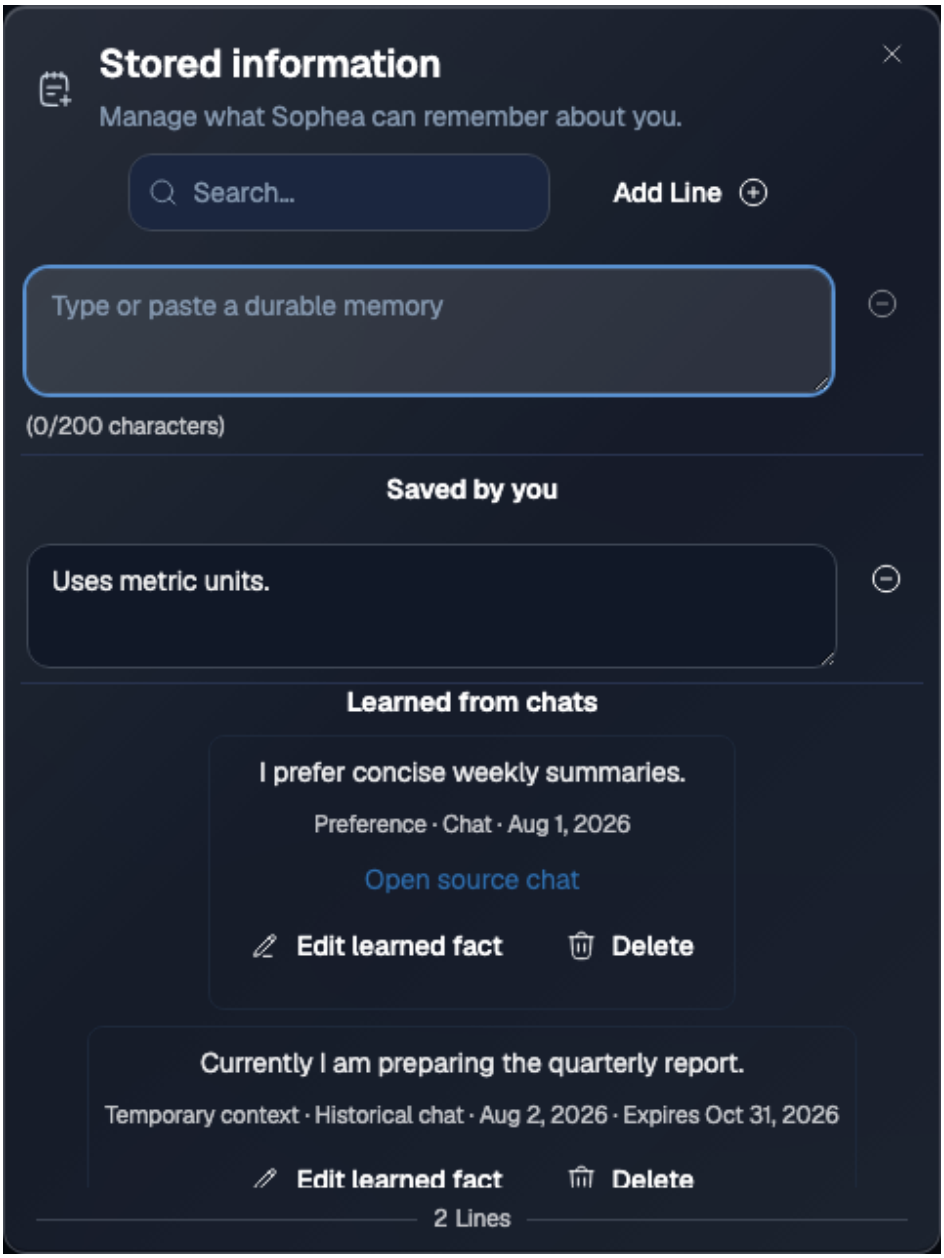

Figure 7.3: Stored information dialog with Saved by you and Learned from chats lists

Saved memories remain limited to 200 characters each. Reaching the 10-item limit does not evict an older memory. Concurrent edits fail closed and reload the authoritative list instead of overwriting

another session.

When Sophea changes a saved memory after an explicit request, the conversation timeline reports the result without repeating private memory text.

Figure 7.4: Content-free conversation status confirming that a memory was updated

> **Note**
>
> Saved memories override learned facts when they conflict. Both groups are data-only user context, never instructions, tool requests, or policy changes.

7.6 Actions and Tools

The **Actions & Tools** section controls which capabilities are available in chat. Toggling a tool off removes it from the chat composer for all workspace users. These settings do not affect agent runs.

These workspace controls are the outer availability boundary. When a capability is available, each user still chooses which apps and actions to expose for each agent in the composer's **Tools** menu. Enabling a capability here does not turn on the corresponding user app or force the model to call it. A user cannot enable a capability that the workspace has turned off.

The built-in **Sophea Nous** app is active on the default assistant by default. Web Search, Open URL, Code Interpreter, Image Generation, and File Reader start enabled. The app is active whenever at least one of its actions is enabled, so its switch and its action switches are the same setting seen from two places. Workspace-created agents make available exactly the built-in actions their creator switched on, and the model decides when to use an attached tool. MCP apps for the default assistant are not configured on this page: an accessible MCP app appears directly in the chat composer and remains off until the user selects it for one submitted message. That selection makes the app's enabled actions available but does not guarantee a call. It clears after a completed response, including one that uses no action, and remains available to **Try again** after a technical failure. Manage MCP registration, authentication, and Private, Team, or Workspace access under **MCP Actions**.

7.6.1 Internal Search

Enables the connected knowledge base as a tool the model can query during chat. When on, the model can search through your organization's indexed documents and connectors to ground answers in internal content.

Turn this off if your workspace uses Nous purely for LLM access and you do not want the model to pull from indexed content automatically.

7.6.2 Image Generation

Enables image generation during chat sessions. When on, users can ask the model to generate images using the configured image generation provider.

> **Note**
>
> Image generation requires an image-capable provider to be configured in Sophea Portal. If no such provider is active, toggling this on has no visible effect in chat.

7.6.3 Web Search

Enables real-time web search as a chat tool. When on, the model can query the web for up-to-date information and include results in its responses.

> **Tip**
>
> Enable **Web Search** for teams that frequently ask time-sensitive questions (market data, recent news, regulatory updates) that your internal knowledge base is unlikely to cover.

7.6.4 Open URL

Enables the model to fetch and read the content of URLs mentioned in chat. When on, users can paste a link and ask the model to summarize or analyze the page.

7.6.5 Code Interpreter

Enables the code interpreter tool during chat. When on, the model can write and execute code to answer quantitative questions, process data, or generate charts inline in the conversation.

Nous loads task-specific guidance only when it is needed. Curated workflows cover spreadsheet, Word document, PDF report, presentation, data-quality, financial-model, energy-analysis, and transcription-comparison requests. Generated files are reopened and checked against their requested structure, including sheet names, formulas, numeric cells, document headings, PDF pages, presentation slides, tables, links, and package integrity. Domain-specific validation also separates supplied inputs from assumptions and distinguishes measured metrics from estimates.

An execution that fails, times out, exceeds the file-size limit, or does not pass structural validation produces no downloadable files. Users should retry the request after correcting the input or execution error rather than treating a partial file as complete.

7.6.6 Exporting a chat answer as CSV or Excel

Separate from Code Interpreter generation, every assistant answer carries an **Export** action that downloads that single stored answer directly, with no new model or retrieval call and no Code Interpreter involvement. The menu always lists **Markdown**, **Word Document**, **CSV** and **Excel Workbook**; choosing a tabular format on an answer that holds no table returns a short message asking for a table instead of downloading an empty file.

- **CSV** downloads a single table. If the answer holds more than one table, CSV asks the user to pick the Excel workbook instead (or narrow to one table), because one CSV file holds one grid.
- **Excel Workbook** writes one table per sheet, so a multi-table answer keeps every table.

Direct export preserves header order, row order, Unicode text (including Greek), quoted commas, embedded line breaks, empty cells, and leading-zero identifiers such as barcodes. Values that look like

spreadsheet formulas are stored as text so they cannot execute. A note on re-import: CSV is text, so a spreadsheet app re-opening it applies its own column detection and may not preserve leading zeros or Greek encoding on every locale; the Excel workbook carries real cell types and is the safer choice when those matter. Excel limits each cell to 32,767 characters and shows numbers beyond 15 significant digits at reduced precision, so very long identifiers in an answer are kept as text in the workbook rather than converted to numbers. Heavily malformed Markdown tables are exported as parsed and may differ from the rendered chat view.

7.7 Beta Features

7.7.1 Multi-Model Generation

Multi-Model Generation (marked **beta**) lets users query multiple LLMs simultaneously from a single chat prompt. The responses from each model are shown side by side, making it easy to compare output across providers.

Toggle it on to make the feature available to all workspace users. Because the feature is in beta, behavior may change across platform releases.

> Note
>
> Multi-model generation sends each prompt to every selected model independently. Token usage is multiplied by the number of models selected. Review your Portal model configuration and cost expectations before enabling this in high-volume workspaces.

7.7.2 Deep Research

Deep Research enables the Deep Research agent feature. When on, users can invoke a research workflow that plans a multi-step investigation, searches internal and external sources, and synthesizes a cited report.

Because Deep Research makes multiple search and fetch calls per session, it consumes significantly more tokens per query than a standard chat turn. Inform your team before enabling it so they understand the cost profile.

Deep Research always runs on the platform research model, regardless of which model a user picked in the chat composer. The composer keeps showing the user's own choice, which still applies to every ordinary chat turn in the same conversation. If the research model is not available to your Workspace, Deep Research reports that it is not configured instead of falling back to another model, because a model that cannot sustain a long report produces an unusable one. Contact your platform administrator if you see that message.

7.8 Chat Auto-Scroll

Chat Auto-Scroll controls whether the chat window scrolls automatically as the model streams its response. When on, the viewport follows the latest output. Users can override this in their personal settings, so the admin toggle sets the workspace default rather than a hard lock.

Turn it off if your users typically scroll up to re-read earlier parts of a long response while generation is still running.

7.9 Save your changes

Each setting on this page takes effect immediately when toggled. Fields that require explicit submission (Team Name, Team Context, System Prompt) show a **Save** button. Settings that do not show a **Save** button apply on toggle.

> Tip
>
> After enabling or disabling a tool, open a fresh chat session and confirm the change is reflected in the chat composer toolbar. New sessions pick up the updated configuration; sessions already open may need a page refresh.

Part III

Part III: Connectors

This is the connector home page. It explains the common lifecycle and access rules. Use the dedicated guide for the provider you want to connect.

Open **Connectors** at `/app/connectors` or open **Add Connector** at `/app/admin/add-connector`.

Connector lifecycle

Every indexed connector follows this path:

1. **Create.** Choose a provider, enter the form fields, and authorize the provider or save customer credentials.
2. **Source sync.** Nous reads the provider and discovers the selected items.
3. **Search indexing.** Nous processes and indexes supported items.
4. **Ready.** Search can return the indexed items. New and changed items are picked up on later refreshes.
5. **Reconnect, reindex, or remove.** Use the connector detail page when a credential expires, a scope changes, or the connector is no longer needed.

Source sync and search indexing are separate. A connector can show that source sync is complete while some documents still wait for search indexing. Check both statuses before testing a new connector.

Waiting for indexing capacity

Busy workspaces share indexing time equally. Each workspace shares its time among connectors with work ready to process. A large import does not need to finish before a new connector can start.

When a required AI service is busy or unavailable, affected documents wait and resume automatically from saved progress. Other documents can continue when the services they need are available. Large scanned files may need several processing turns. Their remaining pages, text recognition, visual content, and metadata still need to finish before indexing is complete.

Leave the connector in place while it waits. Repeated restarts or reindexing do not add capacity. Check the document counts and errors on its detail page. If a document reports a failure that needs action, resolve the stated cause before retrying it.

When the detail page says **Waiting for model capacity**, the pending count shows documents waiting for a shared model service. The page identifies the waiting steps in plain language, such as text indexing, text recognition, or image indexing. **Retrying automatically** means the connector will continue from its saved progress when capacity returns. Pausing the connector removes that automatic-retry guidance until you resume it.

If a file makes no indexing progress for two hours while waiting for model capacity, automatic retries stop. Resolve the issue, then retry indexing from the connector page.

Access and sharing

Choose the document access that matches the business need:

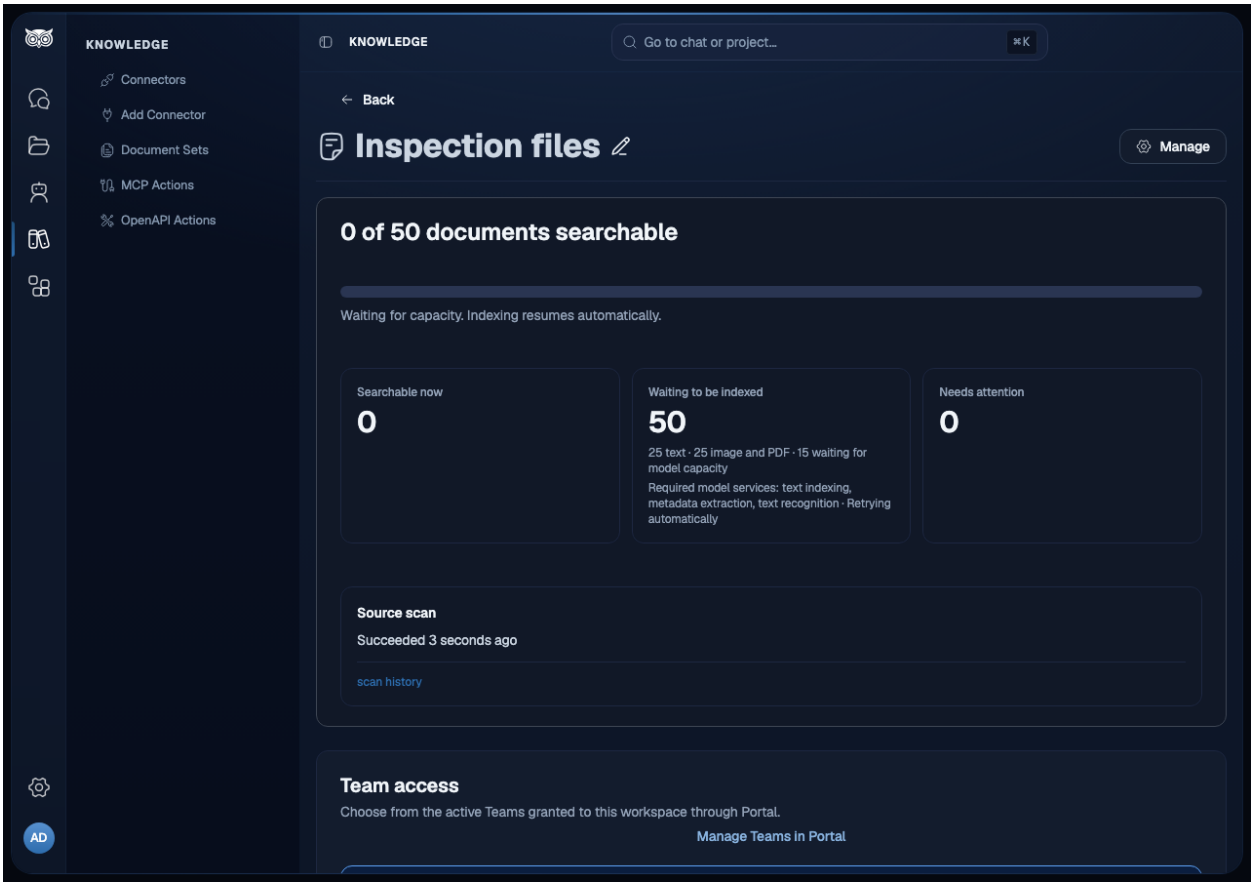

Figure 7.5: A connector waiting for model capacity, with the affected document count and automatic-retry guidance.

Access	Result
Everyone in the workspace	Every workspace member can search the indexed content.
Specific Teams	The content is available only to the selected active Portal Teams.
Sync permissions	Supported SharePoint and Google Drive credentials copy source permissions into search.

Manage Team membership in Portal. Manage the connector and its Team bindings in Nous. Choosing **Specific Teams** does not make a connector public. A document set can narrow access, but it cannot bypass the connector access rules. See Document Sets.

For SharePoint, permission sync starts after the first document batch is indexed. It continues in saved batches while content indexing is still in progress. Documents that are not indexed yet stay unavailable. When content indexing finishes, Nous runs a final pass for files that were not covered by an earlier permission batch.

Indexing status and testing

Indexed connections no longer have a shared indexing start date. After the upgrade, connections that used that setting discover older content in the background, within their existing source selection and permissions. You do not need to request a full reindex. This discovery does not force unchanged content to be processed again; subsequent refreshes remain incremental.

Paused connections wait until you resume them. Reconnect invalid credentials before discovery can continue. If a source cannot be fully read, its sync history keeps the failure visible. Correct the source problem and resume the connection to retry. Provider-specific retention and selection rules still apply, and live-app connections are not added to the index.

Open the connector detail page and check:

- the source-sync state;
- the search-indexing state and document counts;
- the last successful refresh;
- the **Last completed permission sync** when **Sync permissions** is enabled;
- the **Current access in Nous** sample for a quick check of stored document access.

Queued means the refresh is scheduled and waiting to begin. It is not a stalled scan, so do not reindex or remove the connector while it is queued. Nous shows a stalled warning only after an older refresh can be confirmed as no longer waiting or running. If that check is temporarily unavailable, the state stays unknown instead of showing a false warning.

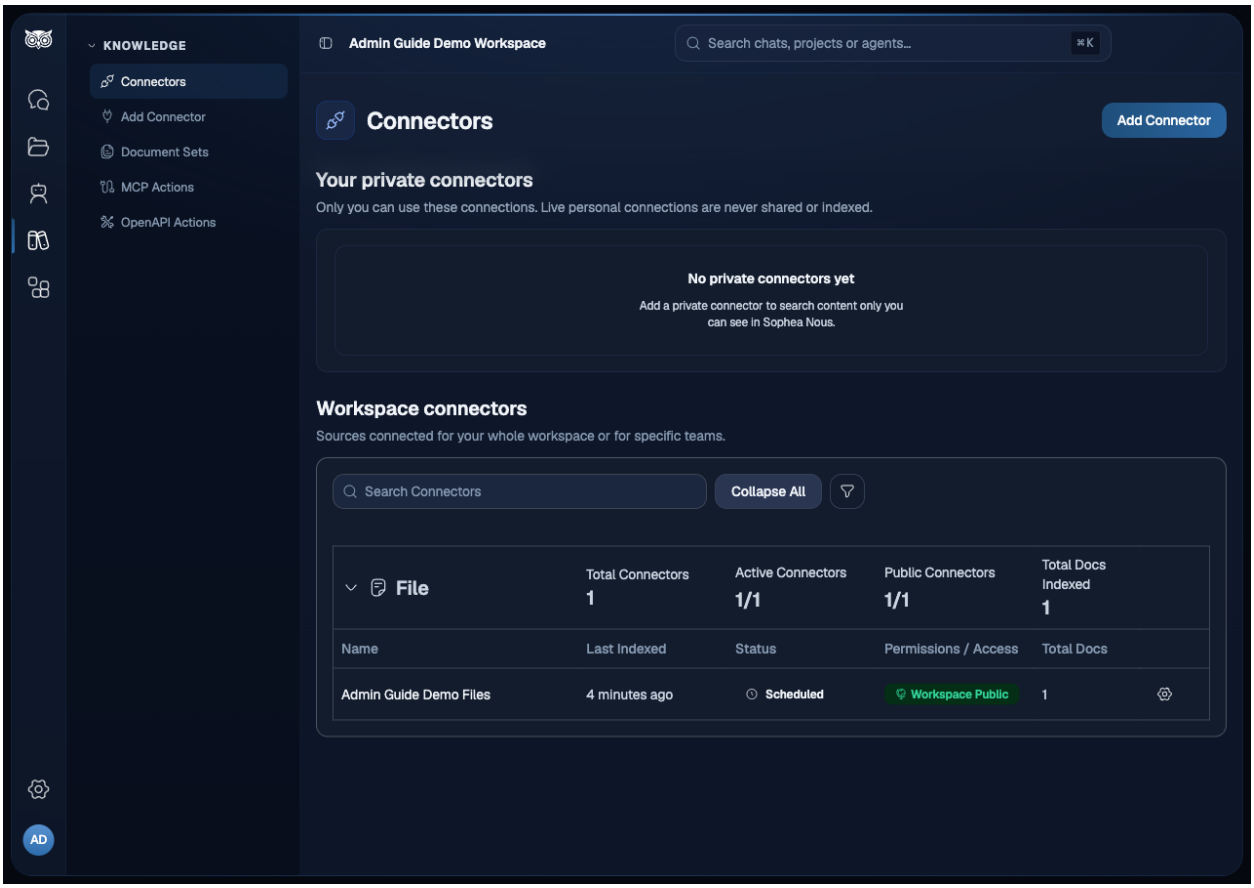

Figure 7.6: Connector list with a queued refresh

After the states are ready, search for a known item in **Knowledge Search**. Test once as a member who should have access and once as a member who should not have access. A private connector must fail closed for the second member.

Connector comparison

The current customer setup flow supports these 18 sources. The old internal Craft catalog and other legacy source types are outside this guide.

Guide	Indexed or live	Customer setup
SharePoint	Sites, libraries, folders, files, and supported pages are indexed.	Sophea-managed OAuth. Customer certificate credentials are an alternate path only for permission sync.
Google Drive	Selected Drive files, folders, shared drives, and shared files are indexed.	Sophea-managed OAuth.

Guide	Indexed or live	Customer setup
Microsoft Teams	The member's chats and visible team channel messages are searched live for the signed-in member. Nothing is indexed.	Sophea-managed OAuth.
Slack	The member's visible channels and messages are searched live for the signed-in member. Nothing is indexed.	Sophea-managed OAuth.
Confluence	Pages, attachments, comments, and selected spaces are indexed.	Sophea-managed OAuth.
Notion	Authorized pages and child pages are indexed.	Sophea-managed OAuth.
Dropbox	Selected Dropbox folders and files are indexed.	Sophea-managed OAuth.
Gmail	Personal mailbox messages are searched live for the signed-in member. They are not copied into the workspace index.	Sophea-managed OAuth.
Microsoft Outlook	Personal mailbox messages are searched live for the signed-in member. They are not copied into the workspace index.	Sophea-managed OAuth.
Microsoft Outlook Calendar	Personal calendar events are searched live; nothing is indexed.	Sophea-managed OAuth.
ClickUp	Tasks and Docs in one personal workspace are read live. Clear current-message requests can directly update supported task and Doc page fields with audit and replay protection; nothing is indexed.	Customer personal API token.
Jira	Issues, comments, and supported project data are indexed.	Customer Jira account and API token.
Linear	Issues, projects, teams, and supported comments are indexed.	Sophea-managed OAuth.
HubSpot	Personal deals, companies, contacts, and tickets are searched live; nothing is indexed.	Personal Sophea-managed OAuth.
Salesforce	Selected Salesforce objects and records are indexed.	Customer Salesforce credentials.
Amazon S3	Objects in the selected bucket and prefix are indexed.	Customer AWS access keys.
Google Cloud Storage	Objects in the selected bucket and path prefix are indexed.	Customer HMAC access key and secret.

Guide	Indexed or live	Customer setup
Web	Pages from a public website are indexed.	No credentials.

Sophea-managed OAuth means that Sophea operates the provider app. The customer authorizes access in the provider consent screen. The customer does not create or paste an OAuth client secret into Nous. Customer credentials mean that the customer creates and rotates the provider credential and enters it in the setup form.

Microsoft Teams, Slack, Gmail, Microsoft Outlook, Microsoft Outlook Calendar, and ClickUp are personal live-app connections. They use the signed-in member's account at question time. They do not add content to the workspace index, Knowledge Search, or document sets.

Choose a connector guide

Use the guide that matches the source:

- SharePoint
- Google Drive
- Microsoft Teams
- Slack
- Confluence
- Notion
- Dropbox
- Gmail
- Microsoft Outlook
- Microsoft Outlook Calendar
- ClickUp
- Jira
- Linear
- HubSpot
- Salesforce
- Amazon S3
- Google Cloud Storage
- Web

File connectors

A file connector holds documents you upload instead of reading a provider, so it is not in the comparison table above. Both admins and members can create one.

Text in PNG, JPEG, WebP, BMP and TIFF images is extracted for search. Visual indexing is optional and keeps the images available for visual search. Multi-page TIFF files include every page. If an image is damaged, its document shows an error instead of being marked indexed. Each image or TIFF page can contain up to 16,777,216 pixels. If indexing stops during a temporary service outage, it resumes from saved work.

PDFs that require a password cannot be indexed. Upload a copy that opens without a password. Retrying the same protected file will not resolve the error.

Role	Where	Access
Admin	**Connectors**, then **File**	**Only you**, **Specific teams**, or **Workspace**
Member	**Connectors**, then **File**	Private to the member, or shared with Teams the member belongs to

For an admin, **Only you** keeps the files personal, **Specific teams** gives access only to the Teams you select, and **Workspace** lets everyone in the workspace search the files. **Only you** uses the personal limits below, even for an admin. Team and Workspace connectors do not use the personal quota. All uploads still use the workspace per-file limit and the same PDF and ZIP checks. A ZIP may contain up to 3,000 entries and expand to at most 1 GB. If an archive exceeds a safety limit, extract it on your computer and select the files directly.

A member cannot share a connector with a Team they do not belong to, and cannot make one available to the whole workspace. Ask an admin to create the connector if the whole workspace needs it.

Personal File connectors work within these limits:

Limit	Value
File connectors per person	10
Files per connector, including extracted ZIP entries	3,000
Total upload size per person	1 GB

Per-file size uses the workspace upload limit in **Settings**, the same limit as every other upload.

An automation does not check these limits when it adds a file to a connector. The files it added still count: the next time the member adds or removes a file themselves, the 3,000 file limit is checked against everything the connector holds. The total upload size counts only what members upload in Nous.

Each upload sends up to 3,000 selected files in one request. ZIP contents count against the stored-file limit after extraction. The selection shows its file count and total size, with filenames in a scrollable list so the form actions stay reachable. Remove files from an over-limit selection before uploading. For an oversized or unreadable file, choose a smaller or repaired copy; for a protected PDF, choose a copy that opens without a password.

If an upload is interrupted, use Retry without changing the selection. The files are sent again. If the upload had already completed, Retry returns the same result without adding duplicates. Keep the page open to retain that Retry operation. Changing the selection starts a new operation. Upload completion means the files were accepted; indexing can continue before they appear in search.

Add or remove files later on the connector page. Removing a file deletes the stored copy and starts removing its content from search automatically. Search results can take a short time to update. You can add replacement files without recreating the connector. If the update is interrupted, it resumes automatically.

To use uploaded files in a scoped set, add the connector to a Document Set.

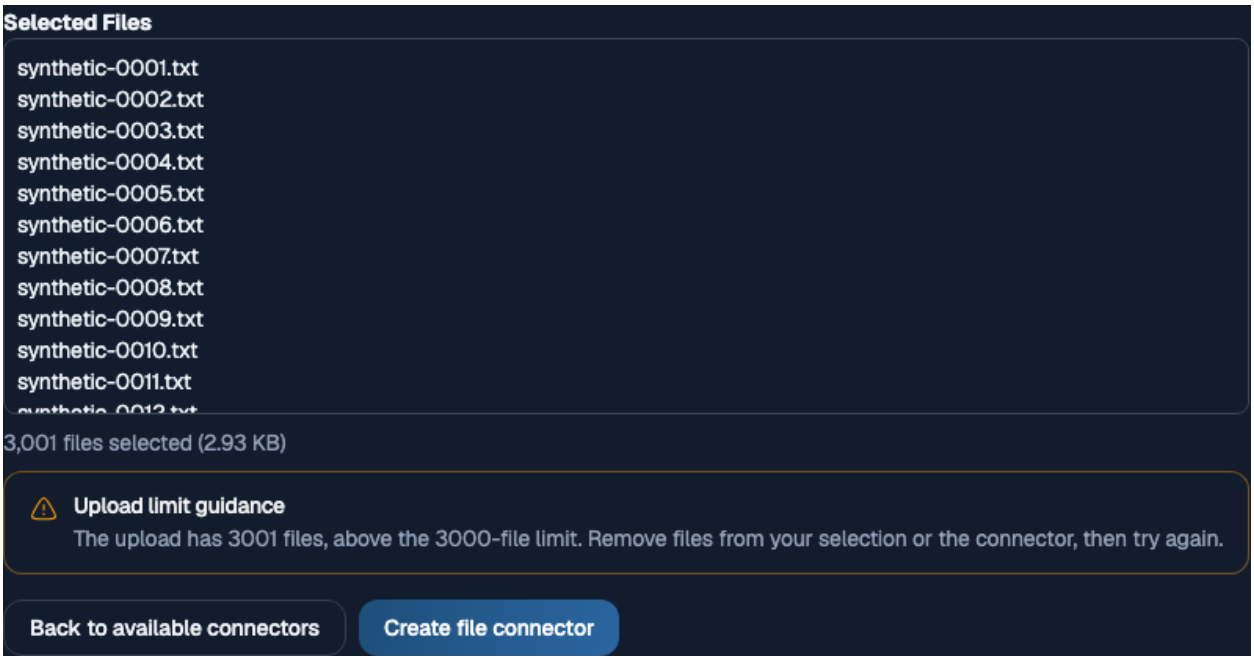

Figure 7.7: Selected filenames scroll within a bounded list. The count, limit guidance and actions stay outside it.

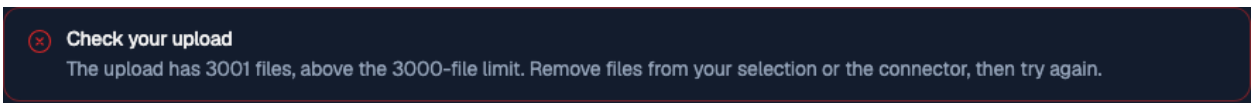

Figure 7.8: An over-limit selection explains how to reduce the file count before trying again.

Sync history and documents needing attention

The Portal connector page keeps the history of every sync, not only the latest one, and lists the documents that could not be added to search.

History shows one row per run: when it started, what kind of run it was, how long it took, how many documents it checked, and the outcome. Read the outcomes this way:

- **Succeeded**: every document the run checked is in search.
- **Completed with errors**: the run finished and some documents failed. The row counts the errors that run recorded; the documents that still need attention are listed on the **Overview** tab, since a later retry can already have fixed some of them.
- **Failed**: the run itself could not finish. The row carries the reason, for example an expired credential or a source that could not be reached.
- **Canceled**: something replaced the run, usually a full re-index.
- **Running**: the run is still going. The **Overview** tab shows how many documents it has found so far and how far through the source it is.

Documents needing attention on the **Overview** tab lists each document that failed, with what would fix it: a full re-index, a correction in the source system, or a closer look. **Retry failed documents** fetches those documents again and reports progress while it runs. It reaches every failed document on the connector, not only the ones shown on the page.

Search readiness counts the documents that were fetched but never became searchable. **Retry documents that are not searchable** processes them again. Unsupported file types are counted separately, and a retry does not change them.

Starting a full re-index cancels a document retry that is running; see Common recovery actions.

Common recovery actions

Full reindex processes missing, failed, and changed content first, then rebuilds healthy unchanged content. This applies to both files and text. Starting full reindex again while it is running keeps the same run and its saved progress. Starting it during a retry of failed or selected documents cancels that retry and preserves work already completed.

After an interruption, processing resumes from saved progress. Some sources need to scan their content again before continuing.

- **Reconnect** when OAuth authorization expires. Reconnect the same provider account and keep the existing scope where possible.
- **Rotate credentials** in the provider first. Test the new credential, then remove the old credential according to the provider's security policy.
- **Reindex** after a provider scope or content selection changes. Reindexing does not grant access that the connector ACL does not allow.
- **Remove** a connector only after checking document sets, agents, and Teams that use it. Removing it also removes its indexed content from those scopes.

Personal connection pause and resume

A personal (member-private) connection, such as a personal Sophea Meet connection, pauses automatically after five sync failures in a row. In **Connectors**, the **Your private connectors** table shows the connection as **Paused** together with the last sync error. Only the connection's owner sees the

Resume action. **Resume** releases the pause and requests an immediate sync; it does not wait for the next scheduled refresh. If the source still fails, the connection pauses again after another five failures; fix the cause before resuming again. Raw provider error details are never shown to members; the error line always carries a reviewed safe message.

If a provider rejects the connection, open its dedicated guide for the exact permission and recovery action.

Personal connector search index rebuild

The owner of a personal File connector can rebuild its search index from the connector's own page. Open **Connectors**, select the connector in **Your private connectors**, and choose **Rebuild search index**. Confirm the action. Your files stay in place; search results from that connector may be incomplete until the rebuild finishes, and the connector shows an indexing status while it runs. Starting a rebuild while one is already running keeps the current run and its progress. Workspace admins do not see this action for connectors they do not own.

Chapter 8

SharePoint Connector

The SharePoint connector can index a whole Microsoft 365 tenant or a narrow set of sites, libraries, folders, and files. This chapter is for workspace admins and Microsoft 365 admins who must choose the correct sign-in method, configure Microsoft Entra permissions, and verify document access.

8.1 Choose the setup path

Choose the path before you create the connector. The choice controls who owns the Microsoft app and whether Nous can mirror SharePoint permissions.

Setup path	Microsoft app owner	Document access	Best for
Managed by Sophea	Sophea	Public or Private	The fastest setup when the signed-in user's SharePoint access is enough
Self-managed Client Secret	Your organization	Public or Private	Organization-controlled app-only indexing without permission sync
Self-managed Certificate Authentication	Your organization	Public, Private, or Sync permissions	Organization-wide indexing that must mirror SharePoint users and groups

Important

Sync permissions requires Certificate Authentication. A Client Secret credential cannot use this access mode. Microsoft Entra receives the public certificate. Nous receives the matching password-protected PFX file.

8.2 Before you start

Prepare these roles and values:

- a Sophea Nous workspace admin who can create credentials and connectors
- a Microsoft Entra admin who can create app registrations, add application permissions, and grant tenant-wide admin consent
- the Microsoft 365 tenant ID, also called the Directory ID
- an account that can confirm which SharePoint sites should be indexed
- for certificate authentication, a secure location for the PFX file and its password

Use a dedicated Entra app registration for the connector. Do not reuse a user account password or a certificate used by an unrelated application.

8.3 Managed by Sophea

Use this path when you do not need SharePoint permission sync.

1. In Nous, open **Connectors**, select **Add Connector**, and choose **SharePoint** under **Managed by Sophea**.
2. Select **Connect with SharePoint**.
3. Sign in to Microsoft and approve the requested delegated permissions.
4. Choose the sites, libraries, folders, or files to index.
5. Set **Who has access** to **Specific Teams** unless the content is intended for every member of the workspace.
6. Create the connector and complete the verification.

This path uses the signed-in person's SharePoint access. It does not ask you for a client ID, client secret, public certificate, or PFX file. It does not offer **Sync permissions**.

8.4 Portal customer form fields

Portal first asks how this workspace should connect, then shows the form.

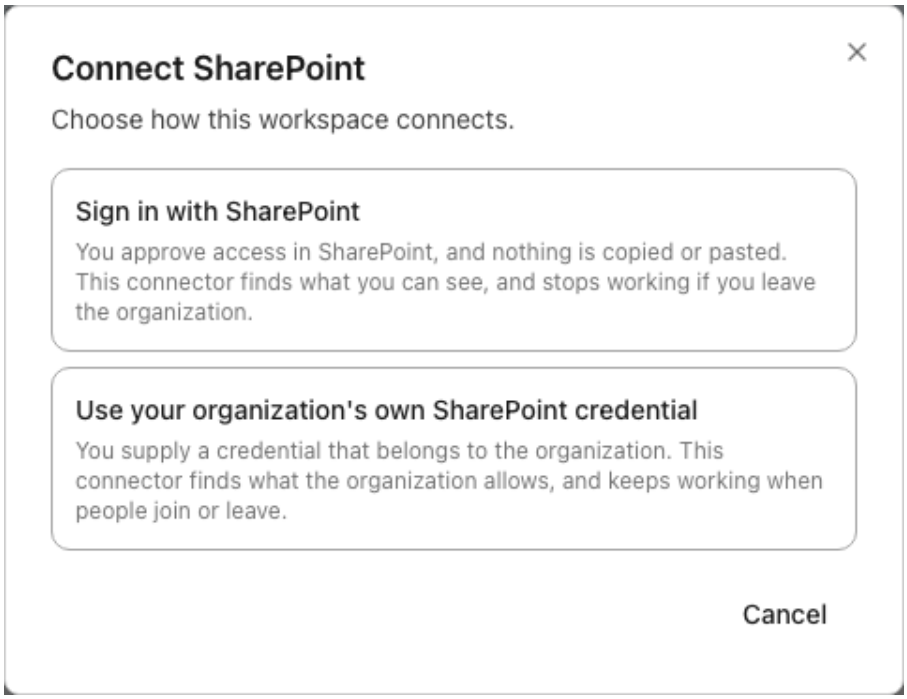

Figure 8.1: Choose a SharePoint connection method in Portal.

Choice	Who the connector acts as	What Portal asks for
Sign in with SharePoint	The person who clicks Connect	Nothing. You approve access in Microsoft
Use your organization's own SharePoint credential	The organization	Client ID, client secret, and Directory ID

Choose the organization credential for a shared connector. A personal sign-in indexes only what that person can see, and it stops working when they leave.

The form then shows **Name**, the SharePoint scope fields, and **Who has access**. Choose **Everyone in the workspace** or **Specific Teams**.

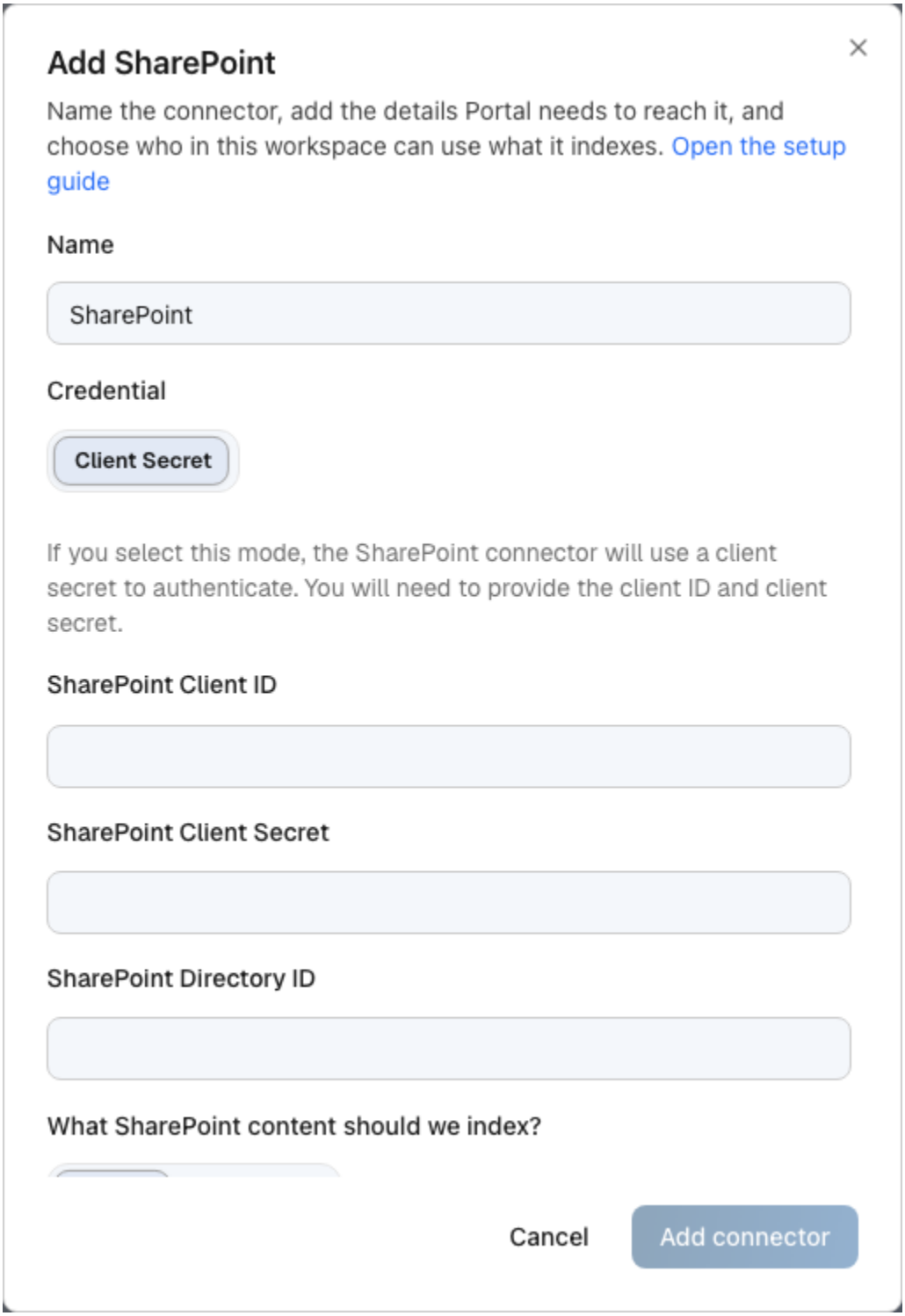

Figure 8.2: SharePoint client-secret form before any credential is entered.

Field	What to enter
Name	A unique connector name.
SharePoint content	In **Browse**, choose sites, libraries, folders, or files. Leave empty for all visible sites.
Sites	In **Paste URLs**, enter SharePoint site or folder URLs, one per line.
Index Documents	Keep enabled to index documents in the selected scope.
Index ASPX Site Pages	Enable only when SharePoint site pages must be indexed.
Treat sharing links as public?	Enable only when anonymous or organization-wide sharing links should be visible to every Nous user.
Excluded Sites	Optional site URLs or glob patterns to skip.
Excluded Paths	Optional file or folder glob patterns to skip.
Authority Host	Default `https://login.microsoftonline.com`.
Graph API Host	Default `https://graph.microsoft.com`.
SharePoint Domain Suffix	Default `sharepoint.com`.
Who has access	**Everyone in the workspace** or **Specific Teams**.

Client Secret is available in Portal, as the organization credential above, and in Nous. Certificate Authentication stays a Nous-only path, because it exists to enable **Sync permissions** and Portal does not offer that access mode. Use the certificate path in Nous when the customer must mirror SharePoint users and groups.

8.5 Create a self-managed Entra app

Use this setup for Client Secret or Certificate Authentication.

1. Open the Microsoft Entra admin center.
2. Go to **Identity > Applications > App registrations**.
3. Select **New registration**.
4. Enter a clear name such as `Sophea Nous SharePoint Production`.
5. Select **Accounts in this organizational directory only**.
6. Register the app. A redirect URI is not required for this app-only flow.
7. Record the **Application (client) ID** and **Directory (tenant) ID**.
8. Open **API permissions > Add a permission** and add the permissions for the access mode you will use.

Microsoft resource	Application permission	Public or Private	Sync permissions
Microsoft Graph	`Sites.Read.All`	Required	Required
Microsoft Graph	`GroupMember.Read.All`	Not required	Required
Office 365 SharePoint Online	`Sites.FullControl.All`	Not required	Required

The permission type must be **Application**, not **Delegated**. Select **Grant admin consent for *your***

tenant after all required permissions are listed. A green **Granted for *your tenant*** status must appear for each one.

> Warning
>
> Microsoft Graph and Office 365 SharePoint Online are separate resources in the permission picker. Adding `Sites.FullControl.All` under Microsoft Graph does not replace the Office 365 SharePoint Online permission required by permission sync.

8.6 Use a client secret

Use a client secret only for Public or Private indexing.

1. In the Entra app, open **Certificates & secrets > Client secrets**.
2. Select **New client secret**, choose an expiry that follows your company policy, and create it.
3. Copy the secret **Value** immediately. Do not copy the Secret ID.
4. In Nous, choose **Use your own Microsoft app** and create a credential.
5. Keep **Client Secret** selected. Enter the Client ID, Client Secret, and Directory ID.
6. Save the credential. Select Public or Private document access when you create the connector.

If the connector must use **Sync permissions**, create a separate Certificate Authentication credential. Adding more permissions does not make a Client Secret credential eligible for Sync permissions.

8.7 How certificate authentication works

One certificate creates two files with different jobs:

File	Contains	Upload destination	Safe to share?
`.cer`, `.pem`, or `.crt`	Public certificate only	Microsoft Entra app registration	Yes, it has no private key
`.pfx`	Certificate and private key	Sophea Nous credential form	No, protect it and its password

Nous signs an application request with the private key inside the PFX file. Microsoft Entra verifies that signature with the public certificate registered on the app, then returns an application token.

> Important
>
> Do not upload the PFX file to Microsoft Entra. Do not give the PFX file or its password to users. Nous currently requires the PFX upload and does not read the private key directly from Azure Key Vault.

8.8 Generate a staging certificate

Use a self-signed certificate for staging or a short test.

8.8.1 Windows PowerShell

Run these commands as the account that will export the files:

```
$cert = New-SelfSignedCertificate `
  -Subject "CN=Sophea Nous SharePoint Staging" `
  -CertStoreLocation "Cert:\CurrentUser\My" `
  -KeyAlgorithm RSA `
  -KeyLength 2048 `
  -HashAlgorithm SHA256 `
  -KeyExportPolicy Exportable `
  -NotAfter (Get-Date).AddYears(1)

$certificatePassword = Read-Host "Enter a password for the PFX file" -AsSecureString

Export-Certificate `
  -Cert $cert `
  -FilePath ".\sophea-sharepoint-staging.cer"

Export-PfxCertificate `
  -Cert $cert `
  -FilePath ".\sophea-sharepoint-staging.pfx" `
  -Password $certificatePassword

$cert | Select-Object Subject, Thumbprint, NotAfter
(Get-Item ".\sophea-sharepoint-staging.pfx").Length / 1KB
```

The final command shows the PFX size in KB.

8.8.2 macOS or Linux with OpenSSL

Run these commands in a terminal. OpenSSL asks you to protect the private `.key` file, then asks for the export password for the PFX file:

```
umask 077

openssl req -x509 -newkey rsa:2048 -sha256 -days 365 \
  -keyout sophea-sharepoint-staging.key \
  -out sophea-sharepoint-staging.crt \
  -subj "/CN=Sophea Nous SharePoint Staging"

openssl pkcs12 -export \
  -out sophea-sharepoint-staging.pfx \
  -inkey sophea-sharepoint-staging.key \
  -in sophea-sharepoint-staging.crt \
  -name "Sophea Nous SharePoint Staging"

openssl x509 -in sophea-sharepoint-staging.crt \
  -noout -subject -fingerprint -sha256 -enddate

wc -c < sophea-sharepoint-staging.pfx
```

Use the PFX export password when Nous asks for the certificate password. The final command shows

the exact PFX size in bytes. It must be 10,240 bytes or less.

Upload only `sophea-sharepoint-staging.crt` to Microsoft Entra. Upload `sophea-sharepoint-staging.pfx` to Nous. Never upload the private `sophea-sharepoint-staging.key` file to either system. Store or remove the private key according to your organization's secrets policy after you confirm that the PFX works.

For both methods, keep the PFX password out of command history, tickets, chat, and source control. The Nous upload must be a `.pfx` file of 10 KB or less.

Microsoft documents this process in Create a self-signed public certificate to authenticate your application.

8.9 Prepare a production certificate

For production, request a certificate from your company certificate authority or approved public key infrastructure. Follow these requirements:

- RSA key with at least 2048 bits and SHA-256 signing
- an exportable private key, because Nous must receive a PFX file
- a password-protected PFX file of 10 KB or less
- a public `.cer`, `.pem`, or `.crt` file from the same certificate
- an expiry date recorded in your certificate inventory and monitoring process

Store the PFX file and password through your normal secrets process. Keep them in separate controlled locations when company policy requires it. A company certificate authority may include a certificate chain in the PFX. Confirm that the final file still meets the Nous size limit before the change window.

Microsoft recommends a certificate issued by a trusted certificate authority for production. See Certificate credentials.

8.10 Upload the public certificate to Entra

1. Open the Entra app registration.
2. Go to **Certificates & secrets > Certificates**.
3. Select **Upload certificate**.
4. Upload the public `.cer`, `.pem`, or `.crt` file. Do not upload the PFX.
5. Record the displayed thumbprint and expiry date.
6. Confirm the thumbprint and expiry match the certificate you generated or received from your certificate team.

An Entra app can hold more than one active certificate. This allows safe rotation with an overlap period.

8.11 Upload the matching PFX to Nous

1. In Nous, open **Connectors**, select **Add Connector**, and choose **SharePoint**.
2. Select **Use your own Microsoft app**.
3. Create a credential and choose **Certificate Authentication**.
4. Enter the **SharePoint Client ID** and **SharePoint Directory ID** from the Entra app.

5. Enter the password used when the PFX was exported.
6. Upload the matching file under **SharePoint PFX File**. It must end in `.pfx` and be 10 KB or less.
7. Save and select the credential.

The public certificate in Entra and the PFX in Nous must come from the same certificate. The PFX password must match the password used during export.

8.12 Select SharePoint content

The connector provides two ways to select content:

- **Browse**: search and expand sites, document libraries, and folders. Select any scope down to an individual file. Unsupported files remain visible but cannot be selected.
- **Paste URLs**: enter site or folder URLs when browsing is not convenient. URL parsing currently supports English, Spanish, and German SharePoint site paths.

Leave the selection empty to index every site visible to the connected app or account. Use a narrow selection when the workspace needs only a defined part of the tenant.

Selecting a site, library, or folder includes supported documents below it. An individual file selection includes only that file. Renaming that file or moving it inside the same library keeps it selected. A successful refresh removes it if it is deleted, access is permanently removed, or it moves to another library. A temporary Microsoft service failure does not remove the previously indexed copy.

Older SharePoint files within the selected content are included regardless of the former shared indexing start date. Existing connections that used that setting discover the missing history automatically after the upgrade. Their sites, folders, exclusions, and document permissions do not change. Follow Indexing status and testing to check progress and verify a historical file in search.

8.13 Choose document access

Access mode	Search visibility	Supported credentials
Public	Every workspace member can retrieve indexed content	Managed OAuth, Client Secret, or Certificate
Private	Only supported Nous users or Portal Teams can retrieve indexed content	Managed OAuth, Client Secret, or Certificate
Sync permissions	Search mirrors SharePoint users, groups, inherited grants, and supported public grants	Self-managed Certificate Authentication only

Choose **Sync permissions** when you create the connector, before indexing starts. After indexing starts, delete and recreate the connector to change from Public or Private to Sync permissions.

Once a connector uses **Sync permissions**, its access mode cannot change to Public or Private. Recreate the connector to leave permission sync.

SharePoint group membership refreshes about every five minutes. Document access entries refresh about every thirty minutes. The **Uses source permissions** badge identifies the access mode. It does not prove that a run finished. Open the connector detail page and check **Last completed permission**

sync. **Not completed yet** means no successful permission sync has finished. Expand **Current access in Nous** to inspect a small sample of stored document access.

If SharePoint returns a recognized folder-specific permission error or an inconsistent folder location, the affected folder and its subfolders retain their last synchronized folder access for the current pass. Other folders can continue refreshing. Files whose permissions can still be read continue to refresh, including files beneath those folders. A completed pass does not mean every folder's permissions refreshed. Unreadable folders are checked again on the next pass. An unexpected permission-sync error stops the current pass; stored access remains until a later successful pass can refresh it.

New documents without synchronized permissions remain hidden. Unknown or unsupported permission entries grant no access.

> Warning
>
> **Treat sharing links as public?** is off by default. Enable it only when anonymous and organization-wide SharePoint sharing links should make a document searchable by every Nous workspace member.

8.14 Advanced settings

The SharePoint form provides these advanced settings:

- **Index Documents**: on by default
- **Index ASPX Site Pages**: off by default
- **Treat sharing links as public?**: off by default
- **Excluded Sites** and **Excluded Paths**: optional glob patterns
- **Authority Host**: `https://login.microsoftonline.com`
- **Graph API Host**: `https://graph.microsoft.com`
- **SharePoint Domain Suffix**: `sharepoint.com`

For Microsoft 365 GCC High or DoD, use the hosts approved for that tenant. The current form documents `https://login.microsoftonline.us`, `https://graph.microsoft.us`, and the GCC High suffix `sharepoint.us`. Confirm the correct cloud with your Microsoft 365 administrator before creating the connector.

8.15 Exclude files or sites

Exclusions stop SharePoint content from being indexed when it cannot be removed or moved at the source.

1. Open the SharePoint connector detail page.
2. In **Connector configuration**, select **Edit excluded paths and sites**.
3. Enter one pattern per line, such as `report.pdf`, `*.tmp`, `~$*`, or `Archive/*`, under **Excluded Paths**. Matching is case-insensitive.
4. Enter a safe site pattern such as `*://*/sites/archive-*` under **Excluded Sites**.
5. Select **Save** for future refreshes, or **Save and prune now** to remove matching indexed copies.

Do not enter customer filenames, tenant names, private URLs, or secrets in a screenshot or example.

8.16 Verify the connector

Complete all checks before you give the connector to users:

1. Open the site browser. Confirm the intended sites and libraries appear.
2. Confirm **Sync permissions** is enabled only for a Certificate Authentication credential.
3. Create the connector. A failed permission check must be corrected before the connector is accepted.
4. Wait for source sync and search indexing to complete.
5. For permission sync, wait until **Last completed permission sync** shows a real time.
6. Search for a known SharePoint document as a user who has source access.
7. Repeat the search as a user who does not have source access. The document must not appear.
8. If sharing links are treated as public, verify that decision with a normal workspace member who has no direct SharePoint grant.

Do not treat successful site discovery as a complete access test. Site listing, document indexing, and permission sync use different Microsoft permissions.

8.17 Troubleshooting

What you see	Likely cause	What to do
The certificate note appears before creating the connector	A self-managed Client Secret credential is selected	This is a setup note. Use Public or Private, or create a Certificate Authentication credential for Sync permissions.
Sync permissions is not shown	The connector uses Managed by Sophea OAuth	Use Public or Private, or recreate it with a self-managed certificate credential.
Sync permissions is disabled	A Client Secret credential is selected	Upload the public certificate to Entra and the matching PFX to a Nous certificate credential.
“The Client ID does not match the app registration this certificate belongs to. Use the Application (client) ID of the exact Entra app registration where the public certificate was uploaded, and check the Directory (tenant) ID.”	The certificate is registered on a different Entra app registration, or a wrong Client ID was entered	Use the Application (client) ID of the app registration that holds the public certificate (.cer), and check the Directory (tenant) ID.
“Microsoft could not verify the certificate signature. Upload the public certificate (.cer) that matches this PFX to the Entra app registration’s Certificates list, and replace it if it expired.”	The PFX private key has no matching public certificate on the app registration, or that certificate expired	Upload the public certificate (.cer) that matches the PFX to the Entra app registration’s Certificates list, and replace it if it expired.
“The client secret was rejected. Paste the secret VALUE from Entra, not the secret ID; create a new secret if it expired.”	The secret value is wrong or expired	Paste the secret VALUE from Entra, not the secret ID; create a new secret if it expired.

What you see	Likely cause	What to do
No sites appear	`Sites.Read.All` is missing or lacks admin consent	Add Microsoft Graph Application `Sites.Read.All`, grant tenant-wide admin consent, and retry.
Site role assignments cannot be read	`Sites.FullControl.All` is missing or was added under the wrong resource	Add Office 365 SharePoint Online Application `Sites.FullControl.All`, grant admin consent, and retry.
Group membership cannot be read	`GroupMember.Read.All` is missing	Add Microsoft Graph Application `GroupMember.Read.All`, grant admin consent, and retry.
Nous rejects the upload, or reports “Failed to load the SharePoint certificate. Verify the PFX and certificate password are correct.”	The file is not a PFX, is larger than 10 KB, is corrupt, does not contain an exportable private key, or the entered PFX password does not match	Export a valid password-protected `.pfx` of 10 KB or less from the same certificate registered in Entra, and re-enter its exact password.
Nous reports that no site is available for validation	The app can authenticate but no site can be discovered	Select at least one site and confirm the app has access to it.
Last completed permission sync shows **Not completed yet**	No successful permission run has completed	Check the connector error state and the three required application permissions.

8.18 Rotate the certificate

Start rotation 30 to 60 days before expiry:

1. Generate or receive the replacement certificate and matching PFX.
2. Upload the new public certificate to the existing Entra app. Keep the old certificate active.
3. Update the Nous certificate credential with the new PFX and password.
4. Test site discovery, connector creation or refresh, and permission sync.
5. Confirm **Last completed permission sync** advances and run the authorized and unauthorized user search checks.
6. Remove the old certificate from Entra only after the new credential works.
7. Remove the old PFX and password according to your secrets-retention policy.

Do not delete the old Entra certificate before Nous uses the replacement. The overlap prevents an avoidable indexing outage.

8.19 Microsoft references

- Create a self-signed public certificate to authenticate your application
- Add credentials to an application
- Certificate credentials
- Microsoft Graph permissions reference

For general connector lifecycle, indexing, retry, and deletion behavior, see Connectors.

Chapter 9

Google Drive Connector

Use this guide for a workspace Google Drive connector. Google Drive content is indexed and is available to Knowledge Search after indexing completes.

9.1 What is indexed or searched live

Nous can index My Drive files, shared drives, selected folders, and files shared with the connecting account. Google Drive permission sync is available only with a customer service-account credential and domain-wide delegation. A Sophea-managed OAuth connection uses delegated access from the account that authorizes it.

9.2 Administrator role and authentication

Sophea manages the OAuth app. The connecting Google account must be allowed to read the required Drive content. A Google Workspace super administrator must also approve domain-wide delegation when you use permission sync.

You do not create a Google OAuth client for the standard customer path. The consent request uses `drive.readonly` and `drive.metadata.readonly`.

9.3 Provider setup and permissions

For standard OAuth, confirm that the account can open every folder or shared drive you plan to index. For permission sync, create a service account in Google Cloud, enable the Drive and Admin SDK APIs, and authorize the service account client ID in **Admin console > Security > Access and data control > API controls > Manage Domain Wide Delegation**. Add these read-only scopes:

- `https://www.googleapis.com/auth/drive.readonly`
- `https://www.googleapis.com/auth/drive.metadata.readonly`
- `https://www.googleapis.com/auth/admin.directory.user.readonly`
- `https://www.googleapis.com/auth/admin.directory.group.readonly`

9.4 Setup form fields

The Portal form has these common fields:

- **Name**: a unique name for this workspace connector.
- **Who has access**: choose **Everyone in the workspace** or **Specific Teams**. Permission sync is shown only for a supported service-account credential.

The Google Drive scope fields are:

Field	What to enter
How should we index your Google Drive?	Choose **General** or **Specific**.
Google Drive content	In **General**, choose all accessible Drive content or selected folders.
Include shared drives?	Include all shared drives visible to the connected account.
Include My Drive?	Include all files in the connected account's My Drive.
Include All Files Shared With You?	Include files shared directly with the connected account.
Shared Drive URLs	In **Specific**, enter comma-separated shared-drive URLs.
Folder URLs	In **Specific**, enter comma-separated folder URLs. Subfolders are included.
My Drive Emails	In **Specific**, enter comma-separated account emails whose My Drive should be indexed.
Specific User Emails	In advanced settings, limit indexing to files visible to these users.
Hide domain link-only files?	Hide files that need a link even when shared to the domain or public.

Leave a scope empty only when you intend to use the broader default. Set **Who has access** and any Portal Team assignments in the common connector form.

9.5 Use the organization's own credential

Portal asks how this workspace connects before it shows the connector form. **Sign in with Google Drive** connects as the person who clicks Connect. **Use your organization's own Google Drive credential** connects as a Google service account that the organization created.

Choose the organization's credential for a shared connector. It reaches what the organization allows rather than what one person can see, and it keeps working when that person leaves. A personal sign-in stops working when the account that authorized it goes away.

The credential form asks for two things:

Field	What to enter
Google service-account key	The JSON key file Google produced when the service account was created. Drag it onto the field or click to browse.
Delegated Workspace admin email	The Google Workspace admin the connector acts as. What that admin can reach in Drive is what Sophea can find.

Create the service account and authorize domain-wide delegation with the read-only scopes in Provider setup and permissions before you fill this in. Sophea checks the key with Google while you wait. A key Google rejects is reported in the form, and Sophea keeps nothing.

The key file is write-only. Sophea stores it securely and never shows it again, not in this form and not in the credentials list. To connect again later, your organization supplies the key file again.

When the workspace already holds credentials, the form lists them instead. Choose one, or choose **Add a new credential** to supply another.

This method has no folder browser. Signing in lets Portal list your Drive folders because it holds that person's Google session. A service account holds no session, so **Google Drive content** is a plain box you type links into, one per line. Leave it empty and Sophea indexes everything the credential can reach. Fill it in when the connector should cover less than that.

Credentials this workspace holds are listed under **Organization credentials** on the **Connectors** tab, with the service-account address, the admin each one acts as, and the date it was added. **Remove** deletes one. A credential that a connector is using cannot be removed: delete that connector first, then remove the credential.

9.6 Workspace and Team access

Use **Everyone in the workspace** only for content that every member may find. Use **Specific Teams** for a smaller audience. Team membership is managed in Portal. With permission sync, Google Drive users, groups, inherited folder access, and supported public grants are mirrored. A document without a known permission remains hidden.

9.7 Test the connection

1. Save the connector and wait for source sync and search indexing to finish.
2. Open the detail page and check the document count and last refresh.
3. Search for a known file name and a phrase from that file.
4. Test as an intended Team member and as a member outside the Team.
5. If permission sync is enabled, check **Last completed permission sync** and repeat the two-user test after that run completes.

9.8 Common errors

Error	Action
Drive scope is empty	Check the selected folders, shared drives, and account access.
Permission sync is not available	Recreate the connector with a service account and domain-wide delegation.
Admin Directory scope is missing	Add both Admin SDK directory read scopes, then reconnect or recreate the credential.
A file is missing	Confirm that the connecting account can open it and that it is not link-only when that option is enabled.

9.9 Reconnect, rotate, and remove

When Google permanently rejects a standard OAuth credential, Nous stops the first failed indexing attempt and shows the reconnect warning. Use **Reconnect**. After the new credential is saved, indexing resumes from the saved checkpoint instead of starting over.

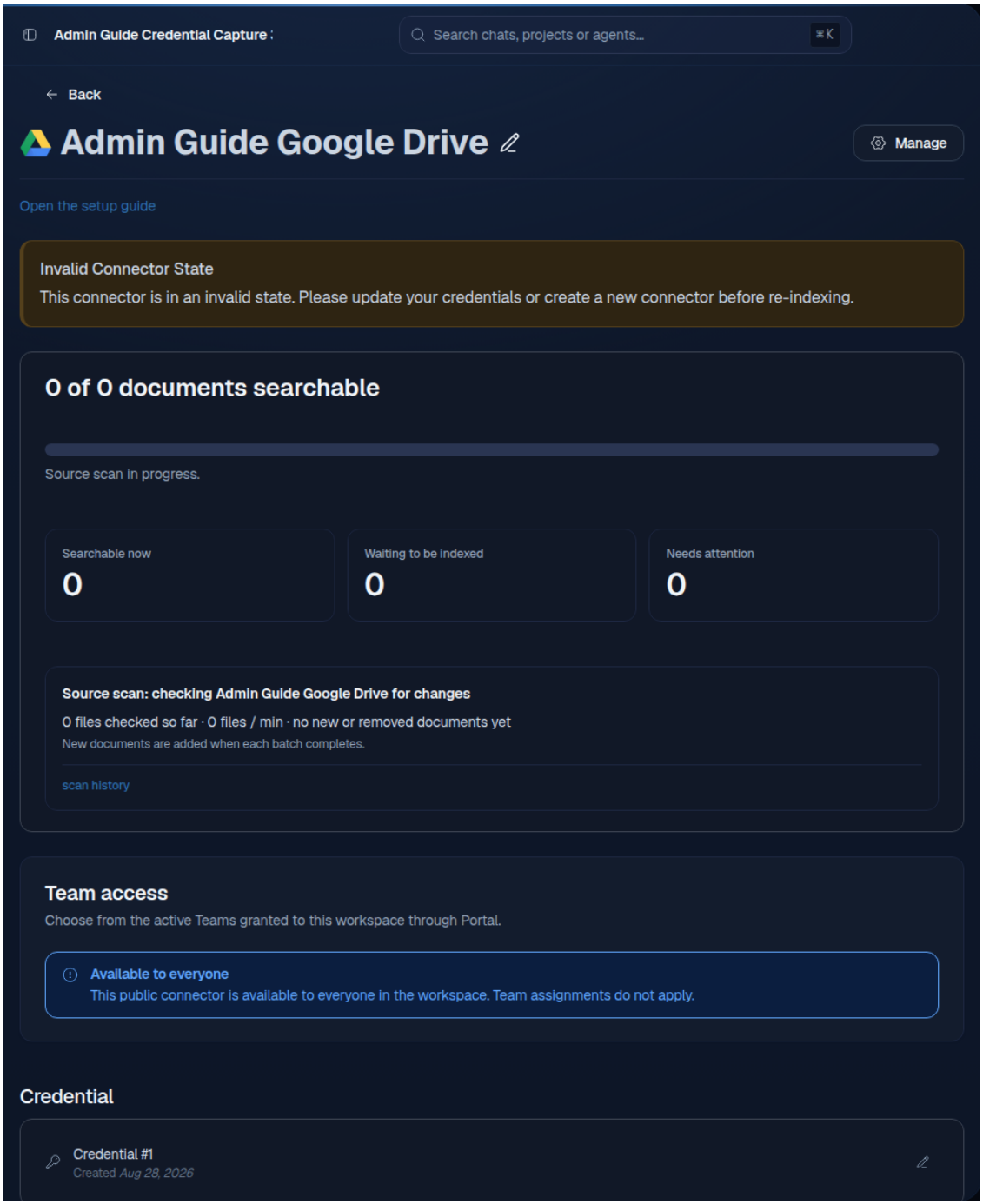

Figure 9.1: Google Drive connector warning after Google rejects its credential

Use **Reconnect** only for standard OAuth. For permission sync or a service-account key change, delete the connector, delete the old credential, add the new credential, and recreate the connector. Portal validates the new key before saving it. Choose permission sync when you create the connector, before indexing starts. After indexing starts, delete and recreate the connector to change from Public or Pri-

vate to permission sync. Permission sync cannot be changed to another access mode after activation. Before removal, check document sets, agents, and Team shares that use the connector.

9.10 Official provider documentation

- Choose Google Drive API scopes
- Google Workspace domain-wide delegation
- Google Drive API files

Chapter 10

Microsoft Teams Connector

Microsoft Teams in the current customer flow is a personal live connection. Each member connects their own Microsoft account, and the assistant answers from that member's own Teams view at request time. Nous does not index Teams content.

10.1 What is indexed or searched live

Teams messages are searched live. Nothing is copied into the workspace search index, Knowledge Search, or document sets. The assistant can search the member's chats and the messages of the team channels the member can already see in Teams. Without keywords, the recency listing returns the member's recent chats only; channel answers need a keyword query. Attachments and photos are not fetched. Each member who needs Teams answers connects their own account from the connector catalog.

10.2 Administrator role and authentication

Sophea manages the Microsoft OAuth app. The member authorizes the personal connection. A Microsoft 365 administrator may need to grant tenant consent for the application, because the Teams message permissions (`ChannelMessage.Read.All`, `Chat.Read`) require admin consent in most tenants. The member can only ever read what their own Microsoft identity can already open in Teams.

The standard delegated grant requests the following Microsoft Graph scopes, plus offline sign-in scopes. You do not create an app registration for the standard customer path.

Scope	Why Nous requests it
`User.Read`	Sign in and read the connecting profile.
`Team.ReadBasic.All`	Resolve the Teams the member belongs to for live search.
`Channel.ReadBasic.All`	Resolve channels inside each Team for live search.
`ChannelMessage.Read.All`	Search live the channel messages the member can read. Requires Microsoft tenant admin consent.

Scope	Why Nous requests it
`Chat.Read`	List and read the member's own chats for live search.

10.3 Provider setup and permissions

Ask a Microsoft 365 or Teams administrator to review the requested Graph permissions. Grant tenant consent when the admin-consent screen requires it. There is no admin-side source scope to pick: a personal connection always follows the connecting member's own Teams visibility. A connected member can narrow it from the setup page: **All chats and channels** (the default) or **Selected chats and channels** with individual picks. The choice only restricts what the assistant may read; reconnecting keeps it, disconnecting and connecting again resets it to All.

`ChannelMessage.Read.All` is a delegated scope but requires Microsoft tenant admin consent. A Microsoft 365 administrator can grant tenant-wide consent directly through the admin-consent endpoint:

```
https://login.microsoftonline.com/{tenant}/v2.0/adminconsent?client_id={client_id}&redirect_uri={connector oauth callback url}&state=adminconsent
```

Replace the placeholders with the Microsoft tenant id or domain, the Sophea OAuth app client id, and the connector OAuth callback URL shown in Portal.

10.4 Setup form fields

Besides the **Connect**, **Reconnect**, and **Disconnect** actions, the personal setup shows the chat and channel scope choice right after the account connects, and the connection's detail page shows the same choice later: **All chats and channels** (default) or **Selected chats and channels** with individual checkboxes. The member selects the account in the Microsoft consent flow. There is no workspace access or refresh setting for this personal connection.

10.5 Workspace and Team access

Workspace and Team access do not apply. The connection belongs to one member. Other members must connect their own Microsoft accounts. Workspace admins cannot make a personal Teams connection public or add it to a document set.

10.6 Test the connection

1. Ask a question whose answer exists only in the member's Teams chats, and a second question with keywords that match a channel conversation.
2. Confirm that the answer cites live Teams messages with openable links.
3. Confirm that the messages do not appear in Knowledge Search or a document set.
4. Disconnect the member account and confirm that live search no longer returns Teams content.

10.7 Common errors

Error	Action
Admin approval is required	Ask a Microsoft 365 administrator to grant consent for the listed Graph permissions.
Chat answers work but channel answers are empty	Channel results need a keyword query; the recency listing covers chats only.
Messages are missing	Confirm that the member can open the chat or channel in Teams, then reconnect.
Consent changed or expired	Use **Reconnect** to refresh the grant.

10.8 Reconnect, rotate, and remove

Use **Reconnect** after an admin-consent change or an expired grant. To change accounts, disconnect the old account first. Sophea manages the OAuth app rotation. **Disconnect** revokes the supported grant and removes the local personal connection.

10.9 Official provider documentation

- Microsoft Graph permissions reference
- Microsoft Graph Teams overview
- Grant tenant-wide admin consent

Chapter 11

Slack Connector

Slack in the current customer flow is a personal live connection. Each member connects their own Slack account, and the assistant answers from that member's own Slack view at request time. Nous does not index Slack content.

11.1 What is indexed or searched live

Slack messages are searched live. Nothing is copied into the workspace search index, Knowledge Search, or document sets. The assistant can search the public channels, private channels, and direct messages that the member can already see in Slack. Each member who needs Slack answers connects their own account from the connector catalog in Settings. A member without a connection gets no Slack results.

11.2 Administrator role and authentication

Sophea manages the Slack OAuth app. The member authorizes the personal connection with their own Slack user token. A Slack workspace owner or administrator may need to approve the app when the workspace requires admin approval for installed apps. The member can only ever read what their own Slack identity can already open in Slack.

The personal connection requests the following Slack user scopes. You do not create a Slack app for the standard customer path.

Scope	Why Nous requests it
`channels:read`, `groups:read`, `im:read`, `mpim:read`	Resolve the channels and conversations the member belongs to for live search.
`search:read`	Search messages live with the member's own token.
`channels:history`, `groups:history`, `im:history`, `mpim:history`	Read the message history the member can already open in Slack.
`users:read`, `users.profile:read`	Resolve member names and profiles for readable results.

11.3 Provider setup and permissions

Ask a Slack workspace owner or administrator to approve the Sophea app when the workspace requires admin approval for new apps. There is no source scope to pick: a personal connection always follows the connecting member's own Slack visibility. Channels and direct messages the member cannot see in Slack never appear in answers.

11.4 Setup form fields

The current personal setup has no editable form fields. It shows **Connect**, **Reconnect**, and **Disconnect** actions. The member selects the account in the Slack consent flow. There is no workspace access or refresh setting for this personal connection.

11.5 Workspace and Team access

Workspace and Team access do not apply. The connection belongs to one member. Other members must connect their own Slack accounts. Workspace admins cannot make a personal Slack connection public or add it to a document set.

11.6 Test the connection

1. Ask a question whose answer exists in a Slack channel the connected member can read.
2. Confirm that the answer cites live Slack messages with openable links.
3. Confirm that the messages do not appear in Knowledge Search or a document set.
4. Disconnect the member account and confirm that live search no longer returns Slack content.

11.7 Common errors

Error	Action
Slack asks for admin approval	Ask a workspace owner or administrator to approve the app.
Channel answers are empty	Confirm that the member can open that channel in Slack, then reconnect.
Messages are missing	Slack live search only covers what the member's own account can read; confirm the member's access in Slack.
Consent changed or expired	Use **Reconnect** to refresh the grant.

11.8 Reconnect, rotate, and remove

Use **Reconnect** after a consent change or an expired grant. To change accounts, disconnect the old account first. Sophea manages the OAuth app. **Disconnect** revokes the supported grant and removes the local personal connection.

11.9 Official provider documentation

- Slack OAuth scopes
- Slack OAuth v2
- Slack search.messages reference

Chapter 12

Confluence Connector

Use this guide for a Confluence Cloud or Server/Data Center connector. Nous indexes pages, supported attachments, comments, and the scope selected in the form.

12.1 What is indexed or searched live

For Confluence Cloud, the Sophea-managed OAuth grant reads the pages and spaces available to the authorizing Atlassian account. For Server or Data Center, confirm that the configured endpoint and credential can read the target wiki.

12.2 Administrator role and authentication

Sophea manages the Atlassian OAuth app for the standard customer path. The authorizing Atlassian account needs access to the target spaces and pages. An Atlassian organization or site administrator may need to approve the OAuth app for the customer organization.

The OAuth grant requests `read:me`, Confluence content and summary read scopes, space summary, search, user, and `read:space:confluence` access. You do not create an OAuth app for the standard path.

12.3 Provider setup and permissions

Confirm the Confluence base URL and the account's space access. Select Cloud for an Atlassian Cloud URL. Clear it for Server or Data Center. Use the least scope that contains the pages the workspace needs.

12.4 Setup form fields

Field	What to enter
Name	A unique connector name. The default is the Cloud site name when available.
Is Cloud	On for Confluence Cloud. Off for Server or Data Center.
Wiki Base URL	The base URL, for example `https://example.atlassian.net/wiki`.
Using scoped token	Turn on only when the supplied Atlassian token uses scoped permissions.
How Should We Index Your Confluence?	Choose Everything, Space, Page, or CQL Query.
Space Key	The space key when **Space** is selected, for example `KB`.
Page ID	The page ID when **Page** is selected.
Index Recursively	Include the selected page and its children.
CQL Query	A query that returns page objects. Do not add time filters.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

12.5 Workspace and Team access

The connector cannot read a page that the Atlassian account cannot open. Use Use **Specific Teams** for a smaller workspace audience. Portal manages Team membership. A Team share does not change Confluence permissions.

12.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Open a known page in Confluence as the authorizing account.
3. Search for its title and a distinctive phrase.
4. Check that attachments and comments expected by the scope appear.
5. Test Workspace or Team visibility with the intended users.

12.7 Common errors

Error	Action
OAuth approval is required	Ask the Atlassian organization or site administrator to approve the app.
No pages are found	Check the base URL, account access, and selected space or page.
CQL returns no content	Return page objects, remove time filters, and check the CQL syntax.
Server connection fails	Clear **Is Cloud**, use the correct base URL, and confirm the server credential.

12.8 Reconnect, rotate, and remove

Use **Reconnect** when the Atlassian grant or account access changes. Sophea rotates the managed OAuth app. If an Atlassian administrator changes app permissions, reconnect and test again. Before removal, check document sets, agents, and Team shares.

12.9 Official provider documentation

- Atlassian OAuth 2.0 (3LO)
- Confluence Cloud REST API
- Confluence Query Language

Chapter 13

Notion Connector

Use this guide for a Notion workspace connector. Nous indexes the pages that the authorized Notion connection can access.

13.1 What is indexed or searched live

Notion pages and child pages shared with the connection are indexed. Databases and page content are included only when the connection can access them. A page selected during the Notion authorization flow limits what Nous can discover.

13.2 Administrator role and authentication

Sophea manages the Notion public OAuth connection. A Notion workspace owner or administrator should approve the connection and share the required top-level pages with it. You do not paste a Notion integration token for the standard customer path.

The current OAuth flow does not request a separate scope string. Notion shows the pages granted during authorization.

13.3 Provider setup and permissions

Before you connect, identify the top-level pages and databases the workspace needs. In the Notion authorization picker, select those pages. Keep the grant limited to the required content. If a child page is needed, share it through an authorized parent or select it explicitly.

13.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Root Page ID	Optional. Enter a page ID to index that page and its children only.

Field	What to enter
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

Leave **Root Page ID** empty to index all pages the connection can access. Use the ID when the connection has broad access but the workspace needs one area.

13.5 Workspace and Team access

Notion sharing controls what Nous can read. **Specific Teams** controls who can search the indexed result. Manage Portal Team membership in Portal. A Team share cannot grant a user access to a page that Notion did not grant to the connection.

13.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Confirm that the selected root page or expected pages appear in the detail page.
3. Search for a known page title and phrase.
4. Test the result with a member in the intended Workspace or Team audience.

13.7 Common errors

Error	Action
Page is missing	Share the page with the connection, then reconnect or refresh.
Root Page ID is rejected	Use the page ID, not the full browser URL, and confirm that the connection can open it.
Authorization does not return a refresh token	Ask the administrator to use the current Sophea connection and keep token expiration enabled for the managed app.
Search is empty	Check the Notion page share and the connector indexing state.

13.8 Reconnect, rotate, and remove

Use **Reconnect** after changing the selected pages or when the grant expires. Sophea rotates the managed OAuth app. Before removal, check document sets, agents, and Team shares that use the connector.

13.9 Official provider documentation

- Notion authorization
- Notion authentication

- Notion API pages

Chapter 14

Dropbox Connector

Use this guide for a workspace Dropbox connector. Nous indexes files in the Dropbox folders selected during setup.

14.1 What is indexed or searched live

The connector indexes Dropbox files and supported metadata from all accessible content or from selected folders. It does not index content that the Dropbox account cannot read.

14.2 Administrator role and authentication

Sophea manages the Dropbox OAuth app. A Dropbox team admin or account owner may need to approve the app for a team. The authorizing Dropbox account must have access to the folders to index.

The grant requests read access to file metadata and content, sharing read and write operations needed to discover the selected scope, and account identity. You do not create or paste a Dropbox app secret for the standard path.

14.3 Provider setup and permissions

Ask the Dropbox team admin to approve the app if the team policy requires it. Confirm that the authorizing account can open every selected folder. Keep the scope at the smallest folder set that meets the workspace need.

1. Select **Connect** and approve the Sophea-managed Dropbox app.
2. Set **Dropbox indexing scope** to **Selected folders**.
3. Select only the folders the workspace needs, such as `Shared/Policies`.
4. Set **Who has access** to the workspace or the required Teams.
5. Save the connector and wait for source sync and search indexing.

14.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Dropbox indexing scope	Choose all accessible Dropbox content or selected folders.
Folders	Select one or more Dropbox folders when selected-folder mode is used.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

14.5 Workspace and Team access

Dropbox permissions control what the connector can read. Nous access controls who can search that indexed content. Use **Specific Teams** for a limited audience. Team membership is managed in Portal. A Team share cannot grant access to a Dropbox file that the connection cannot read.

14.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Check that one known file from each selected folder appears.
3. Search for a distinctive phrase from a known file.
4. Test as an intended workspace or Team member.

14.7 Common errors

Error	Action
Dropbox admin approval is required	Ask the team admin or account owner to approve the managed app.
A folder is missing	Confirm that the authorizing account can open it and reconnect.
The folder picker is empty	Reconnect the Dropbox grant and check the selected account.
Files are not searchable	Wait for search indexing and confirm that the files are supported.

14.8 Reconnect, rotate, and remove

Use **Reconnect** when the Dropbox grant is revoked or the account changes. Sophea manages app credentials and token rotation. Before removal, check document sets, agents, and Team shares.

14.9 Official provider documentation

- Dropbox OAuth guide
- Dropbox API file access
- Dropbox team app permissions

Chapter 15

Gmail Connector

Gmail in the current customer flow is a personal live connection. It answers questions from the signed-in member's mailbox at request time.

15.1 What is indexed or searched live

Gmail messages are searched live. Mail is not copied into the workspace search index, Knowledge Search, or document sets. Attachments are not indexed through this personal live path. Each member who needs mailbox search connects their own account.

15.2 Administrator role and authentication

Sophea manages the Google OAuth app. The mailbox owner authorizes the connection. A Google Workspace administrator may need to allow the app or approve the restricted Gmail scope under the organization's app-access policy.

The grant requests `openid`, `email`, and `https://www.googleapis.com/auth/gmail.readonly`. You do not create or paste a Google client secret for the standard path.

15.3 Provider setup and permissions

The mailbox owner must sign in to the correct Google account and approve read-only Gmail access. Do not use a shared password. If the organization blocks third-party apps, ask the Google Workspace administrator to allow the Sophea app.

15.4 Setup form fields

The current personal setup has no editable form fields. It shows **Connect**, **Reconnect**, and **Disconnect** actions. The member selects the account in the Google consent flow. There is no workspace access or refresh setting for this personal connection.

15.5 Workspace and Team access

Workspace and Team access do not apply. The connection belongs to one member. Other members must connect their own mailboxes. Workspace admins cannot make a personal Gmail connection public or add it to a document set.

15.6 Test the connection

1. Ask a question whose answer exists only in the connected mailbox.
2. Confirm that the answer includes a Gmail message citation.
3. Confirm that the message does not appear in Knowledge Search or a document set.
4. Disconnect the member account and confirm that the live search no longer returns mailbox content.

15.7 Common errors

Error	Action
Google blocks the app	Ask the Workspace administrator to allow the Sophea OAuth app.
The wrong mailbox is connected	Disconnect it, then connect the correct Google account.
No message is found	Check the mailbox search terms and that the member can open the message.
A Team member expects the mail	Explain that Gmail is personal and live only. Each member must connect their own mailbox.

15.8 Reconnect, rotate, and remove

Use **Reconnect** when the Google grant expires. To change accounts, disconnect the old account first. Sophea manages the OAuth app rotation. **Disconnect** revokes the supported grant and removes the local personal connection.

For a classic workspace Gmail connector that still indexes mail, a permanent Google rejection stops the first failed polling attempt and shows the reconnect warning. Saving a fresh credential resumes indexing from the saved checkpoint.

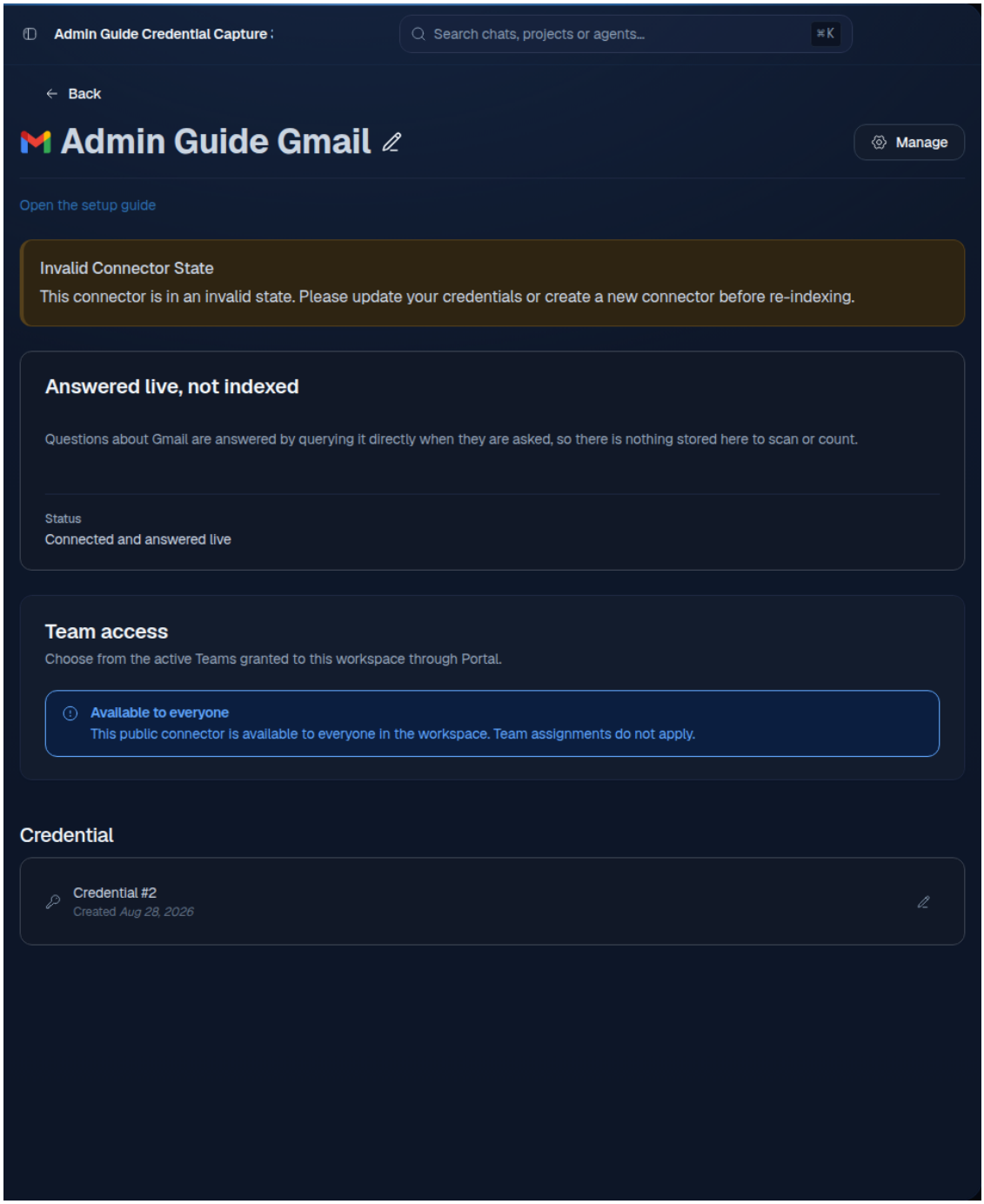

Figure 15.1: Classic Gmail connector warning after Google rejects its credential

15.9 Official provider documentation

- Gmail API scopes
- Gmail API authorization

- Google Workspace app access control

Chapter 16

Microsoft Outlook Connector

Microsoft Outlook in the current customer flow is a personal live connection. It answers questions from the signed-in member's mailbox at request time.

16.1 What is indexed or searched live

Outlook messages are searched live through the signed-in member's `/me` mailbox. Mail is not copied into the workspace search index, Knowledge Search, or document sets. Each member who needs mailbox search connects their own Microsoft 365 or Outlook.com account.

16.2 Administrator role and authentication

Sophea manages the Microsoft OAuth app. The mailbox owner authorizes the grant. A Microsoft 365 administrator may need to approve the app under the tenant's consent policy.

The delegated grant requests `openid`, `profile`, `offline_access`, `User.Read`, and `Mail.Read`. You do not create or paste a Microsoft client secret for the standard path.

16.3 Provider setup and permissions

The mailbox owner must sign in to the correct account and approve read-only mail access. Ask a Microsoft 365 administrator to grant consent when user consent is blocked. Personal Outlook.com accounts follow the same member-owned flow.

16.4 Setup form fields

The current personal setup has no editable form fields. It shows **Connect**, **Reconnect**, and **Disconnect** actions. The member selects the account in the Microsoft consent flow. There is no workspace access or refresh setting for this personal connection.

16.5 Workspace and Team access

Workspace and Team access do not apply. Other members must connect their own mailboxes. An admin cannot make a personal Outlook connection public or add it to a document set.

16.6 Test the connection

1. Ask a question whose answer exists only in the connected mailbox.
2. Confirm that the answer includes an Outlook message citation.
3. Confirm that the message does not appear in Knowledge Search or a document set.
4. Disconnect the account and confirm that live search no longer returns mail.

16.7 Common errors

Error	Action
Admin approval is required	Ask a Microsoft 365 administrator to approve the managed app.
The wrong account is connected	Disconnect it, then connect the correct Microsoft account.
No message is found	Check the mailbox search terms and that the member can open the message.
A Team member expects the mail	Explain that Outlook is personal and live only. Each member must connect their own mailbox.

16.8 Reconnect, rotate, and remove

Use **Reconnect** when the Microsoft grant expires. To change accounts, disconnect the old account first. Sophea manages OAuth app rotation. **Disconnect** removes the local personal connection.

16.9 Official provider documentation

- Microsoft Graph mail permissions
- Microsoft Graph mail API
- Microsoft identity admin consent

Chapter 17

Microsoft Outlook Calendar

Microsoft Outlook Calendar in the current customer flow is a personal live connection. It answers questions from the signed-in member's default calendar at request time. With the member's confirmation in chat, it can also create, reschedule, cancel, and RSVP to events.

17.1 What is indexed or searched live

Calendar events are searched live through the signed-in member's `/me` default calendar in Microsoft Graph. Events are never copied into the workspace search index, Knowledge Search, or document sets. Searches stay inside a bounded time window around today. Each member who needs calendar search connects their own Microsoft 365 or Outlook.com account, with one connection per member. When an event's details are not readable, the assistant sees a Private event placeholder with only start, end, and timezone.

17.2 Write actions

Members can ask the assistant to change their default calendar:

- Create an event (subject, time, timezone, location, attendees, Teams link).
- Reschedule or update an event the member organizes.
- Cancel an event the member organizes (attendees are notified).
- Accept, tentatively accept, or decline an invitation.

Every write follows the same two steps. The assistant first states exactly what will change and waits. Only after the member's explicit confirmation in the same chat does the change happen. Text inside events or invitations never counts as confirmation, and each confirmation covers one change.

Events are created with an idempotency reference, so a retried request after a network failure never books the same event twice. Create, reschedule, and cancel apply to the organizer's side only: the member cannot edit another organizer's event.

Every write is recorded with the actor, the action, the provider response, and the member's confirmation, for after-the-fact review.

17.3 Administrator role and authentication

Sophea manages the Microsoft OAuth app. The calendar owner authorizes the grant. A Microsoft 365 administrator may need to approve the app under the tenant's consent policy.

The delegated grant requests `openid`, `profile`, `offline_access`, `User.Read`, and `Calendars.ReadWrite`. You do not create or paste a Microsoft client secret for the standard path. Calendar consent is separate from Outlook Mail consent: connecting one does not grant the other. The granted scope also covers calendar writes. The assistant never writes on its own: it shows exactly what will change and only acts after the member's explicit confirmation.

17.4 Provider setup and permissions

The calendar owner must sign in to the correct account and approve calendar access. Ask a Microsoft 365 administrator to grant consent when user consent is blocked. Personal Outlook.com accounts follow the same member-owned flow.

17.5 Setup form fields

The current personal setup has no editable form fields. It shows **Connect**, **Reconnect**, and **Disconnect** actions. The member selects the account in the Microsoft consent flow. There is no workspace access or refresh setting for this personal connection.

17.6 Workspace and Team access

Workspace and Team access do not apply. Other members must connect their own calendars. An admin cannot make a personal Outlook Calendar connection public or add it to a document set. The connection reads the default calendar only; shared and secondary calendars are not read in this version.

17.7 Test the connection

1. Ask a question whose answer exists only in the connected calendar, for example "What do I have tomorrow?".
2. Confirm that the answer includes a calendar event citation.
3. Confirm that the event does not appear in Knowledge Search or a document set.
4. Disconnect the account and confirm that live search no longer returns events.

17.8 Common errors

Error	Action
The grant has expired or been revoked	Use **Reconnect** on the calendar card to grant access again.

Error	Action
The calendar permission is missing	Confirm that `Calendars.ReadWrite` is consented. Ask a Microsoft 365 administrator to approve the managed app when consent is blocked.
The wrong account is connected	Disconnect it, then connect the correct Microsoft account.
The provider is busy (Microsoft throttling)	Wait a moment and ask the question again.
An event shows as Private event	The event details are not readable. Only start, end, and timezone are visible.

17.9 Reconnect, rotate, and remove

Use **Reconnect** when the Microsoft grant expires. To change accounts, disconnect the old account first. Sophea manages OAuth app rotation. **Disconnect** removes the local personal connection.

17.10 Official provider documentation

- Microsoft Graph calendarView API
- Microsoft Graph event resource
- Microsoft Graph Calendars.ReadWrite permission
- Microsoft identity admin consent

Chapter 18

ClickUp Connector

ClickUp is a personal live connection. It searches the signed-in member's authorized workspace when a question is asked. It can also read Docs, update a task or Doc page, and create tasks, private Docs, and Doc pages after the member confirms the exact proposal. ClickUp content is not copied into the workspace search index.

18.1 What is indexed or searched live

Nothing is indexed. A text search scans up to 300 recently updated tasks and returns at most 25 matches. Search checks the task title, description, status, tags, assignees, List, Folder, and Space. The answer reports this coverage, so an empty result does not claim that older tasks do not exist.

The live reader can also open one task ID, task URL, numeric List URL, View URL, Doc URL, or Doc page URL. It accepts `https://app.clickup.com` and a one-label enterprise `*.clickup.com` host. The supplied host is validated but never fetched. Nous always calls the fixed ClickUp API origin.

18.2 Administrator role and authentication

Each member connects one ClickUp workspace with their own personal API token. The token must start with `pk_`. It is sent only in the setup request body and is stored as a private member credential. Workspace and Team sharing do not apply.

18.3 Provider setup and permissions

1. In ClickUp, open your personal settings and create or copy a personal API token under **Apps**.
2. Confirm that the ClickUp account can open the workspace and tasks you need. The same account must be allowed to edit or create the tasks, Docs, and Doc pages the member wants to manage through Nous.
3. In Nous, open **Connectors**, choose **ClickUp**, and paste the token.
4. Nous discovers the workspaces authorized for that token. Select one when several are returned. You cannot type a workspace ID.

If the token can access one workspace, Nous selects it automatically. Existing connections receive the supported write actions automatically. There is no reconnect step, permission toggle, or separate confirmation setting.

18.4 Setup form fields

Field	Required	Purpose
Connector name	Yes	A private label that helps the member recognize the connection.
ClickUp API Token	Yes	A personal token beginning with `pk_`. Whitespace is not accepted.
ClickUp workspace	Yes	One workspace returned by ClickUp for the submitted token.

There are no refresh, pruning, indexing, workspace-access, or Team fields.

18.5 Workspace and Team access

The connection belongs to one member. Other members must connect their own ClickUp accounts. An administrator cannot make it public, assign it to a Team, add it to a document set, schedule scans, pause it, or reindex it.

18.6 Supported confirmed changes

When a member requests a supported change, Nous validates it and replies in the normal chat with confirmation text generated from the exact staged operation. ClickUp remains unchanged at this point. The member can confirm naturally with replies such as **Yes**, **Go ahead**, or **OK, update it and give me the link**. Nous then applies that proposal once and returns the verified ClickUp link in the chat.

A question about the proposal does not discard it. The member can ask whether an append keeps existing images, receive an answer, and then confirm on the same conversation branch. Changing the target, destination, fields, or content creates a new proposal. Cancelling invalidates the old proposal. ClickUp content, older confirmations, assistant text, and tool output cannot authorize a change.

The confirmation identifies the action and target or destination. It states the requested names, subtitles, status, priority, visibility, and content mode. A replaced or superseded proposal requires a new confirmation and cannot silently select an older pending change.

Supported task fields are `name`, description, `status`, and `priority` (1 to 4, or cleared). Task descriptions and Doc page content can be replaced, appended, or prepended. For append and prepend, Nous reads the complete current content from ClickUp and composes the final value on the server. It does not rebuild content from search snippets, so existing Markdown and images remain intact. The final value is stored as one operation, so replay cannot add the same text twice.

Supported creation operations are:

- a task in an existing List, with a name and optional description, status, and priority;
- a private Doc in the connected workspace, an existing Space, Folder, or List, optionally with its first named page;
- a root page or nested page in an existing Doc, with a name and optional subtitle and content.

Nous uses only destinations that it resolved in the connected workspace. If a name matches several destinations, the member must choose one. Docs created by this connection are private; creating public Docs or changing permissions is not supported.

Each operation has a stable tenant-scoped audit record containing the request, confirmation version, member reply, actor, target snapshot, and safe provider result. The confirmation remains attached to the proposal the member saw. Replaying a completed or rejected operation makes no provider call. Nous rechecks existing targets immediately before writing and stops if they changed.

Creation requests are never blindly retried after a timeout or lost response. For a Doc with a first page, Nous records the created Doc before creating the page. If the page then fails, it returns the Doc link and can resume only the missing page when the recorded outcome proves that retry is safe. It never creates a second Doc to recover the page.

Delete, archive, move, comment, custom-field, assignee, date, attachment, public-Doc, permission, and bulk actions are not supported. Creating Lists, Folders, or Spaces is also outside this connection. Background, scheduled, multi-model comparison, Deep Research, Agent Service, and Digital Employee flows cannot write to ClickUp.

18.7 Supported links and testing

Test with non-sensitive tasks that the connected member can open:

1. Ask for a recently updated task by words from two different supported fields. Confirm that the result names the bounded scan.
2. Open a task by its ID, a short `/t/<task-id>` URL, and a workspace task URL.
3. Open one numeric List URL shaped like `/<workspace-id>/v/li/<list-id>`.
4. Open one View URL shaped like `/<workspace-id>/v/l/<view-id>`.
5. Open a Doc and one page shaped like `/<workspace-id>/v/dc/<doc-id>/<page-id>`.
6. Search for a phrase that has no match and confirm the query is not widened.
7. Request a task-description append on content containing an image. Confirm that the proposal does not change ClickUp, reply **OK, update it and give me the link**, and verify that the image and existing text remain.
8. Create a task in a test List, a private Doc with its first page, and a nested page. Confirm each proposal once and verify its returned link.
9. Ask a question about a proposal before confirming, then change another proposal and confirm that only the new version can run.
10. Try a dashboard URL shaped like `/<workspace-id>/v/db/...` and confirm that Nous rejects it safely.

List and View reads return at most 25 tasks and say when more tasks are available.

18.8 Common errors

Error	Action
No workspace appears	Check that the personal token belongs to the correct ClickUp account and that the account can open a workspace.
ClickUp rejects the token	Create a new personal token, paste it without spaces, and retry.
Permission error	Ask the ClickUp workspace owner to grant the member access to the required tasks.
Rate limit or temporary error	Wait briefly, then retry workspace discovery or the question.
A task is not found	Use a direct supported task link, or search with terms from a recently updated task.
Dashboard link is rejected	Use a task, List, View, Doc, or Doc page link. Dashboards are not supported.
A target changed before the write	Read the current item and make a new request with the intended values.
A proposal was replaced or cancelled	Ask Nous to prepare the current operation again; an older proposal cannot run.
A write or creation result is uncertain	Do not repeat it. Inspect the provider state and use the recorded result link when available.
A Doc exists but its first page failed	Open the returned Doc link. Ask Nous to resume only the missing page; do not request another Doc.

18.9 Reconnect and remove

Use **Reconnect** to replace a token or change the selected authorized workspace. Removing a current private live connection disconnects it; there is no indexed ClickUp content to remove.

An older ClickUp row that indexed tasks shows **Reconnect required**, never **Answered live**. Choose **Remove and reconnect**. Nous removes the old pair and its indexed content, waits for removal to finish, and then opens the new private setup. It does not convert the old ownership or credential in place.

18.10 Official provider documentation

- Create a personal API token
- Get authorized workspaces
- Get tasks
- Get View tasks
- Get Doc pages
- Get a Doc page
- Update a task
- Edit a Doc page
- Create a task
- Create a Doc
- Create a Doc page

Chapter 19

Jira Connector

Use this guide for the customer Jira connector. The standard customer path uses a Jira account and API token. It indexes issues and supported project data.

19.1 What is indexed or searched live

Nous indexes Jira issues, comments, project metadata, and supported issue fields that the configured Jira account can read. Use a project scope or JQL to limit the data.

19.2 Administrator role and authentication

The customer creates the Jira API token. Use a dedicated Atlassian service account with **Browse Projects** permission for every project to index. A Jira administrator should create or review the account and project permissions.

The API token is the current customer setup path. Do not use a Jira OAuth grant for this customer connector. Keep the token secret and rotate it in Atlassian.

19.3 Provider setup and permissions

Create an Atlassian account for indexing, grant it only the required Jira project permissions, and create an API token in the account security settings. Use the Jira Cloud base URL, for example `https://example.atlassian.net`.

19.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Jira Base URL	The Jira site base URL without a project path.
Jira User Email	Email for the Atlassian indexing account.

Field	What to enter
Jira API Token	API token created for that account.
Using scoped token	Turn on only for a scoped Jira API token.
How Should We Index Your Jira?	Choose Everything, Project, or JQL Query.
Project Key	The key when **Project** is selected, for example `PROJ`.
JQL Query	A query for the issues to index. Do not use time filters or `ORDER BY`.
Comment Email Blacklist	Optional email addresses whose comments should not be indexed.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

19.5 Workspace and Team access

Jira permissions control what the API account can read. Nous access controls who can search the indexed result. Use **Specific Teams** for a limited audience. Portal manages Team membership. A Team share cannot grant access to an issue outside the API account's Jira permissions.

19.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Open a known issue as the API account and note its key.
3. Search for the issue key and a phrase from its description.
4. Confirm that comments and fields expected by the selected scope appear.
5. Test access with the intended Workspace or Team members.

19.7 Common errors

Error	Action
Authentication fails	Check the Jira user email, API token, and site URL. Create a new token if the old one was revoked.
Project is empty	Grant the API account **Browse Projects** and issue access.
JQL setup fails	Remove time filters and `ORDER BY`, then validate the query in Jira.
Comments include unwanted users	Add their exact email addresses to **Comment Email Blacklist**.

19.8 Reconnect, rotate, and remove

Rotate the API token in Atlassian, save the replacement in the connector, and run a test search before revoking the old token. Recreate the connector when you need to change its permanent access mode. Before removal, check document sets, agents, and Team shares.

19.9 Official provider documentation

- Jira Cloud REST API
- Jira API tokens
- Jira permissions
- Jira advanced search with JQL

Chapter 20

Linear Connector

Use this guide for a workspace Linear connector. Nous indexes issues, projects, teams, and supported comments that the authorized Linear user can read.

20.1 What is indexed or searched live

The connector indexes the Linear workspace data visible to the connecting account. It does not expand access beyond that account's Linear membership.

20.2 Administrator role and authentication

Sophea manages the Linear OAuth app. A Linear workspace administrator should approve the connection and confirm that the authorizing user belongs to the teams and projects to be indexed.

The OAuth request uses the `read` scope. You do not create or paste a Linear client secret for the standard customer path.

20.3 Provider setup and permissions

Choose an account with the smallest Linear access that meets the requirement. Confirm that it can read the target teams and projects before authorizing.

20.4 Setup form fields

The current Portal form has these fields:

- **Name**: a unique workspace connector name.
- **Who has access**: choose **Everyone in the workspace** or **Specific Teams**.

There are no additional Linear source-scope fields in the current form. The provider grant defines the readable workspace content.

20.5 Workspace and Team access

Use **Specific Teams** to limit who can search Linear content in Nous. Portal manages Team membership. A Team share cannot grant access to a Linear issue that the OAuth account cannot read.

20.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Check for a known issue from each intended team.
3. Search for an issue identifier and a phrase from its description.
4. Test the result as an intended Workspace or Team member.

20.7 Common errors

Error	Action
Linear approval is blocked	Ask a Linear workspace administrator to approve the managed OAuth app.
A team is missing	Confirm the authorizing user's team membership and reconnect.
Search is empty	Check the connector state and that the account can open the issue in Linear.
The wrong workspace is linked	Disconnect and reconnect with a user from the intended Linear workspace.

20.8 Reconnect, rotate, and remove

Use **Reconnect** after changing Linear membership or when the grant expires. Sophea manages the OAuth app rotation. Before removal, check document sets, agents, and Team shares.

20.9 Official provider documentation

- Linear OAuth 2.0
- Linear API documentation
- Linear workspace membership

Chapter 21

HubSpot Connector

HubSpot is a private live connection. It searches the signed-in member's CRM when a question is asked. HubSpot records are not copied into the workspace search index.

21.1 What is indexed or searched live

Nothing is indexed. Nous can search deals, companies, contacts, and tickets, and can open one supported record or collection link. A search returns at most 25 records and reports HubSpot's exact match count separately.

Deal and ticket searches can use a pipeline plus a stage or status. Nous safely resolves names to HubSpot IDs before searching. Browser view filters in a pasted list or board URL are not known to Nous and are not applied. State the pipeline and stage or status in the question when those filters matter.

21.2 Administrator role and authentication

Sophea manages the HubSpot OAuth app. Each member connects one HubSpot account from the personal connector catalog. A HubSpot super admin or a user with app installation permission may need to approve the connection.

The OAuth grant requests read access for contacts, companies, deals, and tickets. There is no API key, access-token, client-secret, refresh schedule, or object-indexing form in Nous.

21.3 Provider setup and permissions

1. Confirm that the HubSpot user can read the required CRM records and properties.
2. In Nous, open **Connectors** and choose **HubSpot**.
3. Select **Connect HubSpot** and finish the HubSpot sign-in and approval flow.
4. Return to Nous and confirm that the connection shows the member's email.

Figure 21.1: Private HubSpot OAuth setup using sanitized local fixture data.

21.4 Setup form fields

There are no manual setup fields. HubSpot account identity, scopes, and the numeric Hub ID are read from HubSpot after OAuth. The Hub ID is used only for safe routing, while the member email is shown as the account label.

There are no refresh, pruning, indexing, workspace-access, or Team fields.

21.5 Workspace and Team access

The connection belongs to one member. Other members must connect their own HubSpot accounts. An administrator cannot make it public, assign it to a Team, add it to a document set, schedule scans, pause it, or reindex it.

21.6 Supported links and testing

Test with non-sensitive records that the connected member can open:

1. Search deals and confirm the exact total is separate from the returned records.
2. Search one deal or ticket pipeline and one stage or status.
3. Search companies, contacts, and tickets.
4. Open a typed ID such as `deals/12345` and an official record URL.
5. Open a modern object list or board URL and a legacy deals-board URL.
6. Try a link for another HubSpot account or an untrusted host and confirm that Nous rejects it before any provider request.

The next two images use deterministic, sanitized local fixture data. They show the product contract, not proof from a real HubSpot account. Real-provider proof is recorded separately during release verification.

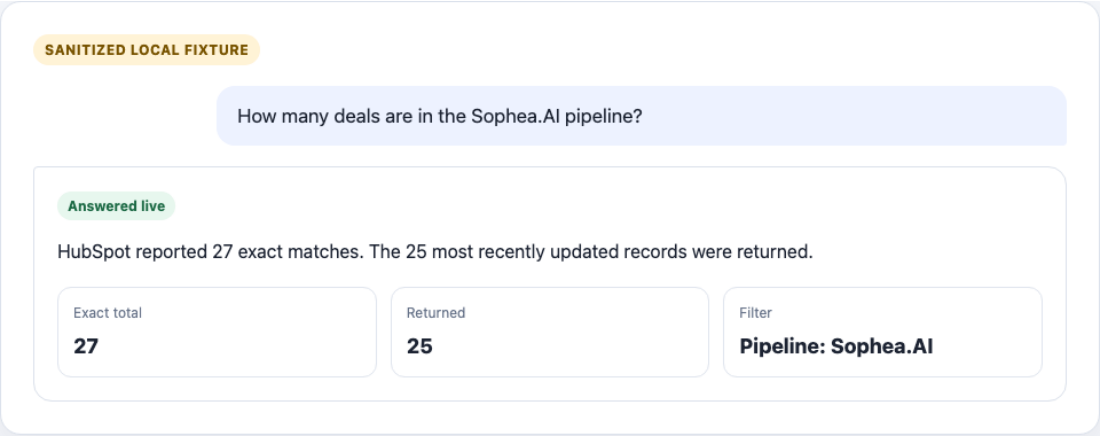

Figure 21.2: HubSpot exact-count answer from a controlled fixture with 27 matches and 25 returned.

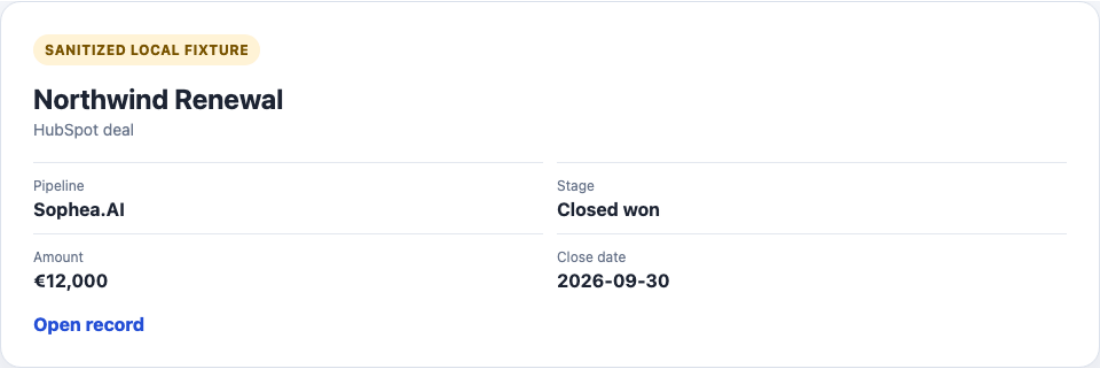

Figure 21.3: HubSpot record result from a controlled fixture.

21.7 Common errors

Error	Action
HubSpot approval is required	Ask a HubSpot super admin to approve the managed app.
Required scope is missing	Reconnect with a user who can grant read access for contacts, companies, deals, and tickets.
A record is missing	Check the connected user's HubSpot record permissions, then search the object type directly.
A pipeline or stage is not unique	Use the exact HubSpot pipeline or stage ID.
Browser filters were not applied	State the pipeline and stage or status in the question.
A link is rejected	Use an HTTPS `app.hubspot.com` or one-label `app-*.hubspot.com` record, list, or board URL for the connected account.
OAuth expired	Reconnect the same personal HubSpot account and test again.

21.8 Reconnect, rotate, and remove

Use **Reconnect** after the HubSpot grant expires or the account changes. Removing the current private live connection disconnects it. There is no new indexed HubSpot content to remove.

An older HubSpot connector that indexed records shows **Reconnect required**. Remove it and create a new private live connection. The cleanup deletes the old indexed pair without deleting the new personal live connection.

21.9 Official provider documentation

- HubSpot OAuth
- HubSpot CRM search
- HubSpot CRM pipelines
- HubSpot user permissions

Chapter 22

Salesforce Connector

Use this guide for a customer Salesforce connector. The customer creates a Salesforce integration user credential. Nous indexes the selected Salesforce objects.

22.1 What is indexed or searched live

The connector indexes Salesforce objects and supported child records that the configured user can read. The simple form lets you list requested objects. The advanced form accepts a JSON query configuration.

22.2 Administrator role and authentication

A Salesforce administrator should create or review the integration user and grant it the least-privilege read permissions needed for the selected objects. The customer supplies the Salesforce username, password, and security token. Use a sandbox checkbox when the source is a Salesforce sandbox.

22.3 Provider setup and permissions

Create a dedicated integration user. Grant object and field read access for the data to index. Reset the security token when the user's trusted network changes or when policy requires it. Keep the password and token in the credential store, not in a document or chat.

22.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Salesforce Username	The integration user's Salesforce username.
Salesforce Password	The integration user's password.
Salesforce Security Token	The security token for that user.
Configuration Type	Choose **Simple** or **Advanced**.

Field	What to enter
Requested Objects	In Simple mode, list object names such as `Account` or `Opportunity`, one per line.
Custom Query Config	In Advanced mode, enter the JSON object and child-field configuration.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

If **Requested Objects** is empty, Nous uses its default object selection. Validate custom JSON before saving and keep only fields the integration user can read.

22.5 Workspace and Team access

Salesforce permissions control what the integration user can read. Nous access controls who can search the indexed result. Use **Specific Teams** for a limited audience. Portal manages Team membership. A Team share cannot grant access to fields the Salesforce user cannot read.

22.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Search for a known record from each selected object.
3. Search for a distinctive field value and confirm the record link.
4. Test as an intended Workspace or Team member.

22.7 Common errors

Error	Action
Login fails	Check username, password, security token, and that the credential belongs to the intended Salesforce production account.
An object is empty	Grant the integration user object and field read permission.
Custom JSON is rejected	Validate JSON, use singular object names, and remove unsupported fields.
The security token is invalid	Reset it in Salesforce, save the replacement, and test again.

22.8 Reconnect, rotate, and remove

Rotate the Salesforce password or security token in Salesforce first. Save the replacement in the connector and run a test search before disabling the old credential. Before removal, check document sets, agents, and Team shares.

22.9 Official provider documentation

- Salesforce REST API
- Salesforce user security token
- Salesforce object permissions

Chapter 23

Amazon S3 Connector

Use this guide for an Amazon S3 connector. Nous indexes objects in the selected bucket and optional prefix.

23.1 What is indexed or searched live

The connector indexes supported objects that the AWS credential can list and read. A prefix limits the object key path. S3 object ACLs do not replace the Nous workspace and Team access rules.

23.2 Administrator role and authentication

An AWS administrator or IAM administrator should create a dedicated least- privilege integration identity. The customer creates the AWS access key and secret for the standard customer path. Never use a root access key.

The identity needs `s3:ListBucket` for the bucket and `s3:GetObject` for the selected objects. Add `s3:GetBucketLocation` when required by the AWS account policy.

23.3 Provider setup and permissions

Create an IAM policy limited to the target bucket and prefix. Create an IAM user access key for the customer connector. Store the secret once, in the connector credential form, and rotate it under the normal AWS key policy.

23.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Bucket Name	The S3 bucket name, without `s3://`.
Prefix	Optional object-key prefix, such as `policies/2026/`.

Field	What to enter
AWS Access Key ID	The customer-created access key ID.
AWS Secret Access Key	The matching secret.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

23.5 Workspace and Team access

AWS IAM controls what Nous can read. Nous access controls who can search the indexed result. Use **Specific Teams** for a limited audience. Portal manages Team membership. A Team share cannot grant access to an object outside the IAM policy.

23.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Confirm that a known object in the selected prefix appears.
3. Search for a phrase from the object content.
4. Test as an intended Workspace or Team member.

23.7 Common errors

Error	Action
Access denied on list	Add `s3:ListBucket` for the bucket and check the prefix condition.
Access denied on read	Add `s3:GetObject` for the selected object path.
Bucket is not found	Check the exact bucket name and AWS account.
Key is rejected	Create a new access key, save it, test, then revoke the old key.

23.8 Reconnect, rotate, and remove

Create and test the replacement AWS access key before retiring the old one. Use **Reconnect** when the connector supports the action. Before removal, check document sets, agents, and Team shares. Remove the old IAM policy or key after the connector is no longer using it.

23.9 Official provider documentation

- AWS access keys
- Amazon S3 policy actions
- AWS IAM best practices

Chapter 24

Google Cloud Storage Connector

Use this guide for a Google Cloud Storage connector. Nous indexes objects in the selected bucket and optional path prefix.

24.1 What is indexed or searched live

The connector indexes supported objects that the Google Cloud Storage HMAC credential can list and read. A path prefix limits the object keys. GCS access is separate from Nous Workspace and Team access.

24.2 Administrator role and authentication

A Google Cloud administrator or Storage administrator should create an HMAC key for a dedicated service account or user with least-privilege access. The customer creates the HMAC access key ID and secret. Do not use a personal key when a dedicated identity is available.

Grant the identity permission to list the bucket and read the selected objects. At minimum, review `storage.objects.list` and `storage.objects.get` for the bucket and prefix.

24.3 Provider setup and permissions

Create the HMAC key in the Google Cloud project that owns the service account. Apply a bucket IAM policy limited to the selected bucket and prefix. Copy the secret only into the credential form and rotate it under your cloud policy.

24.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Bucket Name	The GCS bucket name, without `gs://`.

Field	What to enter
Path Prefix	Optional object-key prefix, such as `policies/2026/`.
HMAC access key ID	The customer-created GCS HMAC access key ID.
HMAC secret access key	The matching HMAC secret.
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.

24.5 Workspace and Team access

GCS IAM controls what Nous can read. Nous access controls who can search the indexed result. Use **Specific Teams** for a limited audience. Portal manages Team membership. A Team share cannot grant access beyond the bucket IAM policy.

24.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Confirm that a known object in the selected prefix appears.
3. Search for a phrase from the object content.
4. Test as an intended Workspace or Team member.

24.7 Common errors

Error	Action
Bucket list is denied	Grant bucket list permission to the HMAC identity and check the bucket name.
Object read is denied	Grant object read permission for the selected bucket and prefix.
HMAC key is invalid	Check the key ID and secret, then create and test a replacement key.
No documents appear	Check the prefix and wait for daily refresh and search indexing.

24.8 Reconnect, rotate, and remove

Create and test a replacement HMAC key before disabling the old one. Save the replacement in the connector and run a search. Before removal, check document sets, agents, and Team shares, then disable the old key.

24.9 Official provider documentation

- Manage HMAC keys
- Cloud Storage IAM permissions

- Cloud Storage roles

Chapter 25

Web Connector

Use this guide for a public website connector. Web has no provider credentials. Nous fetches and indexes public pages that the website allows it to read.

25.1 What is indexed or searched live

The connector indexes pages from the configured public base URL. It follows the selected scrape method and supported links. It cannot index a page that needs login, a private network route, or a blocked request.

25.2 Administrator role and authentication

No credentials are required. A website administrator should approve the crawl, check the site's robots and rate-limit policy, and provide a stable public URL. Do not use the connector to bypass access controls.

25.3 Provider setup and permissions

Confirm that the base URL is public HTTPS and that the site's robots policy allows the intended crawl. Use a small scope first. Remove private query strings and tracking parameters from the base URL.

25.4 Setup form fields

Field	What to enter
Name	A unique workspace connector name.
Base URL	The public HTTPS URL to crawl, for example `https://docs.example.com/`.
Scrape Method	Choose **recursive**, **single**, or **sitemap**.
Scroll before scraping	Turn on when content loads only after page scrolling.

Field	What to enter
Who has access	Choose **Everyone in the workspace** or **Specific Teams**.
URL rewrites	Optional. Property pairs: source prefix target prefix. Advanced section.

Use **single** for one page, **recursive** for linked pages under the site, and **sitemap** when the site publishes a sitemap with the desired URLs.

25.4.1 URL rewrites for mirrored sites

Open **Show advanced settings** in the Portal setup dialog to find **URL rewrites**. Use it when the crawler reaches the site under one address but people open the same pages under another, as with an internal mirror and its public site. Each pair maps a source prefix to a target prefix: Nous fetches the page under the source prefix but stores and links it under the target prefix. For example, the pair `http://intranet.internal/` `https://docs.example.com/` lets Nous read the intranet copy while search results and citations point at the public address. Rewrites change stored document links only; they do not change which sites the connector may read.

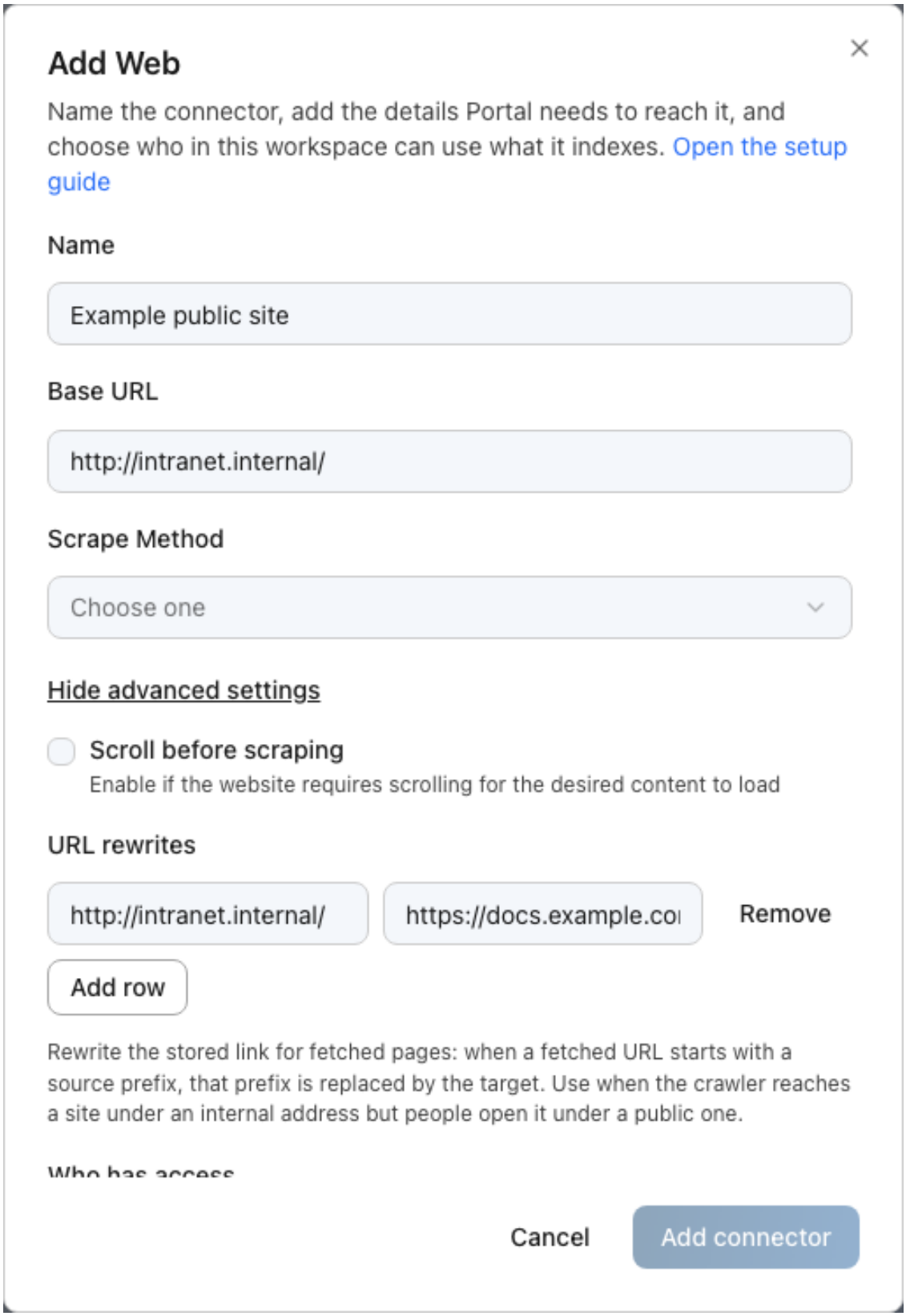

Figure 25.1: Web connector setup dialog with the advanced section open and one URL rewrite pair filled in

25.5 Workspace and Team access

The web source is public at the provider. Nous access still controls who can search the indexed copy. Use **Specific Teams** when the content is for a limited workspace audience. Portal manages Team membership.

25.6 Test the connection

1. Save the connector and wait for source sync and search indexing.
2. Check that the expected URL appears in the document list.
3. Search for a distinctive phrase from the page.
4. Check a linked page when using recursive mode.
5. Test the intended Workspace or Team audience.

25.7 Common errors

Error	Action
URL is not public	Publish the page over HTTPS or use a different public URL.
Sitemap returns no pages	Check the sitemap URL and that it lists the intended pages.
Page needs a login	Web has no credentials. Publish a safe public page or use an authenticated provider connector.
Dynamic content is missing	Enable **Scroll before scraping** or use a source that exposes the content in HTML.

25.8 Reconnect, rotate, and remove

Web has no credential to rotate. Edit the base URL or scrape method, then run a new sync. Before removal, check document sets, agents, and Team shares.

25.9 Official provider documentation

- Google Search Central robots.txt
- Sitemaps overview
- MDN HTTPS overview

Part IV

Part IV: Knowledge and Agents

Chapter 26

Document sets

A document set is a named, reusable scope of indexed content. Use document sets to give Knowledge Search and agents a curated slice of your workspace, rather than searching every connector at once.

This page covers what to do on the **Document Sets** page at `/app/admin/documents/sets`. Reach it via the **Knowledge** flyout in the left sidebar, then click **Document Sets**. Before you create a set, make sure the underlying source is already wired up. See Connectors for that step.

26.1 What document sets do

A document set groups one or more connectors into a scope your users can pick later. Use them to:

- Define a topical scope such as `Finance policies` or `Support playbooks` that combines several connectors.
- Provide a curated slice of indexed content that agents and Knowledge Search can attach without exposing every connector at once.
- Provide a stable handle that agents and Knowledge Search can reference, even as connectors come and go.

Document sets do not re-index your data. They reference content that connectors have already indexed. If a connector is still indexing, documents will appear in the set as they become available.

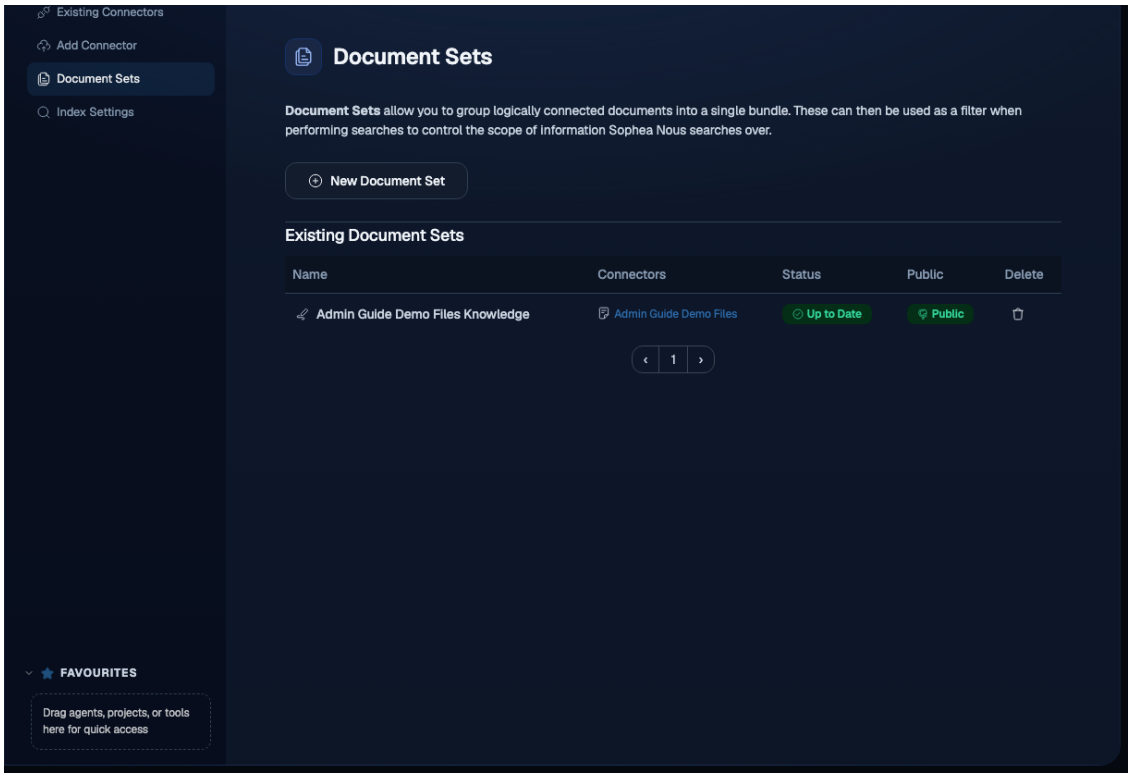

Figure 26.1: Document Sets list page showing existing sets with Name, Connectors, Status, Public, and Delete columns

The **Existing Document Sets** table shows each set's name, the connectors it draws from, its status, and whether it is public. Use the trash icon in the **Delete** column to remove a set.

26.2 Create a document set

Open the **Knowledge** flyout in the left sidebar and click **Document Sets**. Then click **New Document Set**.

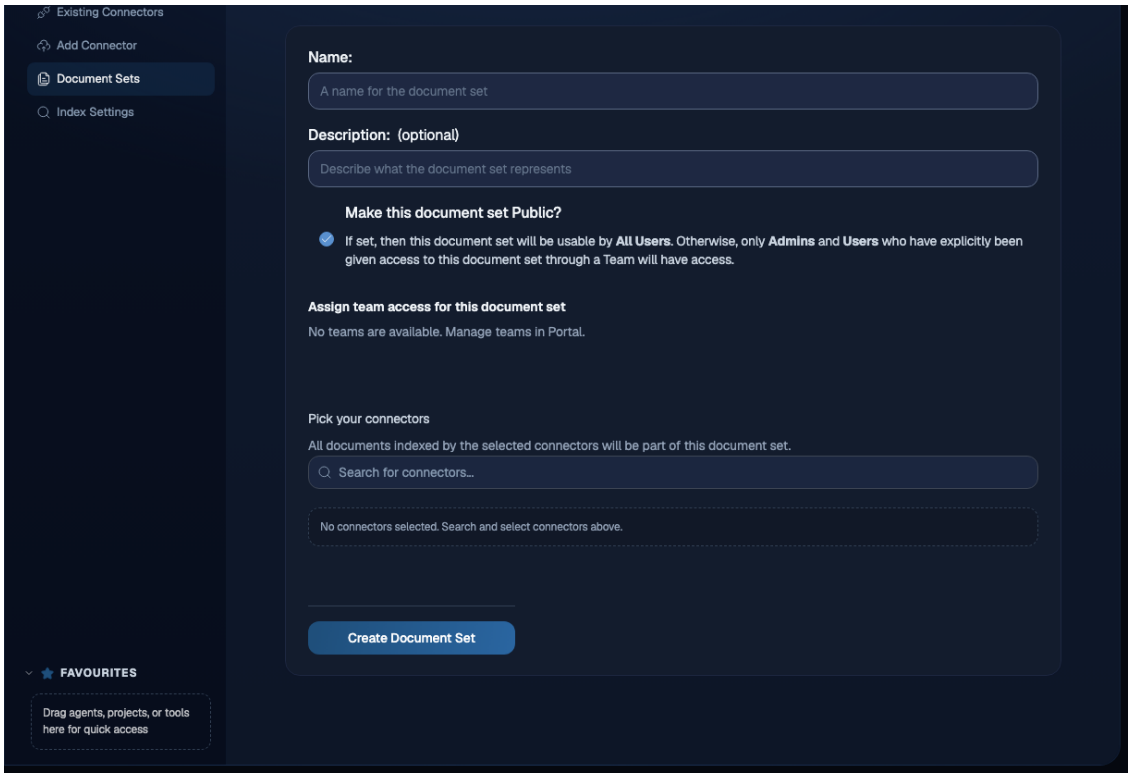

Figure 26.2: New Document Set form with Name, Description, and connector search fields

Fill in the form:

1. **Name.** Use a short, business-friendly label your users will recognise. Names are unique within a workspace.
2. **Description** (optional). One sentence explaining what is inside. This shows up in the picker when members attach the set to a chat or agent.
3. **Pick your connectors.** Use the search box to find and select the connectors whose content should be included. You can select any connector that has at least one indexed document. Mix and match across types if it serves the scope (for example, a Confluence space plus an uploaded policy PDF).
4. Click **Create Document Set**.

Saving the set kicks off a background sync that compiles the document list. The row shows a syncing state until the sync finishes, then flips to **Up to Date**.

> Note
>
> You must create document sets explicitly. Connectors do not generate starter document sets for you. After a file connector finishes indexing, return to this page and create the set yourself.

26.2.1 Add uploaded files

Use a file connector when the content exists only on your computer or in a local export. As an admin, select **Add Connector**, choose **File**, and upload the files. Members create their own file connectors

from **Connectors**; those are private to the member or shared with the member's Teams, so only an admin-created connector can be workspace-wide. Roles, limits, and sharing rules: File connectors.

Use safe business filenames and do not upload secrets, personal data, or temporary exports. After the file connector is indexed, create a document set here and select that connector.

A file connector holds uploaded documents rather than reading a provider, so it is not in the connector comparison.

26.3 Scope and Team access

A document set has its own visibility in addition to the access rules on its connectors:

- **Everyone in this Workspace** makes the set available to all workspace members.
- **Private** keeps the set out of the workspace-wide picker.
- **Shared with Team** makes a private set available through one or more active Portal Teams.

When you create or edit a private set, use **Add Teams** to select the Portal Teams that should receive it. Manage Team identity and membership in Portal. Nous owns only the binding between the synchronized Team and the document set.

Sharing a compatible private document set with a Team adds that Team to the eligible ACL for its connector documents without making the content workspace-public. Selecting that document set during search is a separate operation: the selection narrows the request and cannot bypass the resulting ACL.

> Important
>
> Provider-native file permissions are reproduced only for SharePoint and Google Drive connectors configured with **Sync permissions**. Other connectors use workspace-public access and supported Nous user or Portal Team principals. A document set can narrow this access, but cannot widen or bypass it. Content without a supported principal fails closed. Verify a private set by searching as a real member of each intended Team.

26.3.1 Team-share notifications for document sets

When a private document set is shared with a Portal Team, Nous sends an email and adds one in-app Team-share notification for each active Team member who newly gains effective access to the set. The recipient is the Team member, not the administrator who saved the change. A member who already has access through a direct document-set share or another active Team does not receive a duplicate. The actor, workspace admins who already have access, inactive Team members, and members in another tenant are not recipients. Public sets and direct-only shares do not create Team-share notifications.

Open the in-app inbox from the account menu: click your sidebar avatar, then choose **Notifications**. The avatar badge and menu row show the unread count. Opening the inbox refreshes the access check but does not dismiss a row. Use **Dismiss notification** to clear unread state; dismissed rows remain visible as history and stay dismissed after refresh and sign-in.

The inbox covers Team-share events for Projects, private Connectors, and private Document Sets. An accessible row shows its resource link, the Team names associated with the share, and the received time. Retrying the same Team-share mutation is deduplicated. Removing a Team and adding it again is a new grant and creates a new event rather than replaying the old row.

If the document set is revoked, deleted, or otherwise inaccessible, its row stays in history but the fresh open-time access check removes the set name, connector details, Team names, and link. The row shows only safe unavailable wording. A later regrant creates a new event with fresh details; it never restores private details in the old unavailable row.

Document-set sharing has two boundaries: the set's own visibility and the visibility of each connector it references. Sharing a set with a Team does not make its connectors public, and selecting a set in Knowledge Search does not bypass either ACL. Test retrieval as a real member of the intended Team after the set reaches **Up to Date**.

26.4 Status badges

Each row carries a status badge and a **Public** badge. Use the status to decide when a set is safe to share.

Figure 26.3: Document set row showing an Up to Date status badge and a Public badge

Badge	Meaning	What to do
Up to Date	The set is compiled and matches your current configuration.	Safe to attach in Knowledge Search and agents.
Syncing	The set is compiling its document list after a create or edit.	Wait. Do not advertise it to users yet.
Deleting	A delete request is in flight.	Wait for the row to disappear.

The **Public** badge indicates the set is visible to all workspace members. A private set instead shows its configured Team access in the editor. An **Up to Date** badge means the document set itself is compiled. It does not guarantee that every connector behind it has finished search indexing. If a connector is still processing, new searchable documents join the set as they become ready. Check the Connectors page if you expect more content than the set shows.

If a set stays in a syncing state longer than a few minutes for a small scope, check the underlying connectors first. A connector stuck on its own indexing run will hold up its document sets.

26.5 Edit or delete a set

Click the pencil icon on a row to open the editor. You can change the name, description, and connectors. Saving puts the set back into a syncing state while it recompiles.

To delete a set, click the trash icon in the **Delete** column. The row enters the **Deleting** state and disappears once cleanup finishes.

> **Warning**
>
> Deletion is irreversible. Any agents or saved Knowledge Search configurations that reference this set will lose it. Members will see fewer options in their pickers immediately. Review attached agents before you delete a widely-used set, then recreate it from scratch if you change your mind.

Deleting a document set does not delete the underlying connectors or their indexed content. The data stays in your workspace and is still searchable from any other set that references the same connectors.

Once a set reaches **Up to Date**, head to Knowledge Search to attach it to a chat or to an agent's knowledge pane.

Chapter 27

Searching Your Knowledge Base

Nous exposes connected content through **Knowledge Search** in the chat composer's **Tools** menu. Knowledge Search starts enabled over **All sources** in every chat. The model decides whether that ambient search is useful, so users do not need to enable it before asking a knowledge question. The same menu can turn search off or narrow a message to document sets and connector source types.

27.1 Choose a search scope

Open a chat and use the composer:

1. Click **Tools**.
2. Confirm **Knowledge Search** is enabled. Turn it off only when the current chat should not use connected knowledge.
3. Open **Scope** on the Knowledge Search row.
4. Choose **All sources**, one or more **Document sets**, one or more connector **Sources**, or a combination of sets and sources.
5. Type the question and send it.

All sources is selected when no narrower scope is active. It authorizes ambient search across every eligible source the current user and agent can access, while the model decides whether to search. A selected document set or source makes Knowledge Search the first ordered tool step for that message. The **Document sets** section lists reusable curated scopes. The **Sources** section lists connected source types directly, including connectors that are not part of a document set. A non-default agent may show a smaller list based on the knowledge assigned to that agent.

Create a document set when users or agents need a stable, named scope. A document set is optional for broad search: content from a searchable connector remains available through **All sources** or its connector source.

> Tip
>
> For verification after a new connector goes live, select its connector source or a document set that contains it. A narrow scope removes unrelated results and gives you a cleaner signal.

You can change the scope mid-thread. Earlier messages keep the citations they were generated with; later messages use the updated scope. Knowledge Search and its scope remain active for later mes-

sages in the current chat. Starting or switching chats, assistants, or projects, and refreshing the page reset Knowledge Search to its default enabled **All sources** state.

Deep Research runs alone for its message. It temporarily suppresses Knowledge Search without changing the saved Knowledge Search switch or scope, which returns after Deep Research finishes or is turned off.

> Note
>
> The composer supports at most eight ordered tool steps. Ambient Knowledge Search over **All sources** does not consume a step. A narrowed Knowledge Search scope consumes one step because search must run first.

27.2 Citation reveal

Every answer that consumed indexed content shows citations inline. Click a citation chip in the answer to open the **Cited Sources** panel.

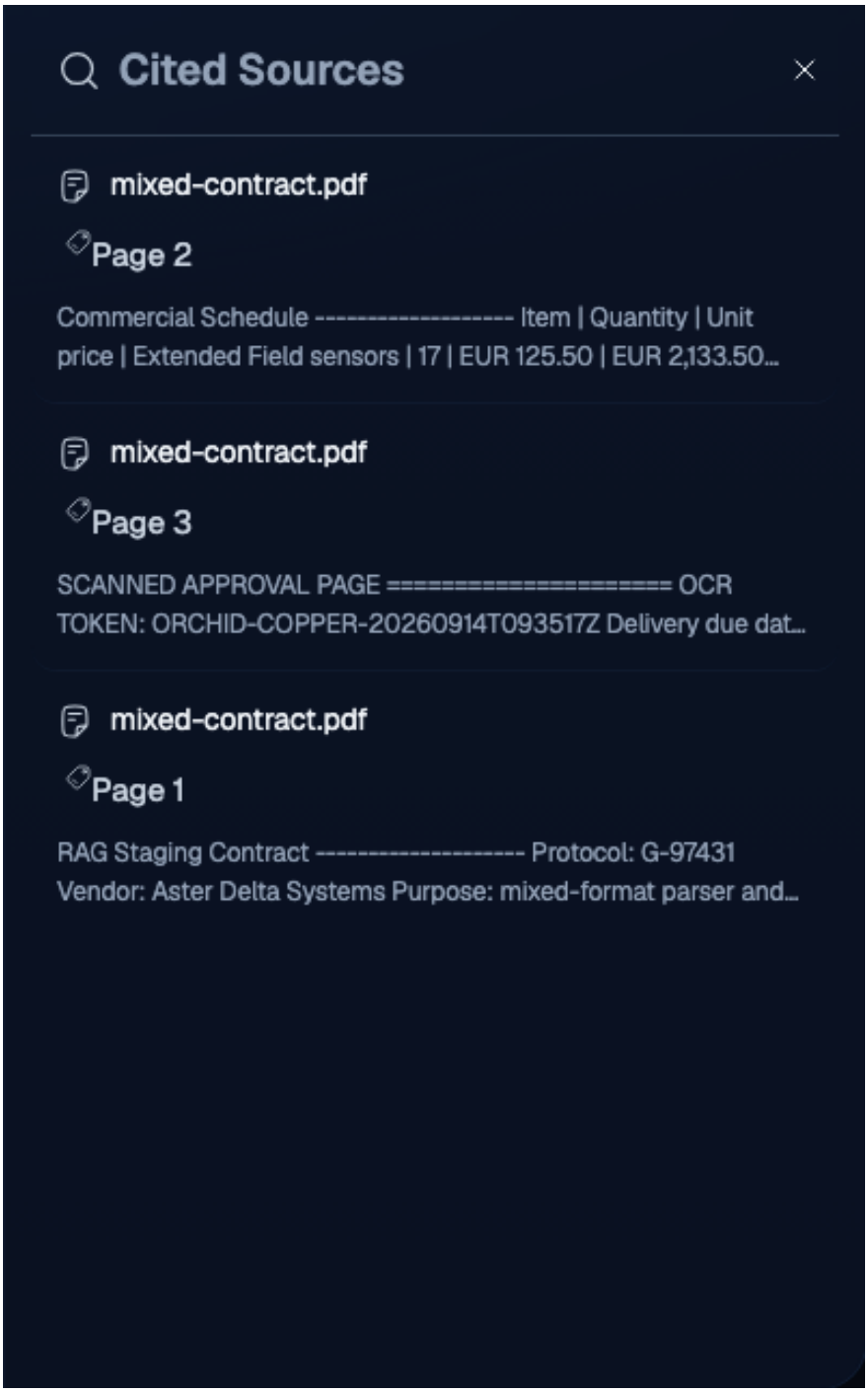

Figure 27.1: Cited Sources shows the original PDF filename, cited pages, and retrieved excerpts.

Each citation shows:

- the source filename and connector
- available page or chunk metadata, such as a page number or source label

- the raw text excerpt when content was retrieved; meeting inventories show dates and source metadata, not transcripts

Use the citation panel as your verification ground truth. The answer text can paraphrase, but the citation chunks are exactly what the retrieval pipeline returned. If a citation points to the wrong document, the problem is retrieval, not generation. Tune the document set or the connector, not the prompt.

When you are verifying a brand-new set, expand each citation and confirm the chunk text matches what you expect from the source file. Title-only matches can be misleading: two files with similar titles can collide, and the excerpt tells you which one was actually retrieved.

> Note
>
> An empty historical meeting list has no source citations. For other answers without citations, re-open **Tools** > **Knowledge Search** > **Scope** and confirm the intended source or set is selected.

> Note
>
> Knowledge Search answers cite the source evidence available for that answer. A multi-part question can combine evidence from documents, text passages, and images, with citations kept beside the relevant claims. If a workbook is large or only partly available, the answer must say that the evidence is partial; it does not establish a whole-workbook total. If no evidence is retrieved, that means the answer was not found in the available results, not that the document does not exist.

The original filename identifies each source, including images and spreadsheets. Worksheet names and section headings provide additional context beneath the filename. PDF citations show available source page numbers; worksheet positions are not PDF page numbers. If a search cannot finish, retry after its error message appears.

27.3 Historical meeting lists

Ask Knowledge Search for completed Sophea Meet meetings in a date range, for example, “List all meetings from 10 through 25 August, with dates and sources.” Both dates are included in your local timezone. The list uses accessible indexed meetings, not a relevance-ranked transcript search or the first page of the live app.

Each result keeps its meeting date and source citation. Lists return the oldest 500 matching indexed meetings at most. If the answer says results are partial, narrow the date range; the returned count is not the total. Older records without a known meeting date appear after dated meetings only when you ask for all history, labeled “Date unavailable”. They are not included in date-range searches. An empty range means no matching accessible indexed meetings, not that the meeting never happened.

For what was said, ask a transcript-content question. For today’s schedule or current meeting status, use the connected live app instead.

27.4 Permissions model

Knowledge Search evaluates the requesting user’s workspace identity and active Portal Team principals on every query. A selected scope can narrow eligible content, but it cannot bypass an ACL.

Current indexed visibility comes from:

- connector or document-set access set to **Everyone in this Workspace**
- supported direct user principals
- active Portal Teams bound to a private connector or compatible document set
- source-system document permissions for SharePoint and Google Drive connectors configured with **Sync permissions**

Sharing a compatible private document set with a Team can add that Team to the eligible ACL for its connector documents without making the content workspace-public. This is different from selecting a document set as the search scope: selection only narrows the request and never grants access or bypasses the resulting ACL.

Provider-native per-document permissions are not a general Nous search ACL path. They are mirrored only for supported SharePoint and Google Drive connectors configured with **Sync permissions**. Other sources continue to use the connector and document-set access paths above. A private document without a supported Nous principal is excluded from results rather than exposed publicly.

> Note
>
> Workspace administrators do not automatically bypass search ACLs. Test restricted knowledge with a real member of the intended Team, not only with the administrator who configured the connector.

If one member can retrieve a file and another cannot, compare their active Portal Teams and the connector or document-set access bindings in Nous. Also check whether the member still has another workspace access source after a recent Portal change.

27.5 Troubleshoot missing results

When search returns nothing, work through the list in order. Stop at the first step that resolves the issue.

1. **Search indexing.** Open the connector detail page and check the **Search indexing** panel. Confirm the expected files are counted as **Searchable** and review any **Needs attention** categories. Workspace administrators can retry failed documents from **Needs attention**; rows that remain after the retry need action in the source.
2. **Scope.** Re-open **Tools** > **Knowledge Search** > **Scope** and confirm the intended connector source or document set is selected. Switch to **All sources** to distinguish a scope mistake from an ingestion problem.
3. **Document set configuration.** If you selected a document set, confirm it is **Up to Date** and includes the expected connector.
4. **Query specificity.** Re-ask with an exact filename, title, or quoted phrase you know exists in the source.
5. **Permissions.** For a **Sync permissions** connector, check the source grant and **Last completed permission sync**. For other connectors, check whether the connector or compatible document set is workspace-public or Team-shared, and whether the affected member belongs to an active attached Team. Credential ownership alone does not grant search access.
6. **Recent reindex.** A connector that recently finished a full reindex can return stale or empty results until the reindex fully completes. Open the connector from the **Existing Connectors** page and check whether an indexing job is still running.

If none of those steps resolve the issue, capture the query, selected scope, and expected filename, then raise the issue with platform support.

27.6 Where to go next

- Document Sets for creating, sharing, and editing the sets you searched here.
- Agents for wiring a verified document set into a guided assistant so users do not have to pick the set manually.
- Search Verification and Troubleshooting for the Document Explorer, Document Feedback, and a troubleshooting sequence for missing or wrong results.

Chapter 28

Agents

> **Note**
>
> Workspace members can create and manage their own agents from **Agents** at `/app/agents` when the product is enabled. Workspace admins also receive the management view at `/app/admin/agents`, where they can administer workspace agents. Portal controls whether the Agents product and Sophea-published agents are enabled for the workspace.

Agents give your users a prepared assistant instead of a blank chat. An agent bundles instructions, knowledge, tools, and starter prompts behind a single name so non-technical users get a guided experience.

There are two flows in the product, and you need to keep them separate in your head:

- **Published Sophea agents** ship inside the platform. Presentation Agent, Financial Agent, Web Search, and others are available here. Your Portal administrator chooses which Sophea-published agents are available in your workspace via the `external_agents` feature. You do not build these inside Nous, and you do not attach document sets to them from the workspace.
- **Workspace-created agents** are agents you build inside Nous. They use your own document sets (see Document sets), your own actions, and your own instructions. This is the flow the rest of this page walks you through.

If you only need ad-hoc exploration, plain workspace chat is enough. Reach for a workspace-created agent when the same task needs to be repeated by several users with the same scope and tone.

When you select a published agent from **Tools** **Agents** in the chat composer, it stays selected for subsequent messages in the same chat, including when a run continues in the background. Turn the agent off in the picker when you no longer want to use it. Switching chat, assistant, or project context can clear the selection, as can choosing Deep Research.

28.1 Projects

> **Tip**
>
> Projects are a user-facing feature, not an admin-only one. Any workspace member can create and use projects directly from the main sidebar.

Projects let users group related chats under a shared set of instructions and context files. Instead of starting every conversation from scratch, a project carries its own instructions and uploaded files into every chat opened under it.

Click the **Projects** icon in the main left sidebar to open the projects panel.

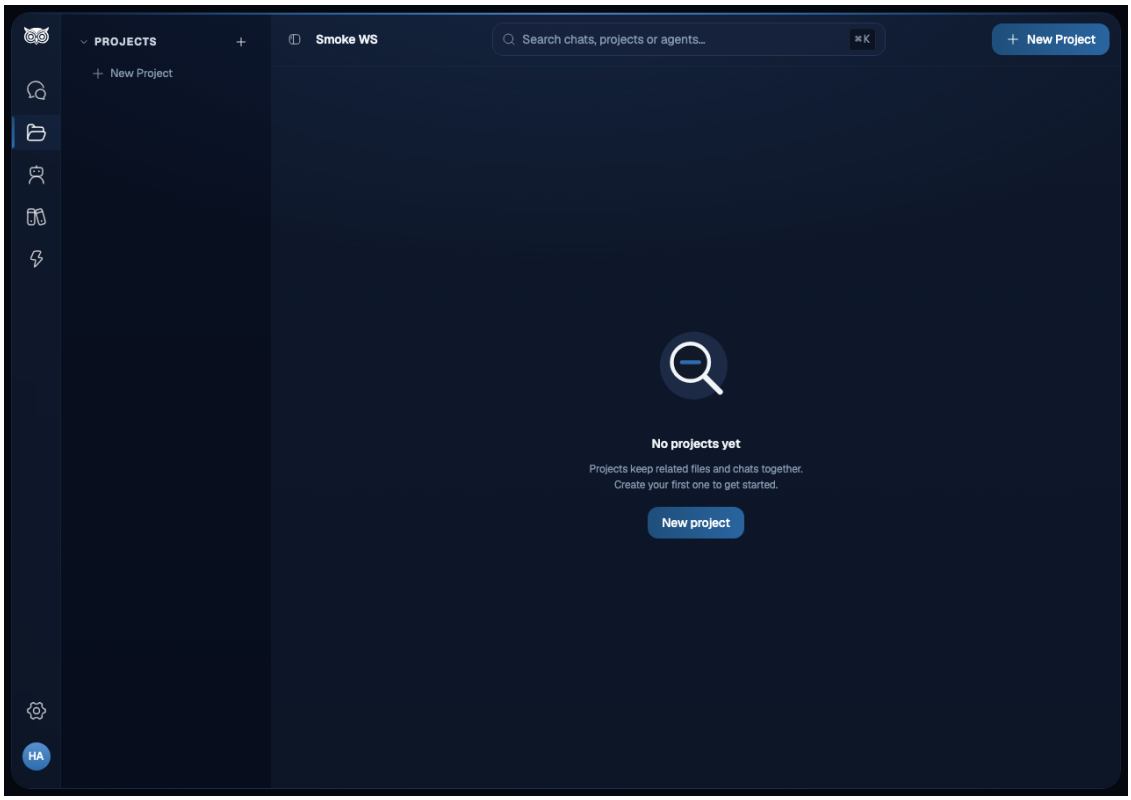

Figure 28.1: Projects panel open in the main sidebar, showing a New Project button and the workspace chat area.

To create a project:

1. Click **New Project** at the top of the projects panel.
2. Give the project a name.
3. Add shared instructions in the instructions field. These apply to every chat opened inside the project.
4. Upload any context files the project should always have available.
5. Save.

Once a project exists, open a new chat from inside it. All chats in that project inherit its instructions and files automatically.

Use projects when a team member regularly works in the same context, for example a standing client engagement, a recurring research topic, or an ongoing audit. Projects reduce the overhead of re-establishing context at the start of every session.

28.1.1 Project Team-share notifications

When a Project is shared with a Portal Team, Nous sends an email and adds one in-app Team-share notification for each active Team member who newly gains access to that Project. The recipient is the Team member, not the person who shared it. A member who already has access directly or through another active Team receives no duplicate. The actor, workspace admins who already have access, inactive Team members, and users in another tenant are not recipients.

Find the in-app inbox by clicking your sidebar avatar and choosing **Notifications**. The avatar badge and the **Notifications** row show the unread Team-share count. Opening the inbox refreshes access but never dismisses a row. Click **Dismiss notification** to clear unread state; the row remains in history and the dismissal persists across refreshes and sign-in.

The inbox covers Team-share events for Projects, private Connectors, private Document Sets, and private agents. An accessible Project row shows its link, the Team names associated with the share, and received time. Retrying one Team-share mutation is deduplicated. Removing a Team and sharing the Project with it again creates a new grant and a new event. If a Project or its access is revoked, the old row remains but a fresh open-time check removes the Project name, file details, Team names, and link. It shows only safe unavailable wording. A later regrant creates a new event and does not restore details in the old row.

Project sharing grants access only to that Project. The notification and email report that grant; they do not extend access to other Projects, connectors, document sets, or agents.

28.2 Create an agent

Open **Agents** from the workspace navigation and click **New Agent**, or navigate to `/app/agents`. Workspace admins can also create and manage agents from `/app/admin/agents`. The editor opens with fields for identity, instructions, knowledge, and tools.

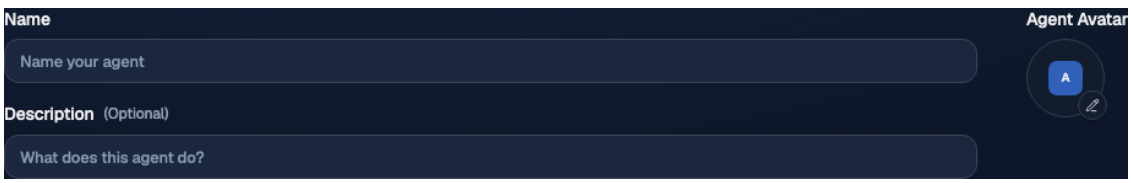

Figure 28.2: Agent editor header showing the Create Agent title, avatar picker, Name field, and Description field.

Fill in the identity fields first:

1. **Name.** Short and verb-led when possible, for example `HR Policy Assistant` or `Finance Helper`. The name is what end users see in the agent picker.

2. **Description** (optional). One sentence on what this agent is for. Users see the description in the agent popover, so write it for them, not for yourself.
3. **Agent Avatar.** Pick an icon. This is the visual the picker shows, so make it distinguishable from your other agents.

Keep each agent focused on one job. If you find yourself describing two unrelated workflows, split it into two agents. Multiple small agents almost always beat one large agent.

28.3 Instructions

The **Instructions** field is the agent's system prompt. This is where you tell the agent what role it plays, what it should and should not do, and how it should format answers.

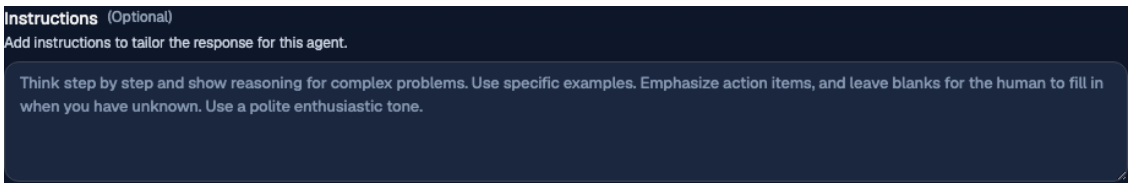

Figure 28.3: Instructions field showing placeholder text describing the expected format for agent instructions.

Use this rough structure when you write instructions:

1. **Role.** One sentence on who the agent is. "You are the HR assistant for Demo Workspace employees."
2. **Scope.** What it should answer. "Answer questions about leave policy, payroll dates, and onboarding paperwork."
3. **Limits.** What it should refuse or escalate. "If the question is about a specific person's contract or compensation, ask the user to contact HR directly."
4. **Format.** How long, how cited, what tone. "Answer in bullet points. Always cite the source document when you quote policy."

Keep instructions under a page. Long prompts dilute the instructions the model actually follows, and they slow every reply. If you need lots of conditional behavior, split it into multiple agents instead.

> Tip
>
> Test the instructions before you ship the agent. Open a chat, ask three real questions from real users, and read the answers carefully. Tweak the prompt until the answers match what you would expect a human in that role to say.

28.4 Conversation Starters

Conversation Starters are the suggested prompts users see when they open a new chat with the agent. They are pure UX scaffolding, but they make a big difference for first-time users who do not yet know what to ask.

There are two sides to this feature, and they live in different places:

- **What users see** is the read-only set of starter chips on the chat home screen. Clicking a chip sends that prompt on the user's behalf.
- **Where starters are edited** is the **Conversation Starters** field inside the agent editor. An agent owner can edit their agent, and workspace admins can manage workspace agents from the admin view.

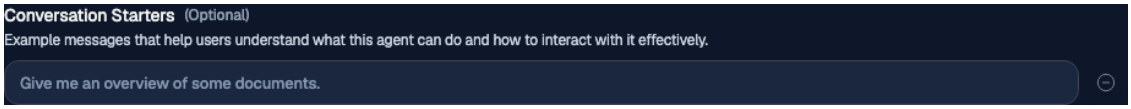

Figure 28.4: Conversation Starters field with placeholder text showing an example starter message.

Each starter has a **Name** (the label on the chip) and a **Message** (the prompt sent on the user's behalf when the chip is clicked). Write between three and five starters. Cover the most common things the agent is asked. Pull the wording from real questions in your team chat or ticketing system if you have access to them.

A good set for an HR Policy Assistant might be:

- "How do I check my annual leave balance and request time off?"
- "Summarize the probation period policy for new joiners."
- "Walk me through filing an expense reimbursement claim."

Refresh starters every few months. If nobody is clicking them, they are not the right questions.

28.5 Knowledge

The **Knowledge** section attaches one or more document sets to the agent. When **Use Knowledge** is enabled, the agent uses those document sets as its retrieval scope for every question, the same way Knowledge Search works when a user picks a document set manually.

Use this sequence to attach knowledge:

1. Toggle **Use Knowledge** on.
2. Open the document set picker.
3. Select one or more document sets you have already created and confirmed ready in Document sets.
4. Save.

A few rules to keep retrieval clean:

- **Attach the document set, not the connector.** Document sets are the reusable, permissioned scope. A connector is just where the bytes came from.
- **One scope per agent when you can.** If you attach three unrelated document sets, the agent will mix them. Two agents with one set each often produce sharper answers than one agent with two sets.
- **Confirm the set is ready first.** A document set that is still syncing will return inconsistent results. Validate it in Knowledge Search before you wire it into an agent.

If your agent does not need any retrieval (for example, a pure tone-rewriter or a translator), leave **Use Knowledge** off. The agent will rely entirely on its instructions and any tools you attach.

28.6 Actions and tools

The **Actions** section is how the agent calls outside services or runs tools. Registered MCP tools and OpenAPI actions appear here as toggles alongside built-in capabilities such as image generation.

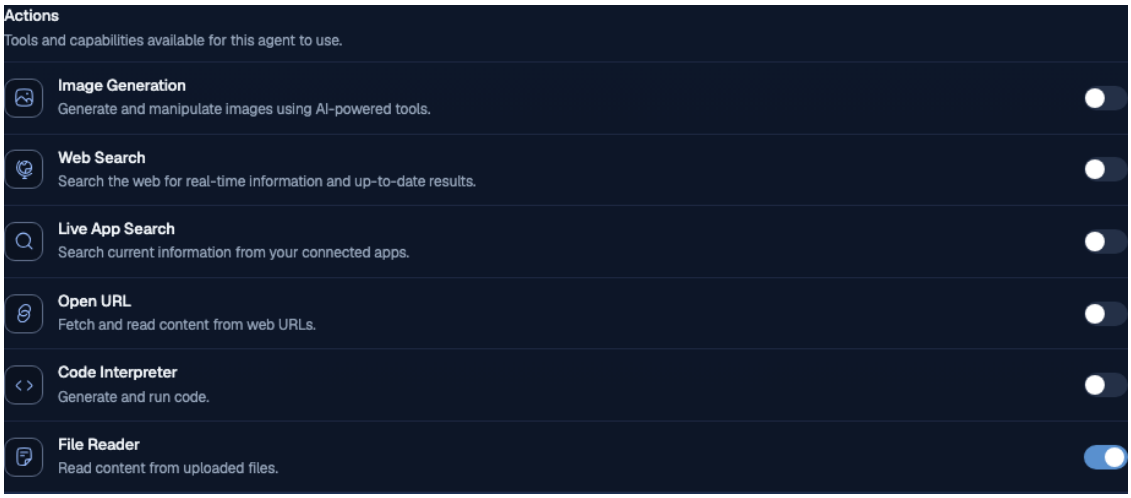

Figure 28.5: Actions section showing the built-in Image Generation, Web Search, Open URL, Code Interpreter, File Reader, and Live App Search tool toggles.

Things you can enable here:

1. **MCP tools.** Tools exposed by MCP servers you have registered in **Admin > Actions > MCP** and that you are permitted to access. Each registered server contributes one or more tools. A workspace-created agent keeps its explicit per-tool selection across chats, and the model decides whether and when to call an attached tool.
2. **OpenAPI actions.** Endpoints from OpenAPI specs you have registered in **Admin > Actions > OpenAPI**. Useful for hitting your internal REST APIs from inside a conversation.
3. **Image Generation.** A per-agent toggle that lets the agent generate and manipulate images using AI-powered tools.
4. **File Reader.** A per-agent toggle that lets the agent read the content of files uploaded to the chat or attached to the agent. It is on by default for new agents.
5. **Live App Search.** Available when you have connected a supported personal app: Sophea Meet, Gmail, Microsoft Outlook, or IMAP. It queries that app as the current user when the question is asked. Mail answers can search message sender, subject, body, and recent dates, while Sophea Meet can find recent meetings by name. Personal mail is never added to Knowledge Search; indexed meeting transcripts continue to use Knowledge Search.

For the full setup of MCP servers and OpenAPI actions (registration, auth modes, tool discovery, testing a spec), see Actions. This page only covers wiring an already-registered action onto an agent.

Attach only the tools the agent needs. Each enabled tool widens the agent's surface area and adds to the latency budget of every reply. A focused agent with two tools usually beats a permissive agent with ten.

The default workspace assistant behaves differently: newly discovered MCP tools attach to it automatically, and each user can select an accessible MCP app in the chat composer to make its enabled actions available throughout the current chat. The model may use a relevant allowed action or answer

without one. The selection remains active after completed, failed, and cancelled responses until the user turns it off or starts or switches chat context. This chat-scoped selection does not change the persistent tools attached to a workspace-created agent. Private, Team, and Workspace access on the MCP server still limits who can see and execute those tools.

28.7 Sharing and visibility

By default a workspace-created agent is private to its creator. Open the **Share** modal from the agent list or editor to change who can see and use it.

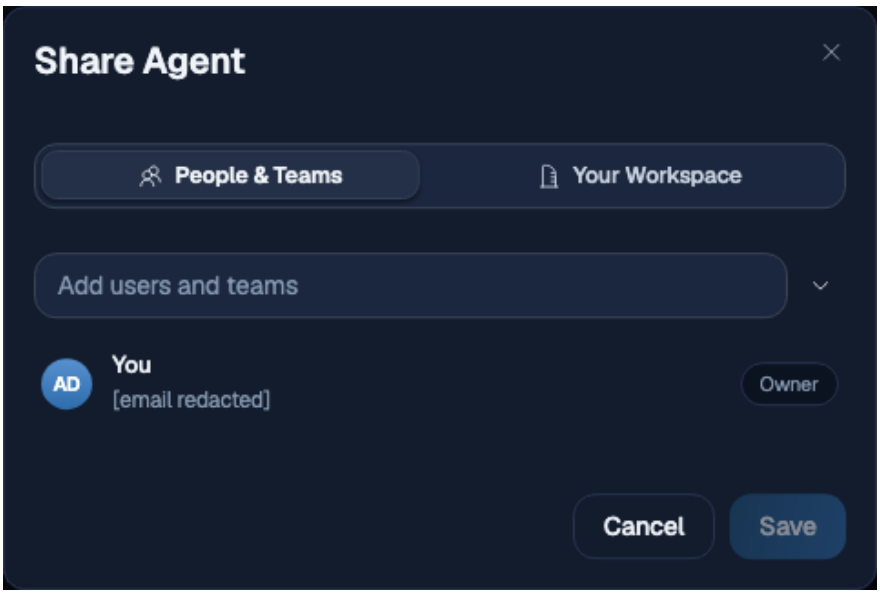

Figure 28.6: Share Agent modal with tabs for People and Teams and Your Workspace.

Use the sharing picker to add specific users or synchronized Portal Teams, or make the agent available to everyone in the workspace.

> Important
>
> Sharing an agent does not grant access to its attached document sets or connectors. Every knowledge lookup still uses the caller's user and Team principals. A user can open a shared agent but receive no restricted citations when they lack access to the underlying knowledge.

Sharing a private agent with a Team follows the same Team-share notification contract as Projects, Connectors, and Document Sets: each active Team member who newly gains access receives one in-app notification and one email, and the row links to the agent gallery. The actor, the agent's owner, workspace admins who already have access, and members who already had access are not recipients. Making an agent available to the whole workspace notifies nobody, because everyone already has access.

An agent's own attached files follow the share: a Team member who gains access to the agent can also cite the files attached to it. Attached document sets and connectors do not follow the share, as the callout above says, so verify the agent and its knowledge scope separately as a real user.

After changing visibility, test both the agent picker and one knowledge question as a representative user. This confirms agent access and document access independently.

28.8 Publish an agent

"Publishing" in the workspace sense means making the agent available to users from the chat composer. There are two things to verify after you save:

1. **The agent appears in the agent picker.** Open a new chat and click the agents control in the composer. Your agent should be in the list, with the icon and description you set.
2. **A real question works end to end.** Pick the agent, ask one concrete question that exercises both its instructions and its attached knowledge, and read the answer carefully. Confirm:
   - The tone matches your instructions.
   - Citations point at the document set you attached, not at unrelated content.
   - Any tools you enabled are called when the question demands them, and not called when it does not.

If something is off, go back to the editor. The most common fixes are tightening the instructions, narrowing the attached document set, or removing a tool the agent should not have had.

The same agent picker lists Sophea-published agents (when your Portal administrator has enabled them for your workspace) alongside your workspace-created agents. Users pick from the same control.

28.9 Chart and Presentation artifacts

When an enabled published agent creates a chart or presentation, Nous shows the result as an artifact card in the chat instead of placing the artifact JSON in the answer. Chart cards include a safe PNG preview and downloads for the JSON, SVG, and PNG files. Presentation cards include best-effort slide thumbnails and downloads for the PowerPoint and PDF files when those formats are available.

Artifact links are authenticated to the current workspace and remain available for 30 days. If a link has expired, ask Sophea to generate the chart or presentation again. SVG files are downloaded as attachments; they are not displayed as executable markup in the chat.

Chart and **Presentation** are published by default for workspaces that inherit the platform agent policy. The assistant automatically uses them for clear chart or deck creation requests in English or Greek, but neither is mandatory and neither appears as a selected composer step. Explicitly selected tools keep precedence. Chart requires exact structured rows in one Markdown table or fenced JSON/CSV dataset and never invents values. Presentation can call Chart internally when complete rows are available. When a deck request explicitly asks for current research, Presentation can also call Web Search internally and only charts rows that are verified against cited source evidence. A Portal administrator can still disable either agent globally or for a workspace.

An immediate follow-up such as "create a presentation about it" or "use that chart in a deck" reuses the exact Chart result from the preceding turn in the same conversation branch. Presentation preserves the source rows and chart visual rather than asking the model to reconstruct them. If the preceding chart is missing, ambiguous, expired, altered, or belongs to another workspace, Sophea asks the user to create or provide the chart again and does not generate a deck from unverified data. If secure storage cannot distinguish a missing chart from a temporary access problem, Sophea instead asks the user to try again.

28.10 Built-in Deep Research mode

Nous also has a separate built-in **Deep Research** mode. A workspace admin controls it through **Settings** > **Chat Preferences**, not through Portal's published-agent selection. When enabled, it appears in the composer's **Tools** menu, runs alone, and deselects other active tools. It is not available in project chats.

Deep Research is a Nous chat mode controlled by Chat Preferences. It is independent of the published agents that your Portal administrator enables for the workspace.

A Deep Research run keeps going on the server when the user refreshes the page, navigates away, closes the tab, or loses the network. The same applies to an ordinary chat answer. Reopening the chat shows the answer as it is being written and follows it to the end, then shows the final result with its citations.

Only the **Stop** control cancels a run. A cancelled run stays cancelled and is still shown as cancelled after a refresh, so users can tell a deliberate stop apart from a connection that dropped.

Published Sophea agents are unaffected by this. They already run in the background after Nous accepts the handoff, and closing the tab never interrupted them.

> Note
>
> One limit is worth telling users about: an answer in progress does not survive a restart of the Nous service, which happens when the platform is updated. The affected message is marked as an interrupted generation that the user can retry, rather than being left to look like a finished answer.

For everything else (writing your own instructions, attaching your own document sets, wiring your own actions), stay in the workspace-created agent flow that the rest of this page describes.

Chapter 29

Actions and tools

> **Note**
>
> Workspace admins can manage MCP and OpenAPI Actions when the Agents product is enabled for the workspace. If the Actions links are absent, check the workspace product settings in Portal rather than looking for a separate platform-admin role.

Actions let agents do more than read indexed documents. An action calls an external system on the user's behalf, then returns the result into the conversation. Use actions when an agent needs live data, needs to look something up by ID, or needs to trigger a workflow you already run elsewhere.

This chapter covers the two action surfaces workspace admins manage:

- **MCP Actions** at `/app/admin/actions/mcp`. Register a Model Context Protocol server, configure how Nous authenticates to it, and pull its catalog of tools.
- **OpenAPI Actions** at `/app/admin/actions/open-api`. Paste an OpenAPI spec, choose how each request is authenticated, and expose individual endpoints as tools.

The default assistant derives its MCP actions at request time from the current user's access. For a workspace-created agent, attach only the MCP or OpenAPI actions it needs in the agent editor. See Agents for that step.

29.1 Use apps and actions in chat

Attaching actions determines which actions the agent can use. For the default assistant, accessible MCP apps appear automatically in the chat composer's **Tools** menu. Selecting an MCP app there creates an optional allowlist of that app's enabled actions for the current chat. It is not a second access setting or a request that guarantees an action will run.

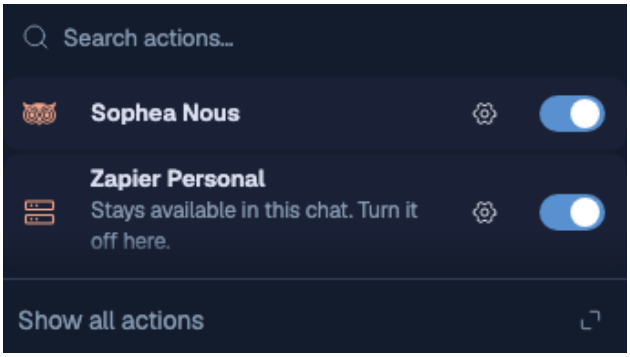

Figure 29.1: Active apps in the Actions flyout

- On the default assistant, **Sophea Nous** starts on. Web Search, Open URL, Code Interpreter, Image Generation, and File Reader are enabled initially. When the current user has connected Sophea Meet, Gmail, Microsoft Outlook, or IMAP, Live App Search is also available and enabled initially. Workspace-created agents offer exactly the built-in actions their creator switched on.
- Each accessible MCP app, such as Zapier, appears automatically when the server is connected and your access scope permits it. If the app still needs a personal connection, its row shows **Connect** and cannot be selected yet. There is no separate per-user enable step in Chat Preferences.
- Selecting an MCP app in the composer makes that app's enabled actions available throughout the current chat. You can keep up to two MCP apps active in a chat. The model decides whether an action is useful and which allowed action to call; selecting the app does not guarantee execution. The app stays selected after completed responses, technical failures, and cancelled responses. Turn it off in **Tools** when later messages no longer need it. Starting a New Chat or switching chats, assistants, or projects clears the selection.
- Refreshing the page clears MCP selections and resets Knowledge Search to its default enabled **All sources** state. It does not cancel an answer that is still being written: that answer keeps running and continues in the reopened chat. See Built-in Deep Research mode.
- The persistent **Sophea Nous** app behaves differently. Its enabled built-in actions stay available across messages, and the model chooses an action only when it is useful; its app switch does not force a call.
- Open **Configuration** for an app to choose its **Enabled actions**. For **Sophea Nous**, the app switch and its action switches are two views of one setting: the app reads as on while at least one built-in action is enabled, turning the app off disables every built-in action, and turning it back on enables them all again. MCP action switches control which actions may run, independently of the app's chat-scoped selection.
- Standalone OpenAPI actions use their own enabled-action switches because they do not belong to an MCP app.

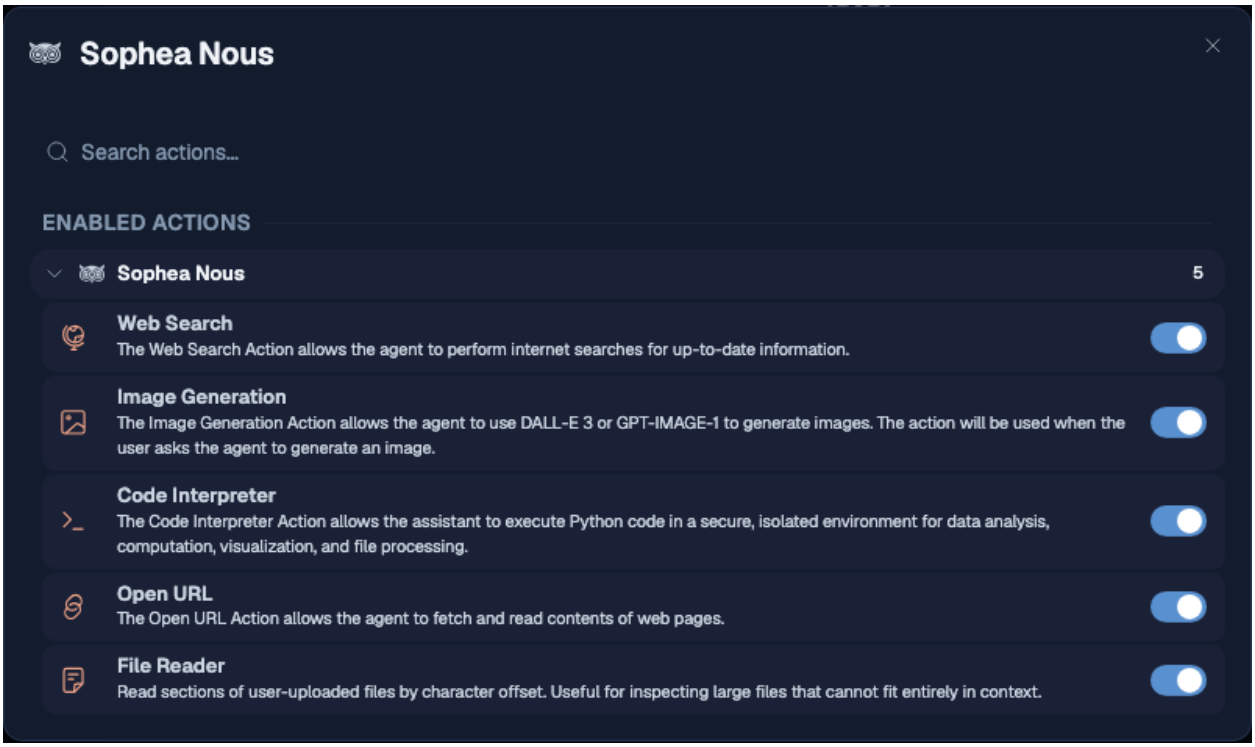

Figure 29.2: Sophea Nous app configuration after the app switch has enabled every action

Live App Search reads directly from the current user's personal connections. Use it for freshness questions such as meetings from today or recent mail from a sender, and for searches by meeting name, mail subject, sender, or body text. Gmail, Microsoft Outlook, and IMAP messages are never indexed in Knowledge Search. Nous never sends another member's personal credentials, and disconnecting the last supported personal app removes its live capability.

Zapier's provider actions are organised one level deeper. Open **Configuration** to browse a provider such as **Gmail** and choose actions individually, then select the Zapier app to make those enabled actions available in the current chat. The model may call a relevant allowed action or answer without Zapier. Every newly discovered Zapier action starts off. An individual **Find Email** switch does not enable **Send Email**, another Gmail action, or another provider.

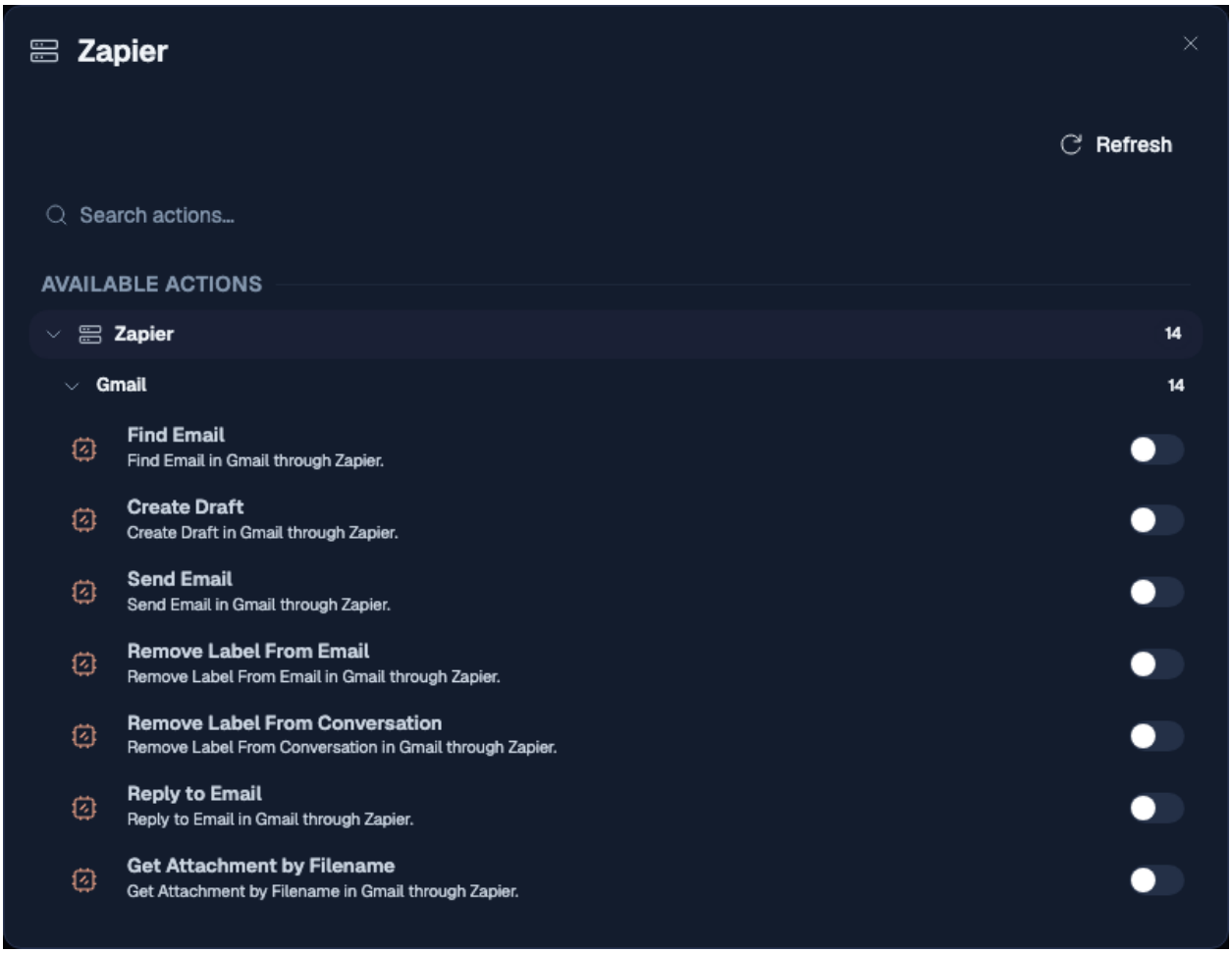

Figure 29.3: Zapier configuration with Gmail actions nested under the provider

Choose **Refresh** after changing the enabled actions in Zapier. The control stays available even when the current catalog contains zero actions. It stays busy and blocks duplicate clicks until the complete refresh finishes. Nous replaces the current user's catalog without changing another user's catalog. If Zapier's management contract is incompatible, Nous hides those actions instead of falling back to its generic management tools. A temporary connection failure or a catalog change during refresh keeps the last validated catalog and offers a retry.

Knowledge Search, Deep Research, and published external agents are composer controls rather than app gates. Knowledge Search starts enabled over **All sources**, and the model searches only when useful. Turning it off disables connected-knowledge retrieval for the chat. A narrowed scope turns Knowledge Search into the first ordered tool step; ambient **All sources** search is not an ordered step. MCP apps remain selected in the current chat until the user turns them off. Deep Research remains selected until the user turns it off and continues to run by itself. A published external agent clears after Nous accepts its background run and remains selected if submission fails before that point. A published external agent can coexist with an MCP app selection, but the MCP allowlist applies only to Nous and is not sent to the remote agent; an accepted handoff clears only the external agent. Knowledge Search can coexist with an MCP app selection: a message can search the knowledge base and then use an allowed action, and the reply covers both. Multi-Model Generation suppresses MCP

apps for the request without turning off an app that is already active. Starting a New Chat or switching chats, assistants, or projects clears other chat-scoped selections and resets Knowledge Search to enabled **All sources**, while keeping the saved app configuration.

Deep Research runs by itself. While it is selected, Nous temporarily hides enabled apps and Knowledge Search from the active-tools summary without changing saved app switches or the Knowledge Search scope. Those settings return afterward. If an allowed app or action is unavailable or does not fit the selected model, Nous removes it from the available tool set and can continue with the remaining valid work. Selecting an MCP app does not turn that warning, or a response that uses no MCP action, into a required-operation failure.

The composer permits at most eight ordered tool steps. Ambient Knowledge Search does not count toward this limit. A narrowed Knowledge Search scope counts once and always runs before other selected steps.

An MCP app being **visible**, **ready**, and **selected** are separate states. Visible means the server is connected and its access scope permits the user. Ready means any required personal authentication is complete. Selected means the user has made that app's enabled actions available for the current chat; the model can still answer without calling one. Removing access, disconnecting the server, revoking personal credentials, or leaving the app with no enabled actions removes an existing selection before a later message.

29.1.1 Connect personal MCP credentials

An MCP server can be available to the workspace while still requiring each user to connect a personal account. The app row reports these states separately. A connected server that still needs your credentials shows **Connect**, and its app and action switches remain unavailable until your connection succeeds.

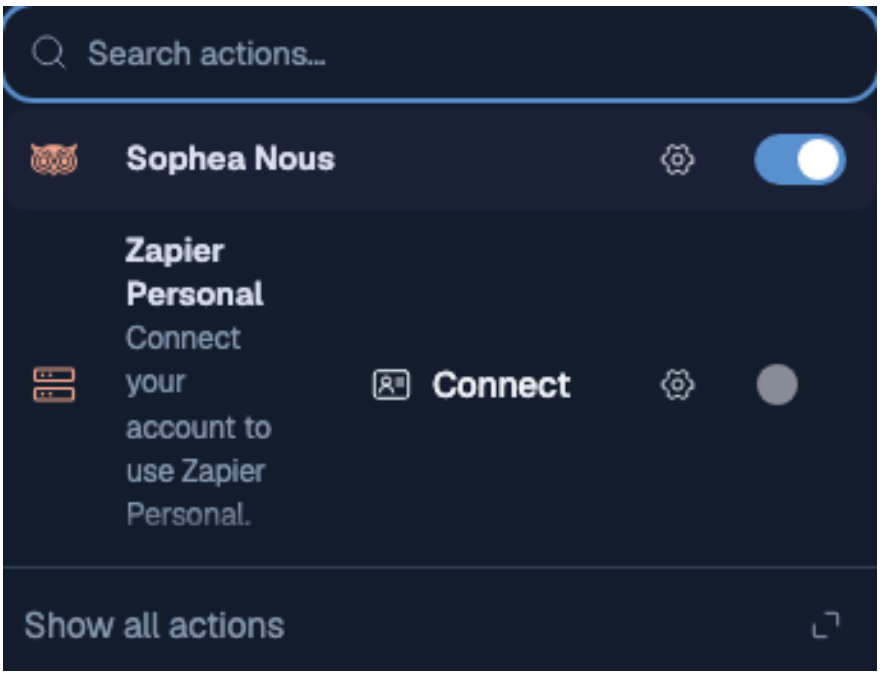

Figure 29.4: Zapier app requiring a personal MCP connection

- For an API-token connection, choose **Connect**, enter every requested credential, and submit the form. Nous tests the credentials against the live MCP server before saving them. A failed test leaves the previous connection unchanged.
- For OAuth, choose **Connect** and complete the provider's authorization page. Nous returns you to the same chat after it verifies the new connection.
- Until your personal connection is complete, Nous excludes the app's actions from the tools sent to the model. A stale selection cannot make Nous use the workspace creator's credential.
- For a supported Zapier connection, a successful connection also discovers that user's enabled provider actions. Use **Refresh** from the app configuration when the enabled set changes in Zapier.

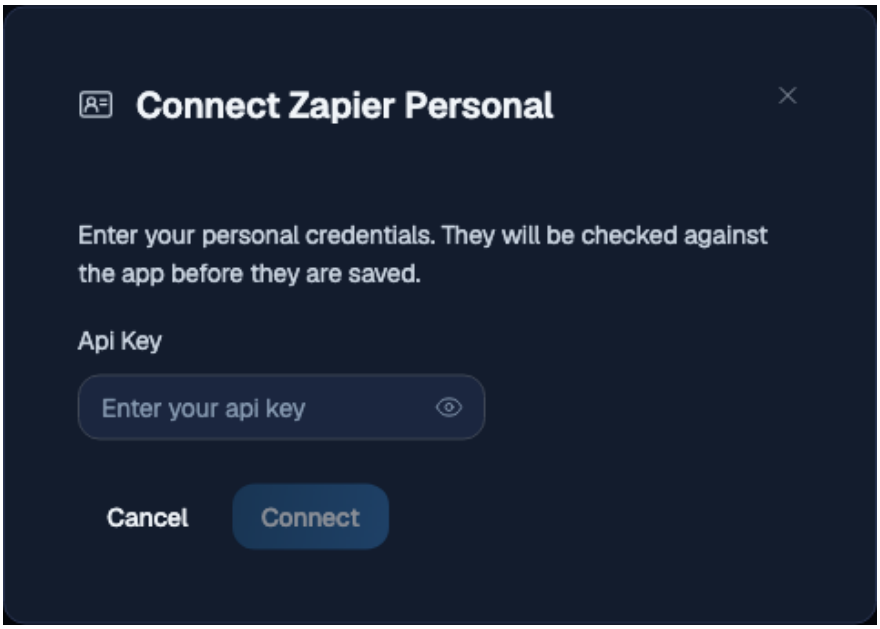

Figure 29.5: Personal MCP credential dialog

29.2 What actions are

An action is a named callable that an agent can invoke during a chat. Each call is a single, scoped request: the agent picks the action, fills in the arguments from the conversation, the request runs against your configured endpoint, and the response is fed back into the model so it can write a grounded answer.

Two action types are available:

- **MCP tools.** You register one MCP server. Nous discovers the tools it exposes, and each tool becomes an attachable action.
- **OpenAPI tools.** You upload one OpenAPI document. Each operation in the spec becomes an attachable action.

A third capability, **Code Interpreter**, is not on these pages. It is a per-agent toggle in the agent editor that lets the agent run sandboxed Python during a chat. Manage it on the agent itself, not here.

> **Tip**
>
> Choose MCP for a tool catalog that evolves on the host side: the MCP server owns the list of tools and their schemas, and Nous picks up changes on the next refresh. Choose OpenAPI for a stable HTTP API you already publish: you have one document, the endpoints rarely change, and you want explicit per-endpoint control over auth and headers.

If a vendor offers both an MCP host and an OpenAPI spec, prefer MCP for richer tool descriptions and built-in argument schemas. Use OpenAPI when you only have a spec and no MCP wrapper.

29.2.1 Zapier provider actions and controller tools

Zapier's MCP server publishes a small catalog of controller tools while managing a larger set of enabled actions for services such as Gmail, HubSpot, and ClickUp. A supported personal Zapier connection translates those enabled provider actions into individually selectable Nous actions.

This distinction affects both administration and chat:

- For an **Individual Key (Per User)** Zapier connection, the MCP Actions page and the app's **Enabled actions** configuration show provider groups and their individual actions. Zapier's controller tools stay hidden and are not sent to the model as alternative actions.
- Every user connects their own Zapier credentials and receives an independent catalog. Newly discovered actions start off until that user enables them.
- A **Shared Key (Admin)** connection remains a standard MCP connection and uses the protocol tools published by its server. Nous does not automatically convert a shared connection or its catalog into a personal provider-action connection.
- A follow-up question can use results already present in the conversation. Select Zapier again only when the follow-up should make another live Zapier action available.

If an expected provider action is missing, enable it in Zapier and choose **Refresh** in the app configuration. A zero-action catalog still offers **Refresh**. After changing an existing server from a shared key to an individual key, any user without a ready personal connection must choose **Connect** and supply personal credentials before Nous can discover or execute that user's provider actions.

29.3 MCP Actions

The Model Context Protocol is a standard for exposing a catalog of tools over HTTP. An MCP server publishes a list of tools, their input schemas, and their human-readable descriptions. Nous fetches that catalog, persists it, and lets you attach individual tools to agents.

Open `/app/admin/actions/mcp` to see every MCP server registered in this workspace. An empty workspace shows a prompt to connect your first server.

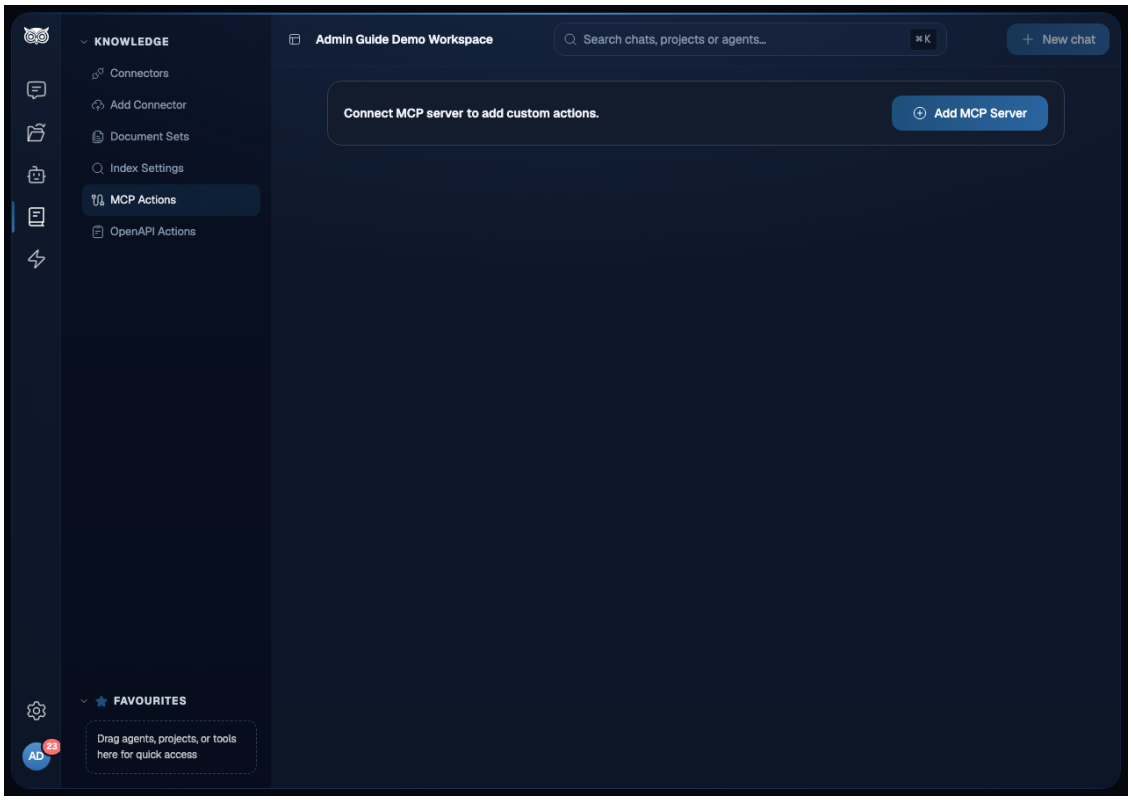

Figure 29.6: MCP Actions list page with Add MCP Server button

The **Knowledge** sidebar shows **MCP Actions** and **OpenAPI Actions** alongside connector and document administration. Navigate between the action pages from that sidebar. **Agents** remains a separate workspace section.

Open a server card to review the same actions that the current user can configure in chat. When an MCP catalog supplies provider metadata, the page groups actions under providers; tools without provider metadata remain in one flat list. A supported Zapier card therefore shows groups such as **Gmail**, **ClickUp**, and **Microsoft Outlook**, while its internal management tools stay hidden. Every provider group starts collapsed. Expand only the provider you need; a search temporarily opens matching groups, and clearing the search restores your previous expansion choices.

Each connected card has an **All tools** switch. Turn it off to disable the entire app's current tool list in one operation, or turn it on to enable the full list. Individual tool switches remain available after you unfold the card. Use **View tools** and **Fold** to expand or collapse each app independently. Settings controls whether a tool is available to run; the app and action switches in chat separately control whether the selected assistant may choose it. For a personal Zapier connection, these Settings controls affect only the current user's discovered actions.

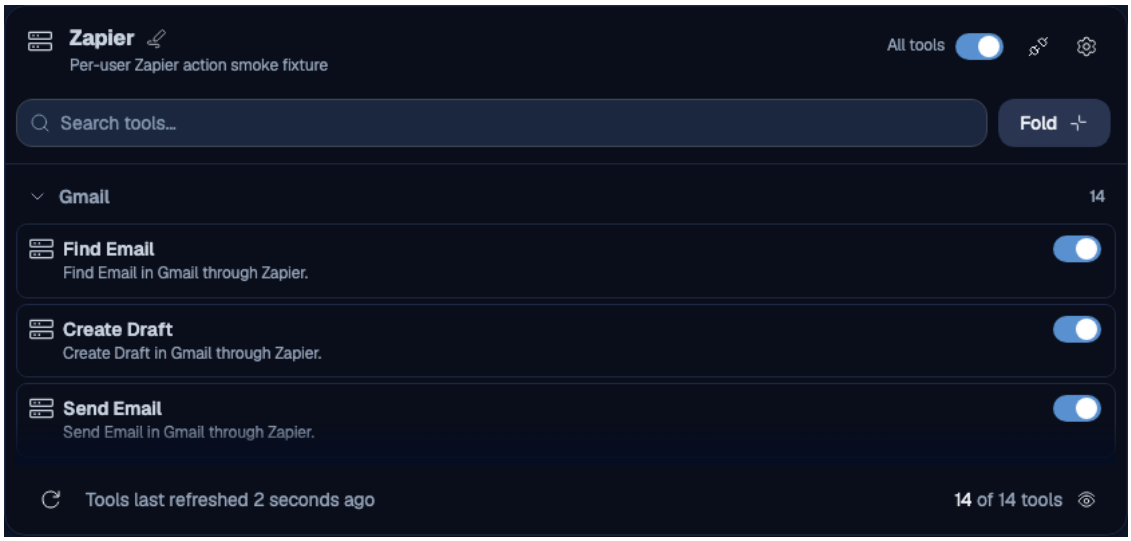

Figure 29.7: Zapier MCP card with current-user actions grouped by provider

29.3.1 Register a new MCP server

Click **Add MCP Server**. The modal opens with connection and access fields.

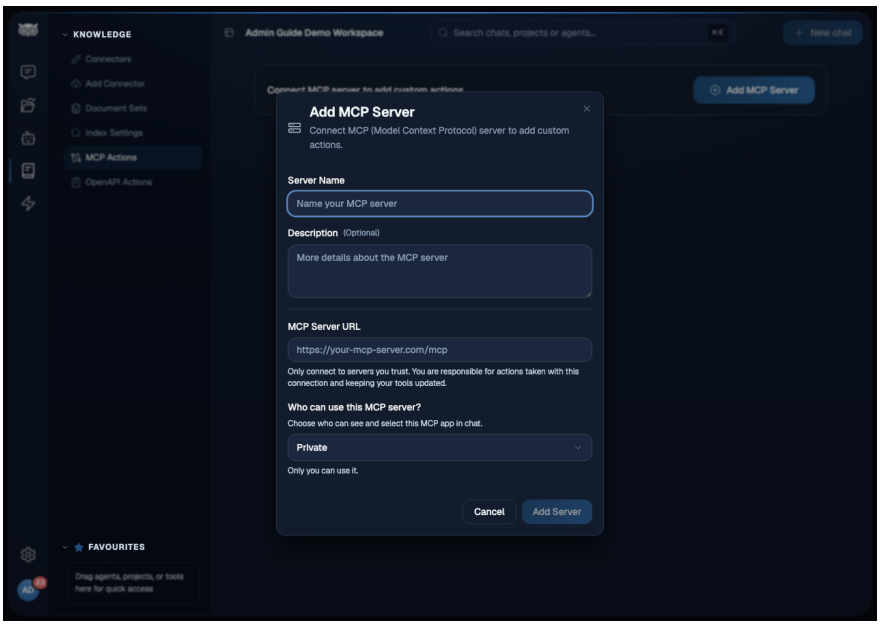

Figure 29.8: Add MCP Server modal with connection and access fields

Fill in the fields:

1. **Server Name.** A short label your admins will recognise. This is what shows up in the picker when attaching the action to an agent.

2. **Description (Optional).** A sentence explaining what the host does. Other admins will see this when they pick which tools to attach.
3. **MCP Server URL.** The endpoint of the MCP host. Use `https://` for anything that leaves your network. Only connect to servers you trust.
4. **Who can use this MCP server?** Choose **Private**, **Teams**, or **Workspace**. Private is the default. Teams requires at least one active Portal Team; Workspace makes the app available to every active member of this workspace.

You can change the access scope later from **Manage MCP Server**. Access controls whether a user can see, connect to, and execute the MCP app. It does not grant membership in the upstream service. Connections created before access scopes were introduced remain available to the Workspace for compatibility; review those connections and narrow their scope where appropriate.

Click **Add Server**. The entry is created without any authentication yet. Configure authentication separately from the server's detail page before Nous can fetch the tool catalog.

29.3.2 Authentication modes

Open the server's row and click **Authenticate**. The authentication modal exposes four modes:

- **None.** No credentials are sent. Use only for fully public, read-only MCP hosts you trust.
- **OAuth.** Each user authenticates against the MCP host with their own OAuth credentials. Optionally paste an OAuth `Client ID` and `Client Secret` if the host does not support Dynamic Client Registration (DCR). The modal shows the redirect URI to register with the host.
- **OAuth Pass-through.** Nous forwards the user's existing access token to the MCP host as an `Authorization` header. Use this only when the host trusts the same identity provider as Nous.
- **API Key.** Pick `Individual Key (Per User)` so each user supplies their own key, or `Shared Key (Admin)` to enter one organisation-wide key. The shared key is sent on every request as a bearer token.

Pick the mode the host supports. Mixing modes does not work: the server accepts OAuth, the pass-through token, the configured API key, or nothing.

> Warning
>
> Tool credentials are workspace secrets. Anyone with admin access can read which tools are attached and trigger them indirectly through any agent that uses them. Rotate keys on the same cadence as your other workspace secrets, and revoke an entry the moment a vendor relationship ends. Use the host's own dashboard to invalidate the issued credential after you remove the entry from Nous.

29.3.3 Discover and review tools

After the server is saved, open its row to see the discovered tools. Each tool has a name (used by the model when calling it), a description (used by the model when deciding whether to call it), and an input schema (the arguments the tool accepts).

Nous derives the default assistant's MCP actions at request time from the current user's access. They appear as one available MCP app in the composer's **Tools** menu only for users allowed by the server's access scope. The app remains unselected until the user chooses to make its enabled actions available for the current chat.

Review every tool before you allow agents to use it. The descriptions the model reads come directly from what the MCP server publishes. If a description is vague, the model will call the tool at the wrong moments. Edit the description on the host side and then refresh.

To refresh the catalog after the host adds or changes tools, click **Refresh tools**. A supported action catalog uses **Refresh actions** instead. The refresh control stays busy until discovery and synchronization finish, then Nous shows the updated list without a page reload.

29.3.4 Remove an MCP server

Open the server's row and click **Delete**. Any agent that referenced one of its tools loses that capability immediately. The agent itself continues to work; the action call disappears from its tool list on the next chat.

> Warning
>
> Deletion is immediate and not reversible. Every tool from this MCP entry is removed from every agent that referenced it the moment you confirm. To restore the tools you must recreate the MCP server entry from scratch, reconfigure authentication, and re-attach each tool to the agents that need it. Prefer disabling authentication or detaching the tools from individual agents first when you only need to pause access temporarily.

29.4 OpenAPI Actions

An OpenAPI action wraps a standard OpenAPI (Swagger) document. Nous parses the spec, lets you pick which operations to expose, and turns each one into an attachable tool. The agent fills in the path, query, and body arguments from the conversation, then Nous sends the request with the auth configuration you set.

Open `/app/admin/actions/open-api` to see every OpenAPI action registered in this workspace.

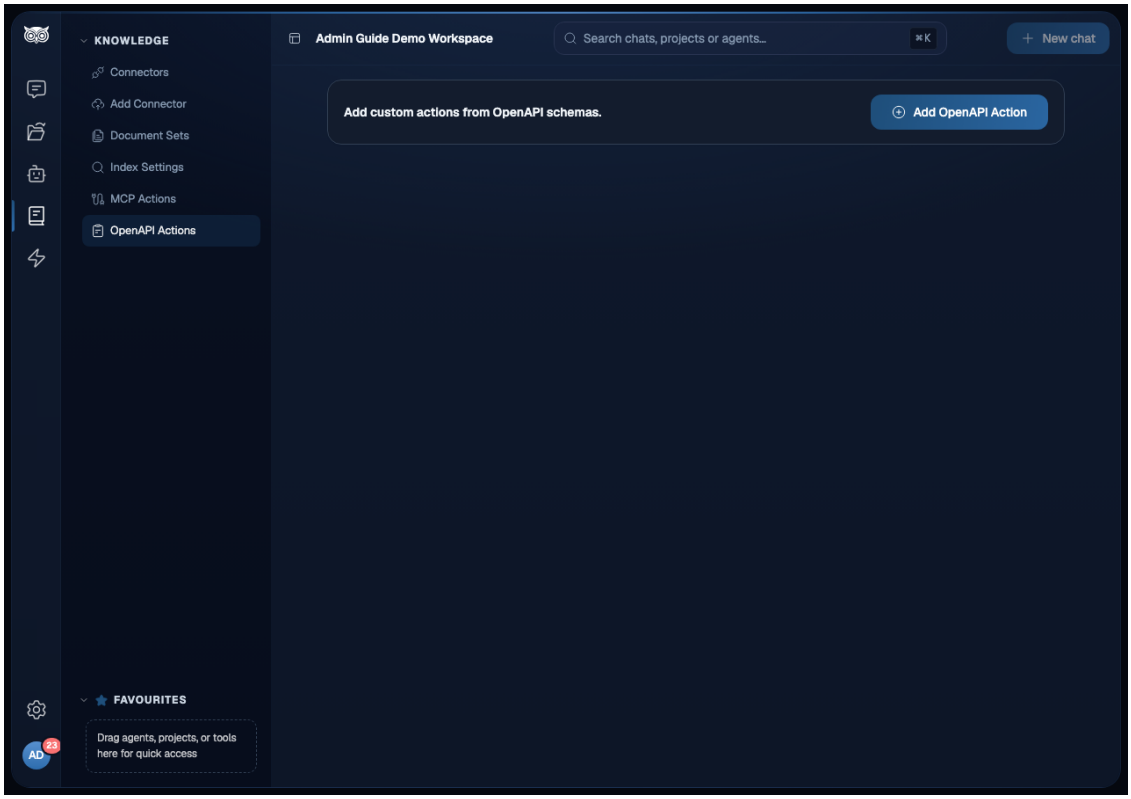

Figure 29.9: OpenAPI Actions list page with Add OpenAPI Action button

29.4.1 Register a new OpenAPI action

Click **Add OpenAPI Action**. The modal opens with a single input area.

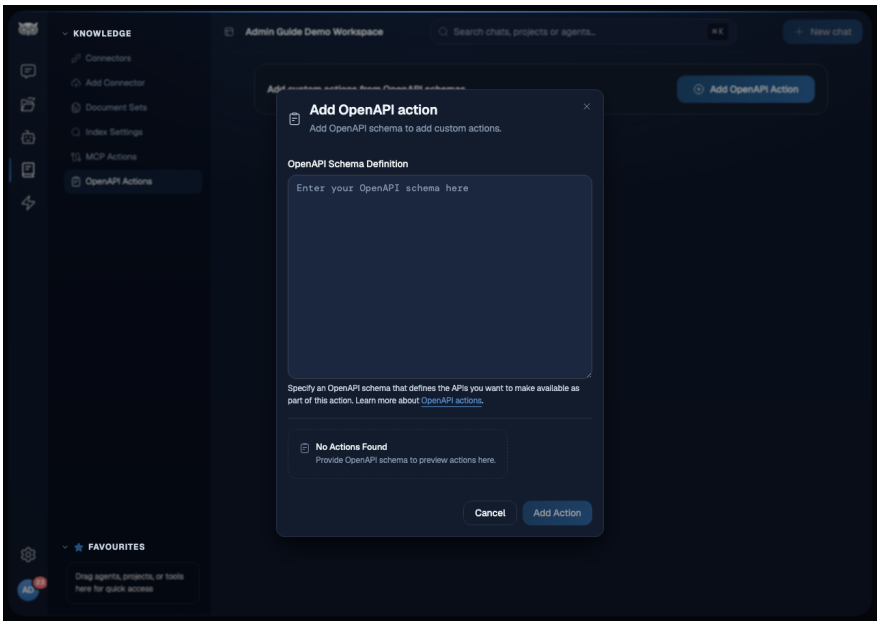

Figure 29.10: Add OpenAPI action modal with schema definition textarea and No Actions Found preview

The modal has one field:

1. **OpenAPI Schema Definition.** Paste a JSON OpenAPI 3.x document. Nous validates it as you type. The action name, description, and base URL are read from the spec: `info.title` becomes the action name, `info.description` becomes the description, and the first entry in `servers[].url` becomes the base URL Nous calls. There is no separate name or URL field. Edit your spec so `info.title` is descriptive before pasting.

Once the schema validates, the preview area below the textarea shows each detected operation as a tool. If the schema is invalid or empty, the preview shows **No Actions Found**. Click **Add Action**. Each operation in the spec is then attachable as an individual tool. The entry is created without any authentication. Configure authentication from the action's detail page.

29.4.2 Authentication modes

Open the action's row and click **Authenticate**. The authentication modal exposes three modes:

- **OAuth.** Nous performs an OAuth flow per user. Provide the `authorization URL`, `token URL`, `client ID`, `client secret`, and any required scopes. Use this when the upstream API supports OAuth and you want per-user access tokens without the user managing the secret themselves.
- **Custom Authorization Header.** Enter a fixed set of headers (typically `Authorization: Bearer ...` or a vendor-specific header) and Nous adds them to every call. Use this for service-to-service APIs that issue one long-lived key.
- **OAuth Pass-through.** Nous forwards the end user's existing access token to the upstream API. Use this when the API has its own per-user identity model under the same identity provider and you do not want Nous to hold a shared key.

Pick the mode that matches how the API expects to be called. Mixing modes inside one OpenAPI action is not supported. If you need two auth styles for the same vendor, register two separate OpenAPI

actions.

> **Warning**
>
> Custom-header values and OAuth client secrets enter the workspace as plain secrets. Treat them like any other production credential: rotate on schedule, restrict who has admin access, and remove the action the moment the credential is rotated upstream.

29.4.3 Remove an OpenAPI action

Open the action and click **Delete**. As with MCP servers, any agent that referenced one of its operations loses access on the next chat. The agent's configuration is not destroyed; only the tool call is removed.

> **Warning**
>
> Deletion is immediate and not reversible. Every operation from this OpenAPI action is removed from every agent that referenced it the moment you confirm. To restore the tools you must recreate the OpenAPI action by pasting the schema again, reconfigure authentication, and re-attach each operation to the agents that need it. Prefer detaching the operations from individual agents first when you only need to pause access temporarily.

29.5 Attach an action to an agent

At request time, the default assistant receives only the MCP actions accessible to the current user. Attachment in the agent editor is required for workspace-created agents and for OpenAPI actions.

To attach a registered action:

1. Open Agents and open the agent you want to extend, or create a new one.
2. Scroll to the **Actions** section in the editor.
3. Toggle on the accessible MCP tools or OpenAPI operations you want this agent to use.
4. Tighten the agent's instructions so the model knows when to use each tool. A clear instruction beats a clever description.
5. Save the agent.

The next time a user starts a chat with this agent, the tools are available to the model. The model decides when to call them; you cannot force a call from the editor.

> **Note**
>
> **Code Interpreter** is a separate per-agent toggle in the agent editor. It does not appear on the action admin pages because it has no external endpoint to register. Treat it the same way you treat any MCP or OpenAPI tool: enable it only on agents that need it, and explain in the agent instructions when the agent should reach for it.

After attaching, validate the agent. Open a new chat, ask a concrete question that should trigger the tool, and confirm the agent calls it with the right arguments. If the model never calls the tool, the description or the agent's instructions are the first place to fix.

Chapter 30

Search Verification and Troubleshooting

Two inspection tools let you verify what was actually indexed and which documents users find useful: the **Document Explorer** and the **Document Feedback** page. Both are reachable by workspace admins from the **Knowledge** area and require no platform-admin access.

30.1 Document Explorer

The **Document Explorer** at `/app/admin/documents/explorer` lets you search the indexed corpus directly, without going through a chat. Use it to confirm that a specific document was indexed and to see how it is filtered by source and time. Any workspace admin can open it from the **Knowledge** area.

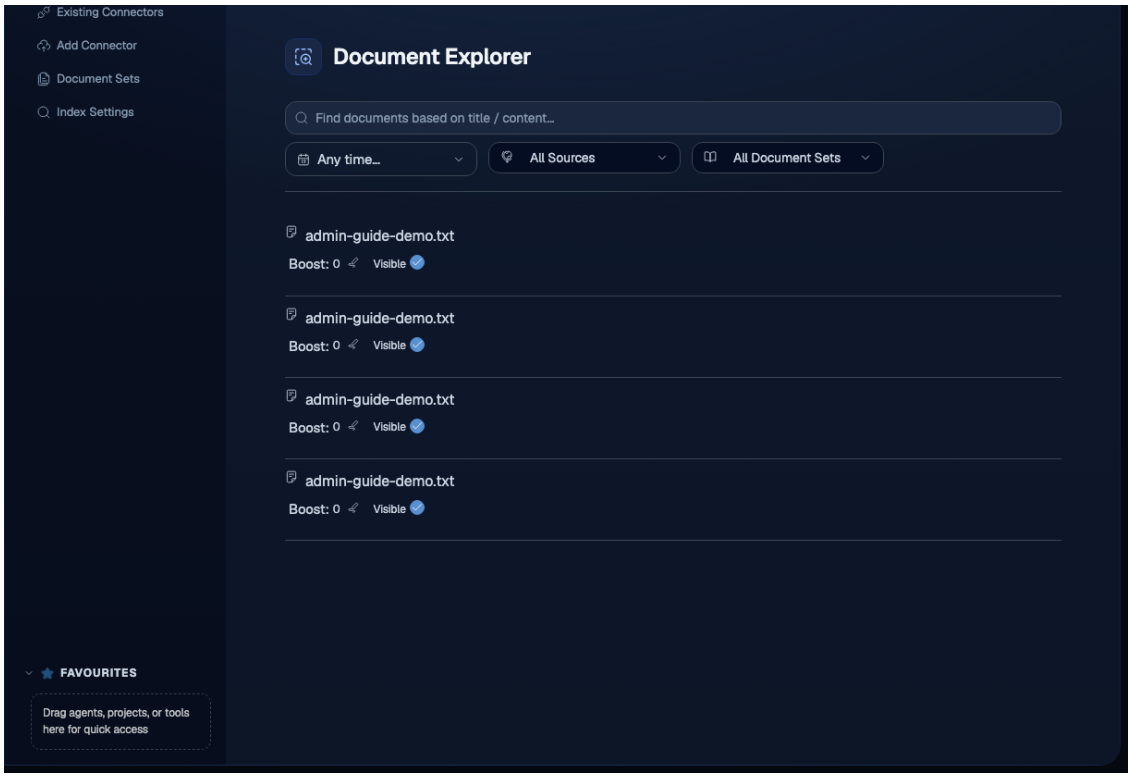

Figure 30.1: Document Explorer page with a search field, an Any time filter, an All Sources filter, and an All Document Sets filter above an empty results area

To inspect the index:

1. Open the **Knowledge** flyout and select **Document Explorer**, or navigate to `/app/admin/documents/explorer`.
2. Type a title or content fragment into the **Find documents based on title / content** field.
3. Narrow the results with the filters above the field:
   - **Any time** scopes results by indexing or document date.
   - **All Sources** scopes results to one or more connector sources.
   - **All Document Sets** scopes results to a specific document set.

Each result row shows the document and whether it is currently searchable. If an expected document is missing here, it was never indexed: check the connector on the **Existing Connectors** page.

> **Tip**
>
> The Document Explorer is the fastest way to answer “is this document even in the index?” Reach for it before assuming a search complaint is a retrieval problem, because no retrieval behavior can surface a document that was never indexed.

30.2 Document Feedback

The **Document Feedback** page at `/app/admin/documents/feedback` ranks documents by the thumbs-up and thumbs-down votes users give to answers that cite them. It is a workspace-admin view of which sources are helping and which are hurting answer quality.

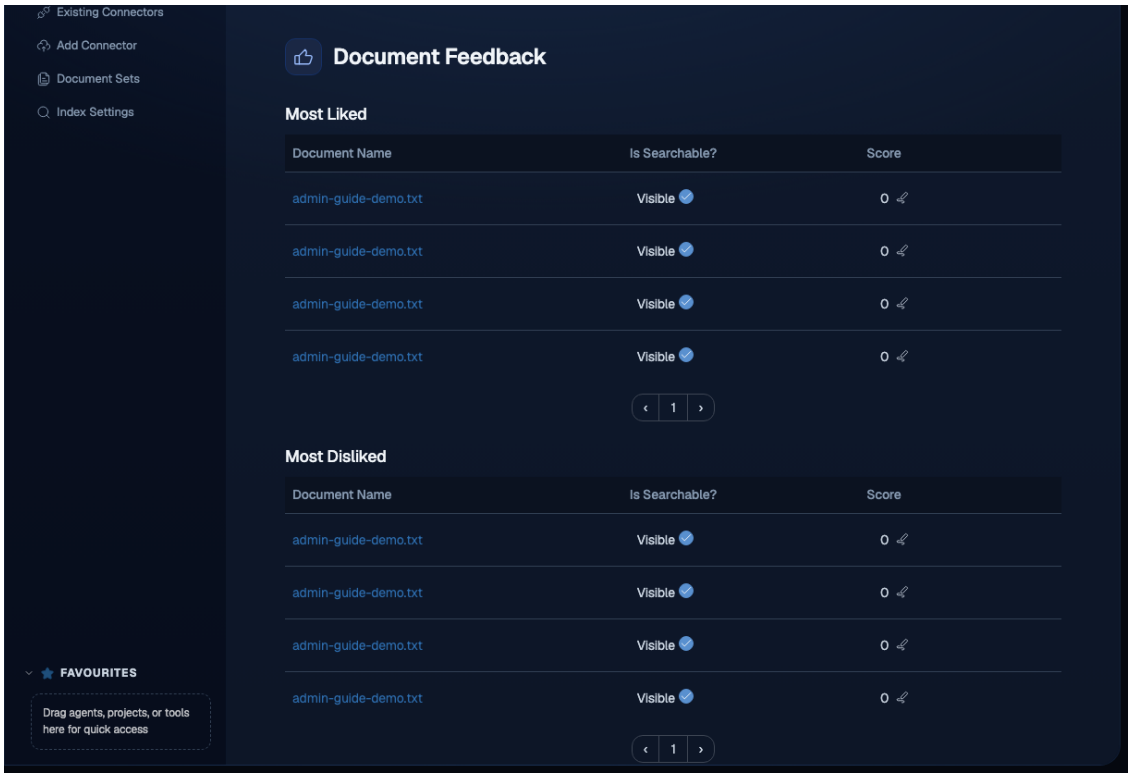

Figure 30.2: Document Feedback page showing a Most Liked table and the start of a Most Disliked table, each with Document Name, Is Searchable?, and Score columns

The page has two tables:

- **Most Liked**: documents whose citations received the most positive feedback.
- **Most Disliked**: documents whose citations received the most negative feedback.

Both tables share the same columns:

- **Document Name**: the indexed document, linked to its source.
- **Is Searchable?**: whether the document is currently retrievable (**Visible**) or has been hidden from search.
- **Score**: the net feedback score for that document.

Use the **Is Searchable?** control to hide a document that consistently produces poor answers without deleting it from the connector. A hidden document stays indexed but is excluded from retrieval, so you can reverse the decision later.

> **Note**
>
> Feedback scores accumulate only as users vote on answers in chat. A new workspace will show empty or low-signal tables until enough usage builds up. Treat these tables as a long-running quality signal, not a one-time check.

30.3 Portal: the Documents tab

Organization owners and admins reach equivalent functionality without leaving Portal. Open the workspace's **Connectors** page and select the **Documents** tab. The tab needs the Connectors product switched on for the workspace; if it is not, the tab explains that instead of showing content.

The tab holds two cards: **Indexed documents**, which searches and curates, and **Document feedback**, a read-only report.

30.3.1 Searching indexed documents

Type a query into the search box and press **Search**, or press Enter. Results are the closest matches to the words you typed, up to 50 at a time. Each result row is one document, showing the connector it came from, when it was last updated, and its current boost and visibility.

A document is often reachable through more than one connector, for example a file that is both a mailed attachment and stored in a shared drive. The row lists every connector the document came from, not only the first one: it shows one inline and a **+N more** control that expands the row to list the rest.

Search intentionally returns hidden documents alongside visible ones. That is how an admin finds a document they hid earlier, in order to unhide it.

30.3.2 Boost

Boost… on a row opens a dialog that sets how strongly the document ranks in the workspace's ordinary search results, the kind people see day to day when they search or ask a question. A positive boost makes the document easier to find; a negative boost makes it harder to find; 0 is normal. Boost does not change the order of results in this tab's own search, which always ranks purely by relevance.

A document can already carry a nonzero boost that no admin set. People in the workspace endorse or reject documents from search results, and that accumulated feedback moves the stored boost value up or down on its own. Finding a boost you did not set is expected, not a sign of a stray setting.

30.3.3 Hide and unhide

Hide… removes a document from the workspace's search results. The confirmation states the consequence before you confirm: nobody in the workspace will find the document in search. Hiding does not delete anything and is reversible at any time with **Unhide…**.

A hidden document still turns up when you search for it from this tab, marked with a **Hidden from search** badge. That is deliberate: it is the only way to find a hidden document again in order to unhide it.

30.3.4 The feedback report

Document feedback lists documents that have received at least one endorsement or rejection from search results, most feedback first, with an endorsed count and a rejected count for each.

These counts are counts of the feedback that currently stands, not counts of people. Sophea keeps one endorsement or rejection per document per search result: rating the same result the same way again changes nothing, and rating it the other way replaces the earlier answer instead of adding to it. So one person can add to a document's counts several times over different searches, but cannot raise a count by rating the same result twice. A high count is still not proof that many different people responded, and a count of zero does not mean nobody has access to the document or nobody has seen it: it only means nobody has endorsed or rejected it in search yet.

There are no comments, no submitter names, and no timestamps on this report. That is deliberate: the underlying feedback data does not carry them. The report itself is read-only; to act on a document that keeps being rejected, search for it under **Indexed documents** above and boost or hide it from there.

The report carries no total count, so it pages with **Previous** and **Next** instead of a page count.

30.3.5 A current limitation on some connectors

Boost and hide are not available yet for a document whose identifier contains a slash. Today that affects documents indexed from Jira, Drupal Wiki, GitLab, and the Web connector, because those connectors build a document's identifier from a URL or a path rather than a plain id, and Sophea's backend cannot currently route a boost or hide request for an identifier shaped that way to the right document. Google Drive, SharePoint, Confluence, and Gmail documents are unaffected.

This is a real, temporary limitation, not a design choice. The **Boost…** and **Hide…** controls on an affected row are disabled and explain why. Search is unaffected: you can still find and inspect these documents in the Documents tab and in the feedback report; boost and hide become available once the limitation is resolved.

30.4 Troubleshooting search quality

Use this sequence when search results look wrong:

1. Confirm the connector finished indexing: **Knowledge** flyout > **Existing Connectors**.
2. Confirm the relevant document set is **Up to Date** if you are scoping by document set.
3. Use the **Document Explorer** to confirm the specific document is actually in the index and is marked searchable.
4. Check **Document Feedback** for documents that users have flagged, and hide any that consistently degrade answers.

Most search complaints resolve at step 1 or step 3.

Part V

Part V: Sophea Tools

Chapter 31

Sophea Agent CLI

Sophea Agent CLI is a terminal client for working with a Sophea workspace from outside the browser. It gives you an interactive coding-agent session anchored to a local project, one-shot prompts for scripts and CI, and direct access to the workspace agents published through Nous, including private knowledge search.

This page covers install, login, workspace selection, interactive sessions and models, one-shot prompts, workspace agents and their artifacts, and the trust restrictions the CLI enforces. Use the CLI when you want a fast verification path that does not depend on opening the web app.

31.1 Install

Sophea Agent CLI ships as the `sophea-cli` package in the Sophea repository. The installer requires Bun 1.3.14 or newer on the machine.

```
cd sophea-cli
./install.sh
```

The installer stages the CLI under `~/.local/share/sophea-cli`, installs its pinned dependencies there, and writes a `sophea` launcher into `~/.local/bin` (set `SOPHEA_INSTALL_BIN_DIR` to choose a different bin directory). The launcher always runs the installed copy, never the source checkout. Confirm the install with:

```
sophea --version
```

```
sophea 0.1.0
```

The installer is idempotent: rerun `./install.sh` to move to a newer checkout or to repair an install. A failed update rolls back and leaves the previous install working.

> **Note**
>
> The CLI keeps all of its state, including credentials and chat sessions, in a dedicated protected profile that it owns exclusively. It does not read or migrate state from earlier Sophea CLI versions or from other tools. First use always starts with a fresh `sophea login`.

31.2 Log in (device auth)

Run `sophea login` to start the device-auth flow against Sophea Portal.

```
sophea login
```

```
Open the following URL in a browser and enter the code:

  https://portal.example.com/auth/device
  Code: BXKR-7Q2M

Waiting for approval...
```

Open the printed URL in a browser. Portal shows the **Authorize Coding Agent** page. Enter the code printed by the CLI into the **Verification code** field and click **Authorize Access**.

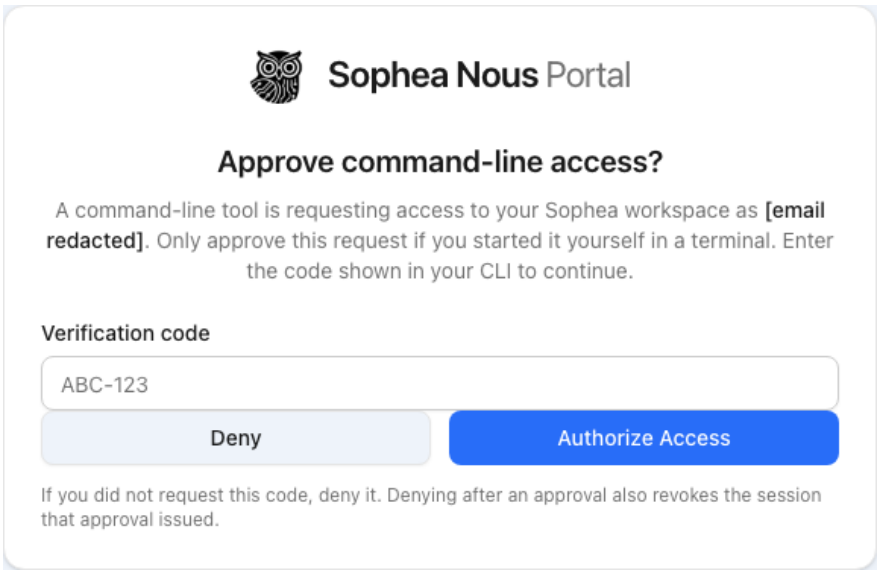

Figure 31.1: Portal Authorize Coding Agent page where you enter the code printed by the CLI

The CLI polls Portal until you approve the code, then stores the credential in its protected profile and prints the workspace list.

The Portal URL is resolved in this order: the `--portal-url` flag, the `SOPHEA_PORTAL_URL` environment variable, then the URL saved by a previous login. The CLI only prompts for a URL on an interactive terminal.

> **Tip**
>
> If your terminal cannot open URLs automatically, copy the printed URL into a browser on any device. The device-code flow is bound to the code, not to the machine running the CLI. This is the supported path for SSH sessions and CI bastions.

31.3 Log out / status

`sophea status` shows the current login, active workspace, and active model. Add `--json` for a machine-readable single JSON object on stdout.

```
sophea status
```

```
Signed in as: Demo Workspace Admin
Portal:       https://portal.example.com
Workspace:    Admin Guide Demo Workspace
Model:        <workspace default model>
```

`sophea logout` revokes the credential on the Portal side and then removes it from the local profile. Use it before handing a shared machine to another operator.

```
sophea logout
```

```
Signed out.
```

If Portal cannot be reached, the logout fails safely: the local credential is kept, the command exits nonzero, and you are asked to retry. After a successful `logout`, every other subcommand exits with code `2` and asks you to run `sophea login` again.

31.4 Pick a workspace

If your Portal account belongs to more than one workspace, the CLI needs to know which one to act against. Run `sophea workspace` to see the list and pick one interactively, or pass an id or name directly to switch.

```
sophea workspace
sophea workspace "Admin Guide Demo Workspace"
```

Switching the workspace rotates your credential to the new workspace and ends any active interactive session. Start a new session after switching.

> **Note**
>
> The interactive picker only renders on a TTY. For CI jobs, cron tasks, and non-interactive shells, pass `--workspace <id-or-name>` on the `sophea login` command to pin the workspace at login time. If the account has several workspaces and none is pinned in a non-TTY shell, login exits with code `3` and asks for `--workspace`.

31.5 Interactive sessions

Running `sophea` (or `sophea chat`) opens an interactive coding-agent session anchored to the current working directory. The agent can read and edit project files, run shell commands, and call your workspace agents, under the boundaries described in **Trust restrictions** below.

```
cd /home/demo/projects/admin-guide
sophea
```

Passing a prompt directly to `sophea` is not supported; use `sophea run "<prompt>"` for non-interactive prompts.

Session flags:

Flag	What it does
`--cwd PATH`	Anchor the session to a different working directory.
`--add-dir PATH`	Give the agent access to an additional directory (repeatable).
`--model ID`	Start on a specific workspace model instead of the default.
`--thinking LEVEL`	Set the reasoning effort for models that support it.
`--hide-thinking`	Hide reasoning output in the session.
`--no-title`	Skip automatic session title generation.
`--continue` / `-c`	Reopen the most recent session.
`--resume` / `-r` / `--session [ID]`	Pick or name a saved session to resume.

Sessions are saved automatically and are bound to the Portal account and workspace that created them. You can resume a session only while logged in as the same account in the same workspace; after switching workspace or re-logging in as a different account, older sessions are refused (exit code 3) instead of being silently mixed.

Inside a session, the Sophea-specific slash commands are:

Slash command	What it does
`/sophea-agents`	List the live workspace agents.
`/sophea-agent <slug> <prompt>`	Run a workspace agent.
`/sophea-agent-artifacts <slug> <operation-id>`	List the artifacts of an agent run.
`/sophea-agent-download <slug> <operation-id> <artifact-id> [relative-path]`	Download one artifact.
`/sophea-workspace`	Show the active workspace.
`/sophea-status`	Show login, workspace, and model status.
`/sophea-logout`	Revoke the credential and sign out.

Standard session navigation and model-selection commands are also available; see the in-session help.

31.6 Models

The model list is your workspace's live catalog from Sophea Portal, fetched fresh before every launch. The first catalog entry is the default model. `--model <id>` is validated against the live catalog before anything runs; an id that is not in the current catalog is rejected. If a saved session or previous selection names a model that is no longer published, the CLI switches to the workspace default and tells you.

31.7 Run a one-shot prompt

`sophea run "<prompt>"` runs a single prompt non-interactively and prints the answer to stdout. Use this for scripted checks and for piping into other tools.

```
sophea run "Summarize the README of the project in the current directory."
```

Options: `--cwd PATH` to pin the working directory, `--model ID` to override the model, `--agent SLUG` to route the prompt through a specific workspace agent, `--operation-id ID` to set an explicit idempotency key for an agent run, and `--json` for machine-readable output.

With `--json`, the command writes exactly one JSON value to stdout on success or failure; all progress and status messages stay on stderr, so piping into tools like `jq` is safe.

31.8 Workspace agents

`sophea agents list` prints the agents enabled for the active workspace. The list is fetched live from Nous and reflects what your workspace can actually use, including the `nous-rag` knowledge-search agent when your administrator has enabled it.

```
sophea agents list
```

```
SLUG                NAME                DESCRIPTION
nous-rag            Knowledge Search    Searches the workspace knowledge base.
my-research-agent   My Research Agent   Researches a topic across workspace sources.
legal-summarizer    Legal Summarizer    Summarizes contracts and clauses.
```

`sophea agents run <slug> "<prompt>"` starts an agent run:

```
sophea agents run my-research-agent \
  "Find the most recent contract that mentions Vendor X and quote the renewal clause."
```

Each run is idempotent. The CLI prints an operation id before the run starts; pass the same id back with `--operation-id` and a repeated command returns the original run's result instead of starting a duplicate. If you interrupt a run with `Ctrl+C`, the CLI cancels the remote run; rerunning with the printed operation id resumes tracking it.

All `agents` subcommands accept `--json` for the single-JSON-value stdout contract described above. When you interrupt an operation-scoped command such as `agents run`, `agents artifacts`, or `agents download`, that one JSON value includes the operation id. For example, `Ctrl+C` returns `{"ok":false,"exit_code":130,"operation_id":"agent_9f2c41ab07e3d5126b48f0a1"}`. An interrupted `agents list` command has no operation id.

31.9 Agent artifacts

Some agents produce files such as reports, exports, or generated documents. List them with `sophea agents artifacts` and fetch them with `sophea agents download`:

```
sophea agents artifacts my-research-agent agent_9f2c41ab07e3d5126b48f0a1
sophea agents download my-research-agent agent_9f2c41ab07e3d5126b48f0a1
↪ vendor-x-report.pdf
```

Downloads land under `.sophea-artifacts/<operation-id>/` in the current working directory by default. Use `--output <relative-path>` to choose a different location inside the project. Downloads never overwrite an existing file, refuse unsafe paths, and reject files larger than 100 MiB.

31.10 Tools and skills in interactive sessions

Inside an interactive session, the coding agent can call four Sophea tools on your behalf:

- `sophea_agents`: list the live workspace agent catalog.
- `sophea_run_agent`: run a workspace agent by slug.
- `sophea_agent_artifacts`: list the artifacts of an agent run.
- `sophea_download_agent_artifact`: download one artifact into the project.

Two built-in skills guide the agent: `nous-workspace-agents` teaches it how to pick the right workspace agent, run it, and verify results, and `nous-knowledge-search` teaches it when to answer from local files and when to search your private workspace knowledge through `nous-rag`.

You do not need to install or manage these; they ship inside the CLI package.

31.11 Exit codes

Code	Meaning
`0`	Success.
`1`	Local usage error or a failure while running.
`2`	Authentication required or expired; run `sophea login`.
`3`	Workspace unavailable, or a session/account/workspace mismatch.
`4`	Unknown, disabled, or failed workspace agent.
`130`	Interrupted with `Ctrl+C` (SIGINT).
`143`	Terminated by SIGTERM.

31.12 Trust restrictions

The CLI limits its models, loaded tools, and remote workspace identity. It does not sandbox the local shell:

- Only models from your workspace's live Portal catalog can be used.
- Allowed tools run automatically, without an approval prompt. This includes shell commands, file writes and edits, workspace agent runs, and artifact downloads.
- The CLI does not load extensions, plugins, MCP servers, skills, or commands from your machine or from the project; the only tools and skills available are the ones shipped inside the CLI package.
- The built-in file tools block access to the CLI profile, installation, and legacy credential file. These checks do not apply to shell commands and are not a shell sandbox.
- Shell commands run with your local user's filesystem access. They can access paths outside the project when your user can access them.
- Agent runs are authenticated as you and scoped to your active workspace; private knowledge search through `nous-rag` uses your own document permissions.

31.13 Current limitations

- Nous personas are not available through the CLI in this release; use the workspace agents listed by `sophea agents list`.
- Standalone image generation is not available through the CLI.
- The CLI is installed locally from the Sophea repository; an npm package or standalone binary distribution is not available yet.

For the bigger picture on how the CLI relates to Knowledge Search and workspace agents, see Knowledge Search and Agents.

Chapter 32

Slack bot

> Note
>
> Workspace admins manage Slack bots from Settings. If **Slack Integration** is absent from Settings, confirm you are a workspace admin.

The Slack bot lets workspace members talk to Sophea agents from inside Slack. The bot uses Slack's **Socket Mode**, so there is no public HTTPS endpoint, no request URL, and no signing secret to configure on the Slack side. You register a Slack app once on api.slack.com, install it to your workspace, then paste a Bot Token plus an App-Level Token into Nous.

This chapter walks through the full path: create the Slack app with the scopes Nous needs, enable Socket Mode and mint an App-Level Token, install the app, add the bot in Nous at `/app/admin/bots`, route channels to agents, and confirm that mentions and direct messages reach Sophea.

32.1 Prerequisites

Before you start, confirm the following:

- You can sign in to the target Slack workspace as an admin, or you have an admin who can install your Slack app for you.
- You can sign in to Nous as a **workspace admin**.
- At least one agent is published in your workspace. See Agents if no agent is published yet.
- Optional but recommended: at least one document set is ready. See Document Sets.

> Tip
>
> Pick a private dev channel in Slack for the first test, for example `#sophea-dev`. Test mentions and direct messages there before you invite the bot to channels your users see.

32.2 Register a Slack app

Open https://api.slack.com/apps in a new tab and follow these steps.

1. Click **Create New App**, then choose **From scratch**.
2. Enter an app name, for example `Sophea Bot`, and select the target Slack workspace.
3. Click **Create App**. Slack drops you on the app's settings page.

32.2.1 Configure bot scopes

Open **OAuth & Permissions** in the left sidebar and add the scopes Nous needs to read mentions, read channel history for context, and post replies.

Add the following bot token scopes:

- `app_mentions:read`: lets the bot see when users `@mention` it in a channel.
- `channels:history`: lets the bot read recent messages in public channels for thread context.
- `groups:history`: same as above for private channels the bot is invited to.
- `im:history`: lets the bot read direct messages to itself.
- `im:read`: lets the bot list direct-message conversations.
- `im:write`: lets the bot open a DM with a user.
- `chat:write`: lets the bot post replies.
- `chat:write.public`: lets the bot post in public channels it has not joined.
- `users:read`: lets the bot resolve user IDs to names for personalized replies.
- `team:read`: lets the bot read basic workspace info.
- `files:read`: optional, only add this if your channel routing needs to inspect attachments.

Add more scopes only if your channel routing needs them. Slack will warn users on reinstall whenever you change scopes.

32.2.2 Enable Socket Mode and mint an App-Level Token

Sophea's Slack bot connects to Slack over a WebSocket, so you do not configure an Event Subscriptions request URL.

1. Open **Socket Mode** in the left sidebar and toggle **Enable Socket Mode** on.
2. When Slack prompts you for an **App-Level Token**, give the token a name (for example `Sophea s ocket`) and add the `connections:write` scope. This is the only scope this token needs.
3. Copy the generated token. It starts with `xapp-`. Slack will not show it again, so keep it in a scratch buffer.

Open **Event Subscriptions** in the left sidebar and toggle **Enable Events** on. Leave the request URL empty; events arrive over the Socket Mode connection. Under **Subscribe to bot events**, add at least:

- `app_mention`: the bot is mentioned in a channel.
- `message.im`: a user sends a direct message to the bot.
- `message.channels`: a message is posted in a public channel the bot is in. Add this only if you want the bot to react without an explicit mention.
- `message.groups`: same as above for private channels.

Save changes.

32.2.3 Install the app and capture the Bot Token

Open **Install App** in the left sidebar and click **Install to Workspace**. Approve the requested scopes on the Slack consent screen.

Slack returns you to the app page. Copy the **Bot User OAuth Token** from **OAuth & Permissions**. It starts with `xoxb-`. You now have the two required values, plus an optional third:

- **Bot Token** (`xoxb-`): from **OAuth & Permissions**.
- **App-Level Token** (`xapp-`): from **Basic Information** -> **App-Level Tokens**, the one you minted with `connections:write`.
- **User Token** (`xoxp-`, optional): only if you need user-impersonation features that the bot cannot perform with its bot token. Most workspaces leave this blank.

> **Warning**
>
> Both tokens are credentials. Treat them like passwords. Do not paste them into chat, do not commit them to git, and do not share them on screenshots when you record onboarding videos. Rotate any token from the Slack developer portal if you suspect it leaked.

32.3 Add the bot in Nous

Open the **Integrations** section in the left sidebar and click **Slack Integration**. This opens the Slack Integration page at `/app/admin/bots`. The page lists every Slack bot already registered, with columns for **Name**, **Status**, **Default Config**, and **Channel Count**.

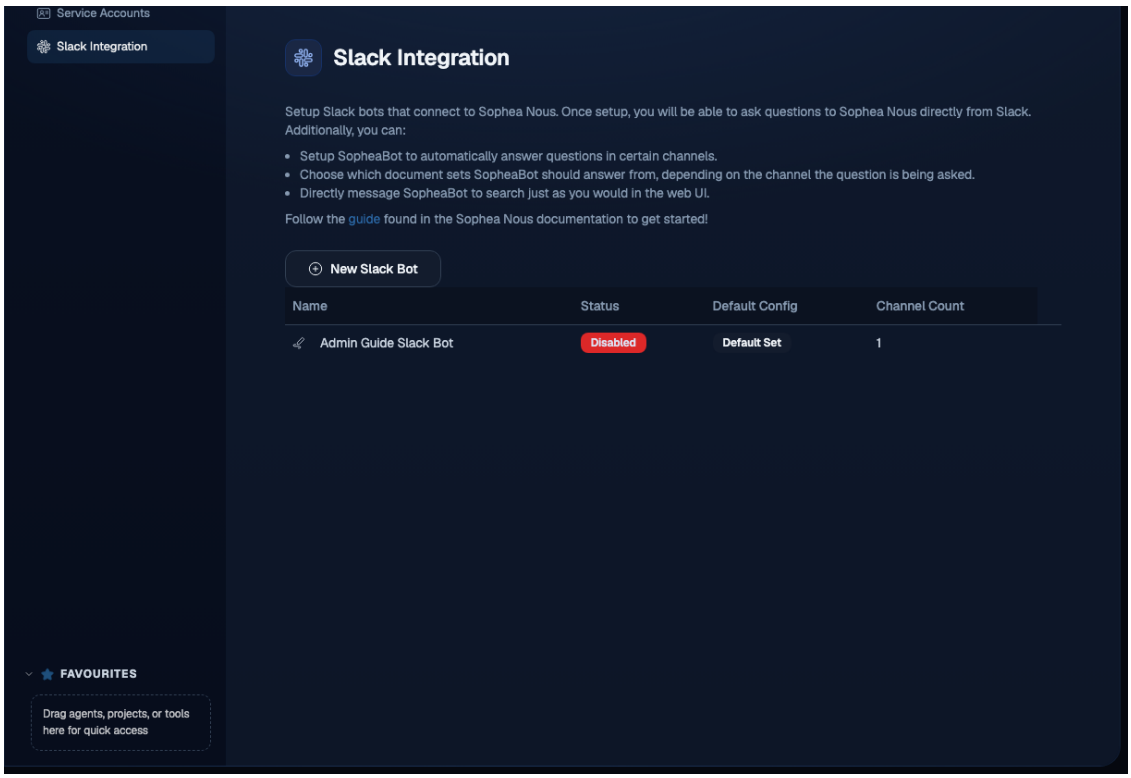

Figure 32.1: Slack Integration page showing the bot list

Click **New Slack Bot** in the top left area of the list to open the registration form.

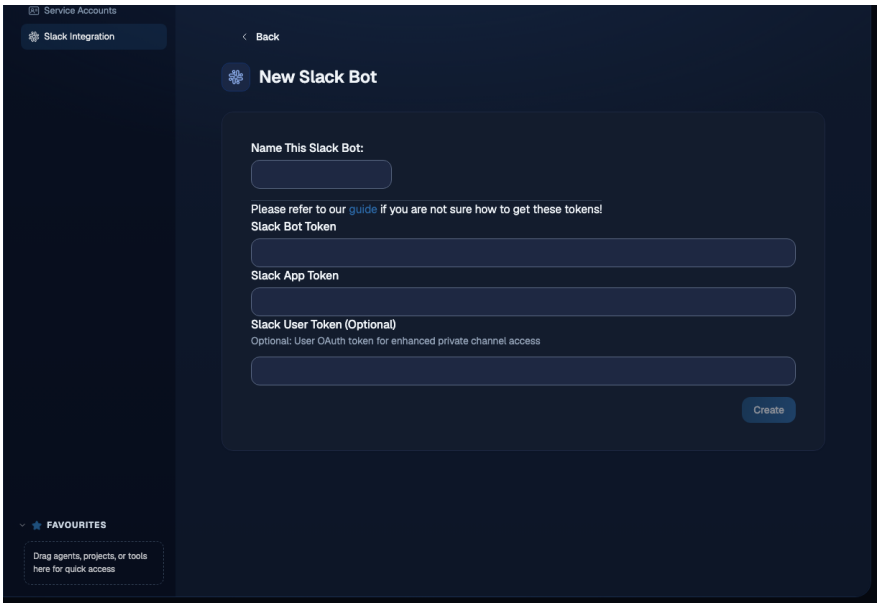

Figure 32.2: New Slack Bot form with name and token fields

Fill in the form:

- **Name This Slack Bot**: a human-readable label, for example `Admin Guide Demo Bot`. This is only shown in the admin UI.
- **Slack Bot Token**: paste the `xoxb-` token from **OAuth & Permissions**.
- **Slack App Token**: paste the `xapp-` token you minted under **Basic Information** -> **App-Level Tokens**.
- **Slack User Token (Optional)**: paste an `xoxp-` token only if you need enhanced private channel access.

Click **Create**. Nous verifies the tokens by calling Slack's API and opening the Socket Mode connection. If a token is invalid or the workspace cannot reach Slack, the form returns an error and nothing is saved.

After creation, the bot appears in the list. Nous opens the Socket Mode connection in the background; once Slack delivers the first event, the **Status** column updates accordingly.

32.4 Route channels to agents

Open the bot from the list. The detail page shows the channel routing configuration.

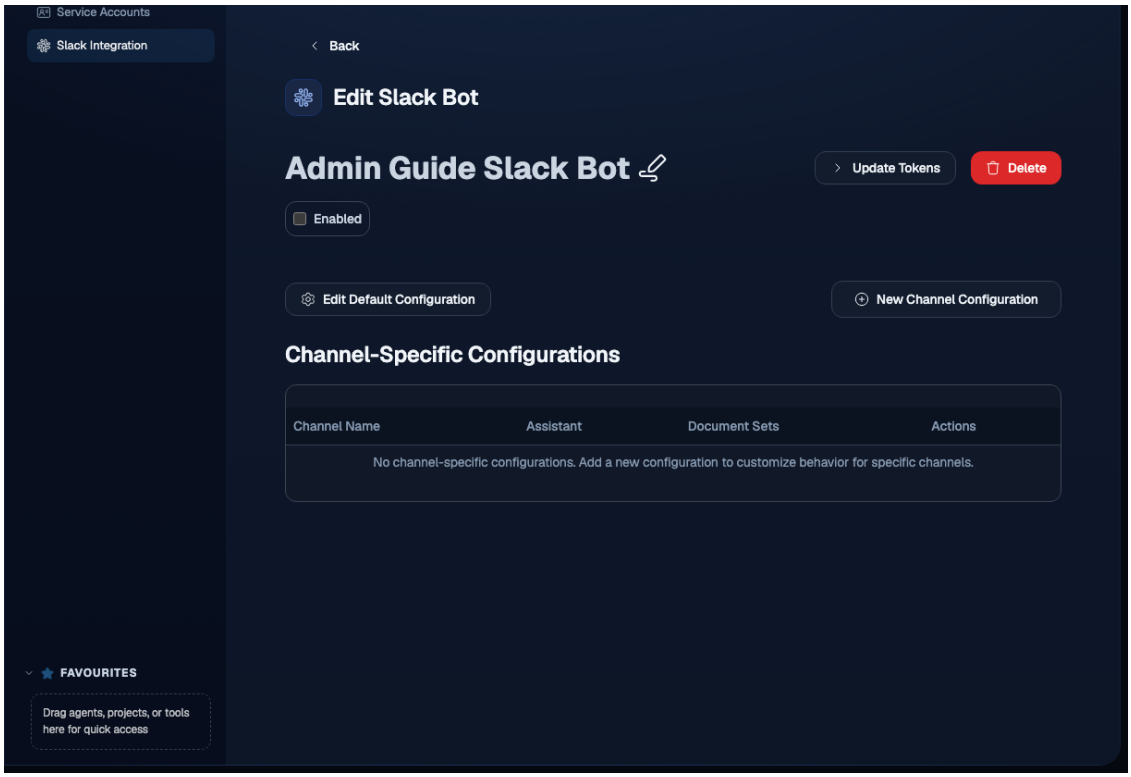

Figure 32.3: Channel routing configuration for a Slack bot

Channel routing tells Nous which agent should answer when the bot is mentioned in a specific channel.

To add a route:

1. Click **Add channel**.
2. Paste the Slack channel ID, for example `C0123456789`. You find this in Slack by right-clicking the channel, then choosing **View channel details** and scrolling to the bottom.
3. Pick the agent from the **Agent** dropdown. Only agents already published in this workspace are listed.
4. Optionally restrict the channel to one or more document sets. The agent will only cite eligible documents from these sets. Ordinary shared-channel replies can search workspace-public content only; a private set does not make its documents visible there.
5. Click **Save**.

Repeat for every channel you want to bind. Channels without a route fall through to the bot's default persona, if one is set. Otherwise the bot stays silent in unrouted channels.

> **Note**
>
> The bot must be invited to a private channel before it can receive events from that channel. In Slack, run `/invite @Sophea Bot` inside the channel. Public channels work as soon as the channel ID is added in Nous.

To remove a route, hover the row and click the trash icon. To change the agent on a route, click the

agent name and pick a different one. Changes apply on the next mention.

32.5 Test and troubleshoot

Run these checks in your dev channel, in order, before you announce the bot to users.

32.5.1 Direct message check

1. In Slack, click the bot's name in the sidebar **Apps** section to open a DM.
2. Send a short prompt, for example `hello, are you up?`.
3. The bot replies within a few seconds. If it does not, the most likely cause is a missing `im:history` or `chat:write` scope. Reinstall the app after adding scopes.

32.5.2 Channel mention check

1. Invite the bot to your dev channel: `/invite @Sophea Bot`.
2. Mention it: `@Sophea Bot what is in our onboarding doc set?`.
3. The bot replies in the same thread. The reply cites the document set bound to this channel.

If the bot stays silent on a mention:

- Confirm the channel ID is registered under channel routing.
- Confirm an agent is selected for that channel, or a default persona is set on the bot.
- Check the **Status** column on the Slack Integration page and confirm it updated after your mention. If it did not update, Slack could not deliver the event over the Socket Mode connection. Open the bot, recheck that the App Token (`xapp-`) is still valid in Slack, and re-save the bot to reopen the connection.

32.5.3 Permissions check

Slack response visibility depends on how the reply is delivered:

- A bot direct message or an ephemeral reply can use the mapped Nous user's private-document permissions.
- An ordinary reply posted into a shared channel deliberately searches as an anonymous user and can use workspace-public documents only, even when the sender personally has access to private content.

If a shared-channel reply has no citations, confirm the routed document sets contain workspace-public content. To test a member's private access, use a direct message or an ephemeral response and compare it with the same query in the member's normal Nous chat at `/app`.

32.5.4 Common failures

- `invalid_auth` **on create**: the Bot Token (`xoxb-`) was copied incorrectly or was rotated in Slack. Recopy from **OAuth & Permissions** -> **Bot User OAuth Token**.
- `invalid app token` **on create**: the App-Level Token (`xapp-`) is missing the `connections:write` scope, was deleted, or was pasted into the wrong field. Recreate it under **Basic Information** -> **App-Level Tokens** and paste the new value.
- **Bot ignores mentions in a public channel it just joined**: confirm `app_mentions:read` and `chat:write.public` scopes, then reinstall the app.

- **Bot replies but with no citations**: the channel route has no document set selected, or the bound agent has Use Knowledge turned off. Open the agent in `/app/agents` and confirm. See Agents.
- **Slack Integration is missing from Settings**: confirm you are a workspace admin.

Once the dev channel passes all three checks, invite the bot to its target user channels and announce it.

Chapter 33

Discord Bot

> **Warning**
>
> This feature has been discontinued. Discord channel indexing is available through the Discord connector in **Knowledge** → **Add Connector**, but workspace-managed Discord bot hosting is no longer supported.

Chapter 34

Digital Employees

Digital Employees are Workspace-scoped AI assistants with dedicated instructions and optional capabilities. Managers configure them in Portal, and Workspace members chat with active employees in Nous.

Digital Employees is currently a Beta product and must be enabled for each Workspace by a Sophea platform administrator.

34.1 Open Digital Employees

For an Organization Workspace in Portal:

1. Open the Organization.
2. Select **Digital Employees** in the account navigation.
3. Find an active Workspace where the product is **Enabled**.
4. Click **Open**.

You can also open the Workspace card and use the Digital Employees product row. For workspace-only access, open the Workspace under **Shared with me** and use its Digital Employees link. Controls remain unavailable until Workspace setup finishes and the product is enabled.

In Nous, select **Digital Employees** from the main navigation to see employees available in the current Workspace.

34.2 Know what each role can do

Role	Portal management	Nous chat
Organization owner or Organization admin	Can manage employees in Organization Workspaces through their effective Workspace authority	Can chat with active employees
Workspace admin	Can create, edit, activate, deactivate, and delete employees in that Workspace	Can chat with active employees

Role	Portal management	Nous chat
Workspace collaborator	Read-only employee view	Can chat with active employees

Managing an employee does not grant extra document-search permission. When an employee uses Workspace Search, results are limited by the access of the person using the chat.

Portal shows only employees assigned to Teams that you actively belong to and that have access to the Workspace. This Team boundary also applies to Organization owners and Workspace admins.

Each Digital Employee belongs to one immutable Team. The current management view shows the Team and the inherited knowledge access as read-only. Revoking the Team's Workspace grant denies the employee access, but does not change its lifecycle. Regranting the Team's Workspace access restores the employee's access. Portal blocks Team deletion until every attached Digital Employee is fully deleted.

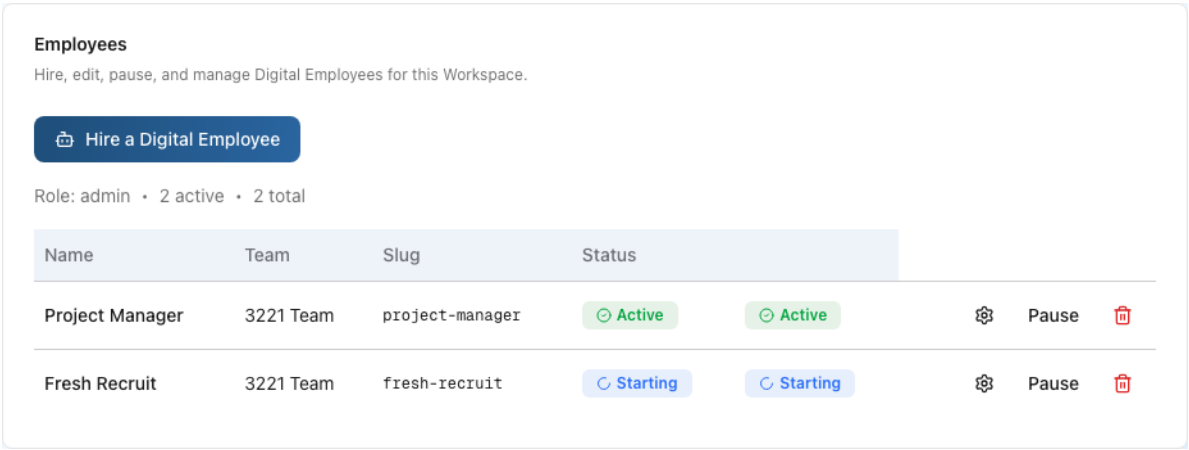

Figure 34.1: Digital Employee management table in Portal

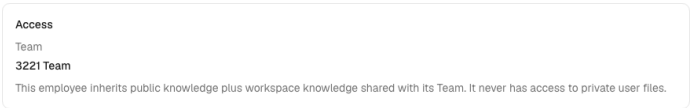

Figure 34.2: Read-only Team access shown while editing a Digital Employee

34.3 Create an employee

On the Workspace's Digital Employees page in Portal, use **Hire a Digital Employee** and complete the wizard:

- **Workspace**: the wizard opens with a summary of the Workspace you came from. Use **Change** to pick a different Workspace instead.
- **Team**: the one Team that owns the employee's access. Portal offers only your Teams with access to this Workspace.
- **Role preset**: a starting point that pre-fills every setting; you can adjust everything afterward.
- **Display name**: the name members see in Nous.

- **Description**: a concise explanation of the employee's purpose.
- **Instructions**: the behavior, responsibilities, constraints, and response style the employee should follow.
- **Capabilities**: enable only the tools required for the role.

Portal derives the slug automatically from the display name.

New employees show a setup state before they are ready. They appear in Nous after the Workspace change finishes applying.

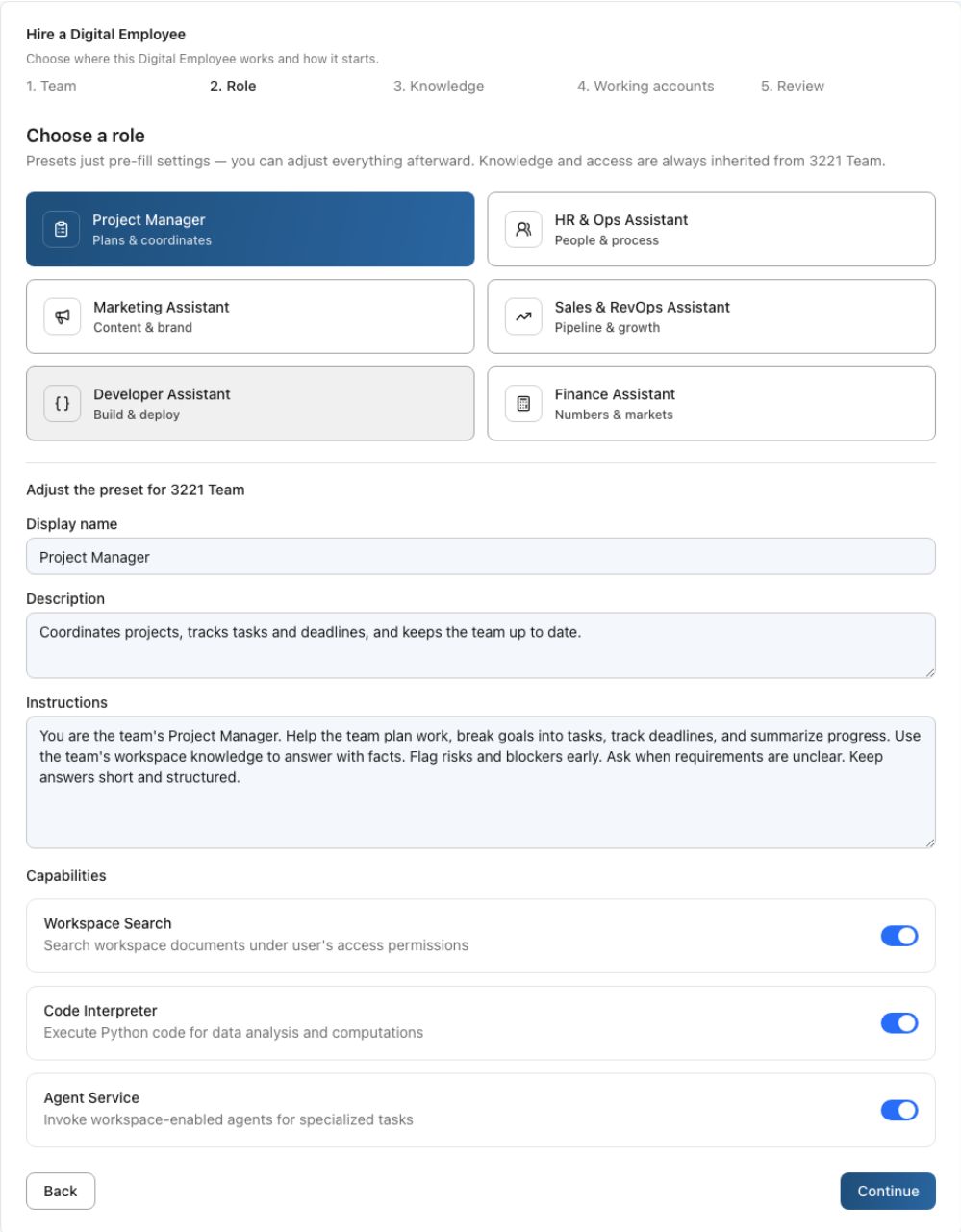

Figure 34.3: Hire a Digital Employee wizard

> **Tip**
>
> Start with one narrow responsibility. Add capabilities only when the employee's instructions require them.

34.4 Choose capabilities

Portal currently offers these capabilities:

Capability	Purpose
Workspace Search	Searches Workspace documents under the current user's access permissions.
Code Interpreter	Runs code for calculations, analysis, and data work.
Agent Service	Invokes Workspace-enabled specialist agents for deeper workflows.

Capabilities control which tools the employee may use. Instructions describe when and how it should use them. Enabling a capability does not bypass resource sharing, Workspace access, or another service's permissions.

34.5 Understand the fixed model policy

Sophea manages the Digital Employee model centrally. Workspace and employee model choices do not change this runtime. This keeps capabilities, image input, reasoning, billing attribution, and support behavior consistent for every employee. Keep role-specific behavior in the employee instructions.

34.6 Activate, pause, edit, or delete

Workspace admins can manage each employee from its Portal row:

- **Pause** pauses new chat work without deleting the employee.
- **Resume** makes a paused employee available again after the change finishes.
- **Delete** removes an employee that is no longer needed.
- **Settings** opens the employee's detail page. Its **Edit configuration** tab changes the display name, description, instructions, or capabilities. The slug stays fixed after creation.

Changes can briefly place the employee in a setup or updating state. Wait for a ready state before testing it in Nous.

34.7 Manage recurring schedules

A recurring schedule fires a task for the employee on a fixed cadence, for example a digest every weekday morning. Each fired task appears on the employee's board like any other task.

Create a schedule from the employee's dashboard in Nous:

1. Open the employee and switch to the **Scheduled** segment.
2. Select **New recurring job**.
3. Enter a title and the work to perform, pick a cadence (**Every day**, **Weekdays**, **Every week**, or **Every month**) and a time, and save.

The dialog notes the detected timezone: schedules always run at that clock time in the creator's timezone.

Each row shows the cadence and the next run. The cadence label renders in the reader's own interface language; a schedule saved as a custom cron expression keeps its author-written label instead.

Row actions:

- **Edit** changes the title, the work to perform, or the cadence.
- **Disable** pauses firing without deleting the schedule. **Enable** starts it again, and the next run counts from that moment.
- **Delete** removes only the schedule. Tasks it already fired stay on the board.

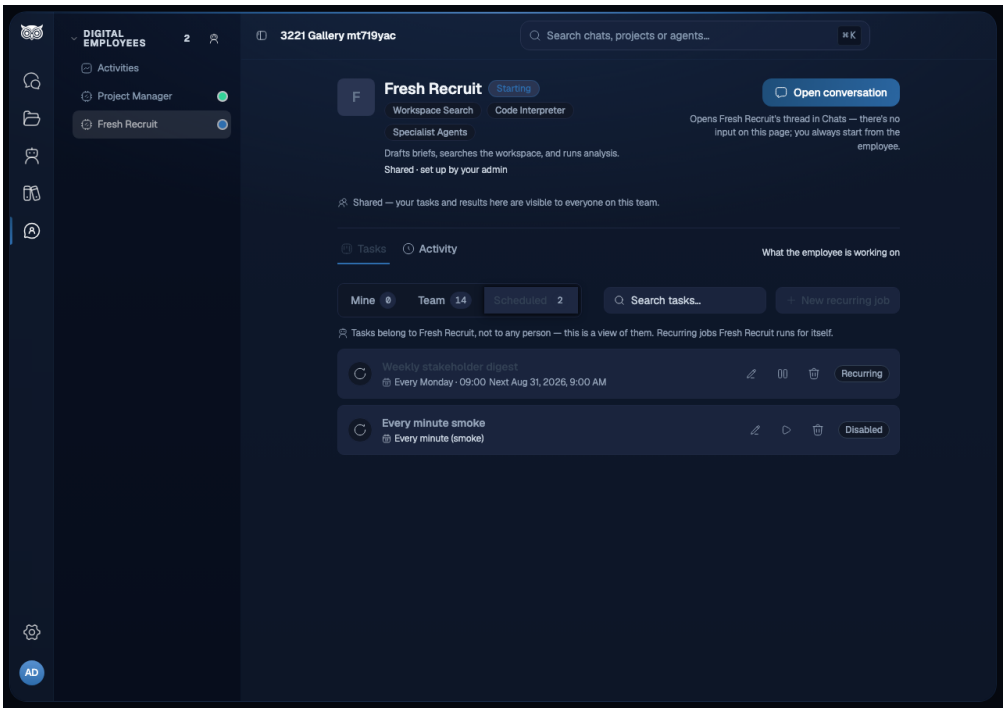

Figure 34.4: Scheduled segment listing a recurring job with its cadence, next run, and row actions

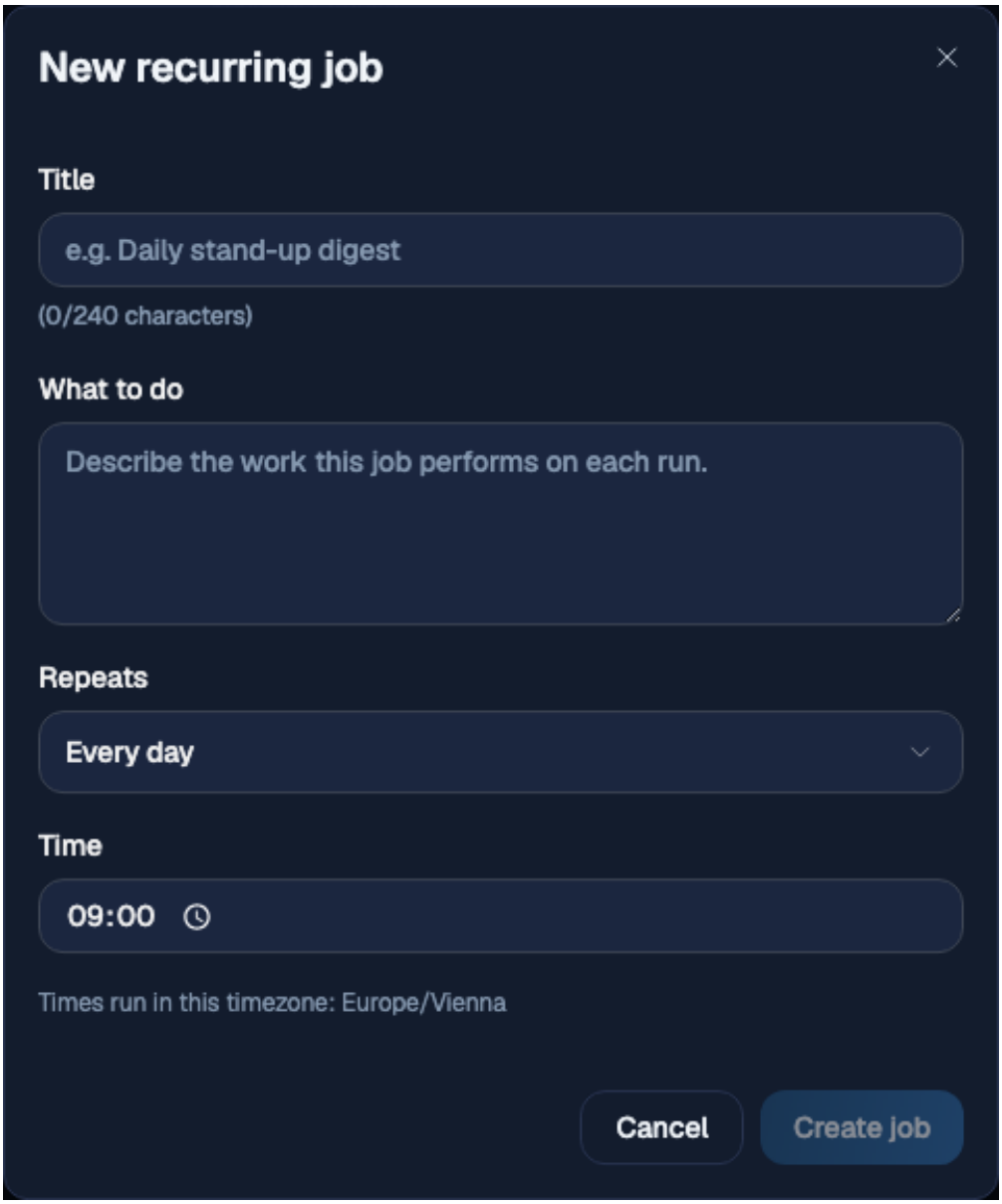

Figure 34.5: New recurring job dialog

34.8 Understand employee status

Nous presents product-language status labels:

Status	Meaning
Ready	The employee is available for chat.
Starting	Setup is still finishing. Wait before sending work.
Paused	The employee is inactive and cannot accept new chat work.

Status	Meaning
Suspended	The employee is temporarily unavailable. Ask a Workspace admin to review it.
Needs attention	Setup or availability failed. A Workspace admin should review the employee in Portal and contact support if it does not recover.

34.9 Chat with an employee in Nous

1. Open the correct Workspace in Nous.
2. Select **Digital Employees**.
3. Choose a **Ready** employee.
4. Review its description and enabled capabilities.
5. Enter a request or select a suggested starter prompt.

The dedicated chat shows the response as it is produced and identifies capability activity when the employee uses a tool. If the employee is not ready, the composer explains why it is unavailable.

For Workspace Search, test with a document the current user is allowed to find. A successful result for a Workspace admin does not prove that a Workspace collaborator can retrieve the same private content.

34.10 Review approval requests

An approval request attaches to a completed task and gates its outcome before anyone acts on it. The current release records the decision; automatic enforcement of the outcome follows later.

Request an approval from a completed task's conversation in Nous:

1. Open the completed task's conversation.
2. Select **Request approval**.

While a request is pending, watchers of the employee see a **Needs your input** strip on the employee's **Activity** feed in Nous.

Decide in either place:

- In Nous, open the task's conversation and choose **Approve** or **Reject**, optionally adding a note. The note posts to the task conversation under the decider's own name.
- In Portal, open the employee's **Activity** tab and use the pending approvals section.

A task carries at most one pending request at a time. Anyone in the Workspace may decide, and the decider is recorded with the decision.

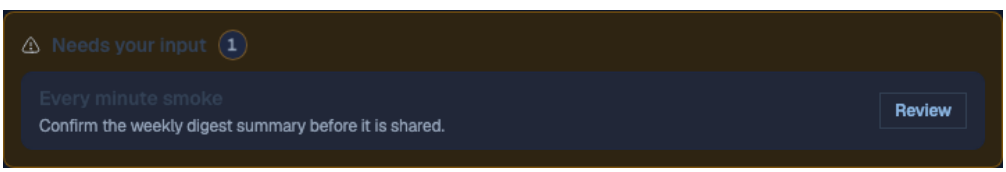

Figure 34.6: Needs your input strip on the employee Activity feed

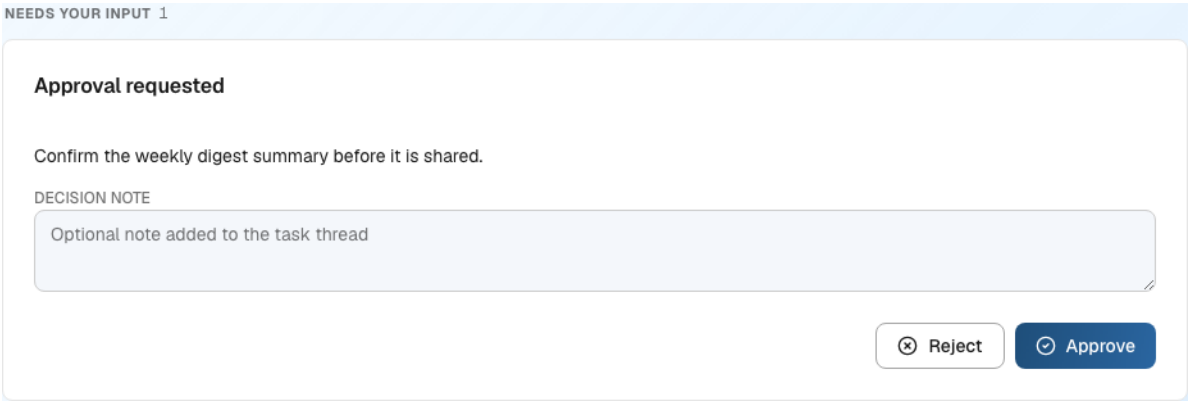

Figure 34.7: Pending approvals section on the Portal employee Activity tab

34.11 Troubleshoot availability

If Digital Employees is unavailable:

1. Confirm that the correct Workspace is selected.
2. In Portal, confirm the Workspace is **Active** and Digital Employees is **Enabled**.
3. Confirm your role. Workspace collaborators can chat but cannot create or change employees.
4. If an employee is **Starting**, wait for setup to finish and refresh Nous.
5. If it is **Needs attention** or remains unavailable, ask a Workspace admin to review it in Portal and contact Sophea support.

For role and access-source details, see Organizations and Workspace access.

Chapter 35

Sophea Nous Desktop App

Sophea Nous Desktop is the macOS client for Sophea Nous: a thin native wrapper around the same web app you use in the browser. The web app stays the product surface; the desktop shell adds a persistent window that remembers its size and position, OS menus and keyboard shortcuts, and quick switching between Sophea deployments.

This page covers installing the app from GitHub Releases, passing the macOS Gatekeeper check on first launch, picking an instance, and signing in. Use the desktop app when you want Sophea in its own window instead of a browser tab.

35.1 Requirements

- A Mac with Apple Silicon (M1 or newer). Intel builds are not published yet.
- macOS 13 or newer.
- Access to a Sophea deployment: your organization's workspace, or the staging/local instances shipped with the app.

35.2 Download and install

Desktop releases are published on the Sophea repository's GitHub Releases page. Each release carries a disk image (`Sophea-<version>-mac-arm64.dmg`), a plain zip archive, and a `SHA256SUMS.txt` with the checksums of both.

1. Download the `.dmg` for the version you want.
2. Open the disk image and drag **Sophea** into **Applications**.
3. Eject the disk image.

To verify the download before installing, compare its checksum with `SHA256SUMS.txt`:

```
shasum -a 256 ~/Downloads/Sophea-*-mac-arm64.dmg
```

35.3 First launch: the Gatekeeper prompt

Desktop builds are not signed with an Apple Developer ID yet, so macOS Gatekeeper blocks the first launch. This is expected and safe to pass for a build you downloaded from the official Releases page:

1. Double-click **Sophea** in Applications. macOS shows a dialog saying the app cannot be opened. Dismiss it.
2. Open **System Settings > Privacy & Security**.
3. Scroll to the Security section, find the message about Sophea being blocked, and click **Open Anyway**.
4. Confirm in the follow-up dialog. The app opens, and macOS remembers your choice for future launches.

> Tip
>
> Right-clicking the app and choosing **Open** from the context menu reaches the same result in one step: the dialog gains an **Open** button when invoked this way.

> Warning
>
> Only pass the Gatekeeper check for a build you downloaded from the official GitHub Releases page. If you received the app through any other channel, do not click **Open Anyway**.

35.4 Choose an instance

On first run the app asks which Sophea instance to connect to:

- **Local**: a development stack running on the same machine (`http://localhost:3000`).
- **Staging**: the Sophea staging deployment.
- **Production**: the Sophea production deployment.
- **Custom URL**: any HTTPS origin (HTTP only for `localhost` targets), plus an optional separate auth host for deployments that keep their identity provider on its own hostname.

The choice is saved and can be changed any time from the **Instance** menu. The active instance drives the window title and the navigation rules: only the instance's app and auth origins may load inside the window, while external links open in your default browser.

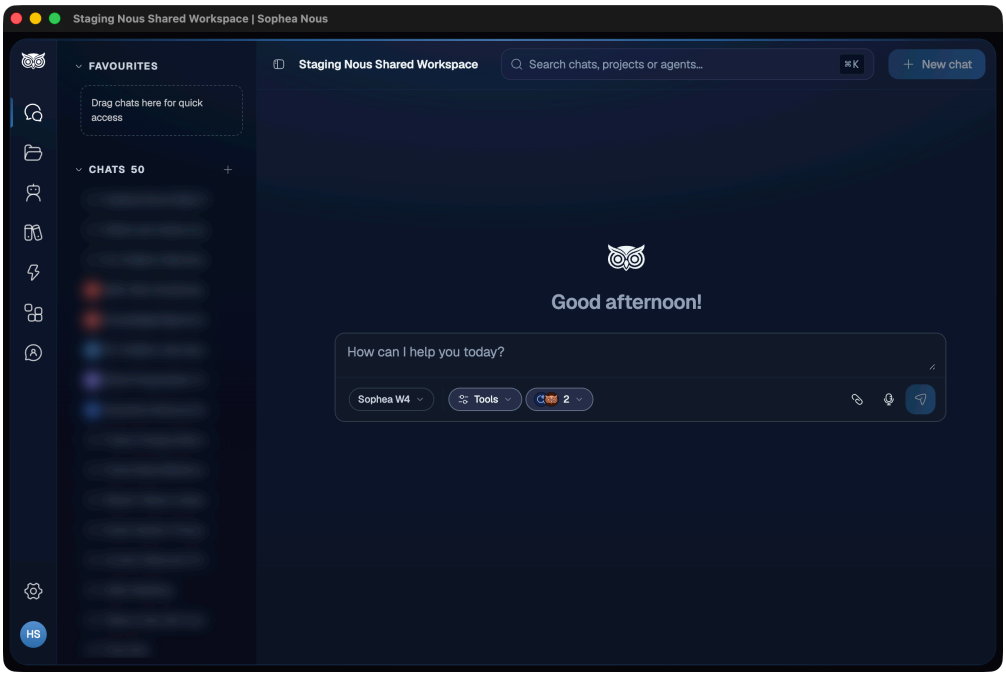

Figure 35.1: Sophea Nous Desktop signed in to Staging Nous Shared Workspace, showing the chat screen in its own window

35.5 Sign in

Login is the same Zitadel flow as the browser: the app redirects to the hosted login page, you sign in, and the session cookie persists across restarts. There is no separate desktop account.

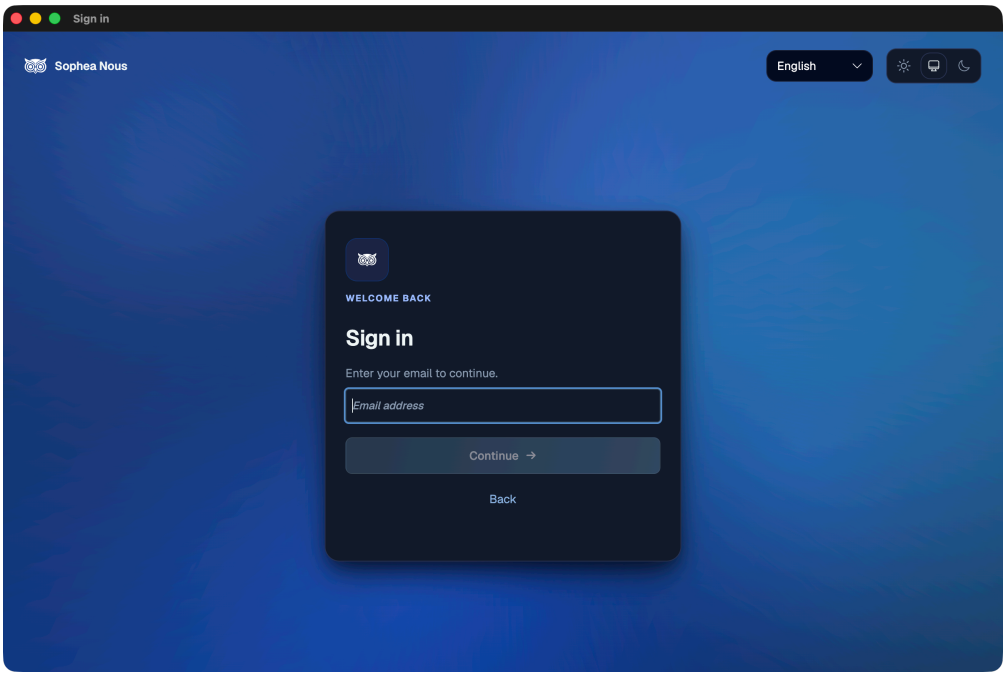

Figure 35.2: Sophea Nous Desktop showing the hosted Sophea Nous sign-in page inside its own window

To sign out or switch accounts, use **Sophea > Clear session and quit** from the menu. This wipes the stored session and window state and closes the app; the next launch asks for an instance and a fresh login.

35.6 Current limitations

- **No automatic updates.** New versions arrive as new GitHub Releases; install them over the old app as in Download and install.
- **macOS only.** Windows and Linux builds are planned follow-ups.
- **Apple Silicon only.** Intel Macs are not supported by the published artifacts.
- **No local folder sync yet.** The shell today is windows, menus, and instance management; folder sync arrives in a later release.

For the terminal-based companion on the same workspace, see the Sophea Agent CLI page in this guide.

Part VI

Part VI: Automations

Chapter 36

How Sophea Automations Works

36.1 Overview

Sophea Automations is the workflow automation layer of the Sophea platform. It runs embedded inside the Nous backend, sharing authentication, tenant isolation, and permissions with the rest of the platform. This document explains the architecture, the broker pattern, and how tenant isolation and ACL propagation work.

36.2 Architecture

Sophea Automations consists of two components:

1. **Automations API server** (TypeScript, Fastify): handles flow management, execution, and the web builder UI
2. **Automations web frontend** (React, Vite): the visual flow builder and dashboard

Both components run inside the Nous namespace in Kubernetes. The Automations API communicates with the Nous backend via HMAC-signed broker endpoints, and with the RAG service via signed ACL principals.

36.2.1 Identity and access lifecycle

The Automations account is bound to the member's stable Portal user identity, not to an email address. Changing or reassigning an email therefore does not merge Automations accounts or transfer their access.

The Automations platform is provisioned by the platform operator before customer access is enabled. Opening Automations can create the member's workspace project and project membership, but it cannot create the platform or grant platform-administrator authority.

For every user request to a workspace project, Automations checks the current effective workspace membership in Portal before applying its local project role. Removing a direct, Team-derived, or Organization-derived workspace grant blocks the member's next request, including requests made with an existing Automations session. If Portal cannot confirm membership, access fails closed.

```
Browser
```

```
  |
  v
Nous (nginx reverse proxy)
  |
  +--> Automations API (Fastify, TypeScript)
  |       |
  |       +--> Nous Broker (HMAC-signed HTTP)
  |       |      |
  |       |      +--> Nous RAG (signed ACL principals)
  |       |      +--> Agent Service (X-Sophea headers)
  |       |      +--> Portal (HMAC bearer auth)
  |       |
  |       +--> Flow Engine (Code steps and {{ }} expressions run in V8 isolates)
  |
  +--> Nous Backend (FastAPI, Python)
```

36.3 The Broker Pattern

The broker pattern is the core security mechanism for Sophea Automations. Instead of giving the Automations API direct database access, all data operations flow through HMAC-signed HTTP endpoints on the Nous backend.

36.3.1 How it works

1. A flow step executes in the Automations sandbox
2. The Sophea piece action calls `postToNousBrokerRaw()` with a broker path and payload
3. The broker client (`broker.ts`) resolves workspace identity from the flow context:
   - `workspaceId` from `context.project.externalId()`
   - `tenantId` from workspace mapping
   - `runAsUserId` from `context.run.triggeredBy.externalId()`
4. The client computes an HMAC-SHA256 signature over the request body and identity fields
5. The signed request is sent to the Nous backend broker endpoint
6. The Nous backend verifies the HMAC signature, checks the workspace membership, and executes the operation

36.3.2 Why this matters

- **No direct database access**: The Automations API never touches the Nous database directly
- **Tenant isolation enforced at the boundary**: Every broker request carries workspace identity, verified on every call
- **Auditable**: Every broker call is logged with workspace ID, run ID, step ID, and attempt number
- **Rate-limited**: Broker endpoints can enforce per-workspace rate limits

36.3.3 HMAC signature

The HMAC signature is computed over:

```
${timestamp}.${sha256(body)}.${projectId}.${runId}.${stepId}.${attemptNo}
```

Signed with SOPHEA_AUTOMATIONS_BROKER_SECRET (shared between Automations API and Nous backend). The signature is sent in the Authorization header as Bearer ${timestamp}.${signature}.

Additional headers: - X-Sophea-Automation-Project-ID - X-Sophea-Automation-Run-ID - X-Sophea-Automation-Step-ID - X-Sophea-Operation-ID - X-Sophea-Tenant-ID (if set) - X-Sophea-User-ID (if set)

36.4 Tenant Isolation

36.4.1 Workspace equals tenant

In Sophea, a workspace is a tenant. Every flow, run, and piece execution is scoped to a workspace. The workspace ID is propagated through:

1. **Flow context**: context.project.externalId() returns the workspace ID
2. **Broker requests**: workspace_id field in every broker request body
3. **Database queries**: The Nous backend opens a tenant-scoped database session for each broker call

36.4.2 Broker endpoint security

Every broker endpoint follows the same security pattern:

1. _headers_match_body(): verifies that the HMAC signature was computed over the same body that was received
2. verify_automation_broker_authorization(): verifies the HMAC signature and timestamp
3. _verify_runtime_member(): checks that the requesting user is an active member of the workspace
4. _tenant_id_from_member(): resolves the tenant ID from the workspace membership
5. _execute_runtime_broker_call(): opens a tenant-scoped database session and executes the operation

36.4.3 Shell mode

Shell mode restricts the available pieces to a vetted allowlist. This prevents untrusted pieces from running in production environments. The allowlist is defined in nous-automations-shell.ts and includes only Sophea-approved pieces.

Shell mode also pins the code execution mode. In every Sophea environment, AP_EXECUTION_MODE is SANDBOX_CODE_ONLY: Code steps and { } template expressions run inside fresh V8 isolates (isolated-vm) instead of the engine process. Sandboxed code receives only copied inputs; Node APIs (process, require, filesystem, network access, timers) are not available, and npm packages cannot be installed. Trusted Sophea piece code continues to run in the engine process, where it signs broker calls with SOPHEA_AUTOMATIONS_BROKER_SECRET; sandboxed user code cannot read that process or its environment. If AP_EXECUTION_MODE is set to UNSANDBOXED while shell mode is enabled, the engine rejects the combination and fails closed, so a configuration error cannot silently restore in-process code execution. UNSANDBOXED remains only for explicit trusted-operator development with shell mode off and is never a Sophea production option. V8 isolation is not a kernel boundary; separate process isolation remains follow-up hardening.

36.5 ACL Propagation

36.5.1 Signed ACL principals

When a flow step searches the knowledge base, the broker endpoint builds signed ACL principals that determine which documents the search can access. The ACL is built from:

1. **Public documents**: `PUBLIC_DOC_PAT` (all workspace members can access)
2. **User-specific documents**: `prefix_user_email(email)` (documents scoped to the user's email)

The ACL is signed with `NOUS_RAG_SERVICE_AUTH_KEY` using HMAC-SHA256 and sent to the RAG service in the `X-Sophea-Principals` header.

36.5.2 How it works

1. The broker endpoint receives the search request with workspace and user identity
2. `_signed_principals_for_member()` builds the ACL for the requesting user
3. The ACL is signed and sent to the RAG service
4. The RAG service verifies the signature and filters search results to only include documents the user can access

This ensures that flow steps can only access documents that the requesting user has permission to see, even when running in an automated context.

36.6 Cross-Service Headers

All cross-service calls carry Sophea-specific headers:

Header	Purpose
`X-Sophea-Tenant-ID`	Required: identifies the tenant
`X-Sophea-Run-ID`	Agent service idempotency key
`X-Sophea-User-ID`	Optional: identifies the user
`X-Sophea-Operation-ID`	Tracing and correlation
`X-Sophea-Principals`	Signed ACL for RAG service
`X-Sophea-Signature`	HMAC signature for webhook triggers
`X-Sophea-Timestamp`	Timestamp for webhook signature verification

36.7 Related documentation

- Enterprise Connectors
- Sophea vs Generic Automation Tools
- Product positioning: `docs/product-06-sophea-automations.md`
- RAG integration: `docs/product-04-sophea-nous-rag.md`
- Portal integration: `docs/product-02-sophea-nous-portal.md`

Chapter 37

Enterprise Connectors

37.1 Overview

Sophea Automations takes a different approach to connectors than generic automation tools. Instead of offering hundreds of connectors to external SaaS apps, Sophea focuses on controlled, enterprise-grade connectors that are scoped to tenants, respect permission boundaries, and integrate with the RAG pipeline.

37.2 How connectors are scoped to tenants

37.2.1 Tenant isolation

Every connector in Sophea is scoped to a workspace (tenant). When a flow step lists available connectors via the broker, the response only includes connectors that belong to the requesting workspace. The broker endpoint:

1. Resolves the tenant ID from the workspace membership
2. Opens a tenant-scoped database session
3. Queries `fetch_connectors()` filtered by tenant
4. Returns only connectors with `DocumentSource.INGESTION_API` (API-sourced connectors)

This means a flow in Workspace A can never see or access connectors from Workspace B.

37.2.2 Permission sync

Connector permissions are synced from the Portal. When a workspace member runs a flow, the broker:

1. Verifies the user is an active member of the workspace (`_verify_runtime_member`)
2. Builds signed ACL principals based on the user's email and workspace membership
3. Propagates the ACL to the RAG service for search operations

The ACL ensures that flow steps can only access documents that the requesting user has permission to see. If a user does not have access to a document set, no flow step they trigger can access it either.

37.3 RAG grounding

37.3.1 What is RAG grounding?

RAG (Retrieval-Augmented Generation) grounding means that AI actions in flows always have access to the workspace knowledge base. Instead of sending prompts to an LLM with no context, Sophea actions search the indexed knowledge base first, then pass the retrieved chunks as context to the LLM.

37.3.2 How it works

1. A flow step calls `ask_nous_knowledge` with a question
2. The broker endpoint builds signed ACL principals for the requesting user
3. The broker calls `run_batch_search()` on the RAG service, passing the signed ACL
4. The RAG service searches the vector index, filtering by ACL
5. Retrieved chunks are passed as context to the LLM
6. The LLM generates an answer grounded in the retrieved documents
7. The answer is returned to the flow step with citations

37.3.3 Available RAG actions

Action	Description
`ask_nous_knowledge`	Ask a question, get a grounded answer with citations
`search_knowledge`	Search the knowledge base, get raw chunks
`batch_search_knowledge`	Run multiple searches in one step
`fetch_chunks`	Fetch all chunks for a specific document

37.3.4 ACL enforcement in RAG

The RAG service enforces ACL at the query level. When a search request arrives:

1. The `X-Sophea-Principals` header is verified (HMAC signature)
2. The signed ACL is decoded: `{tid, uid, pr, exp}`
3. If the tenant ID or user ID does not match the request, the search returns 401
4. The vector search filters results to only include documents matching the ACL
5. If the ACL is expired or missing, the behavior depends on `REQUIRE_SIGNED_PRINCIPALS`:
   - `off` (default): fall back to body `acl_principals`
   - `warn`: log a warning, fall back to body `acl_principals`
   - `enforce`: return 401

37.4 Connector types

37.4.1 Sophea piece (native)

The Sophea piece provides 21 actions and 4 triggers that are native to the platform:

- **Knowledge**: `ask_nous_knowledge`, `search_knowledge`, `batch_search_knowledge`, `fetch_chunks`
- **AI**: `ask_workspace_model`, `run_nous_agent`

- **Agent lifecycle**: `start_agent_run`, `get_agent_run_status`, `cancel_agent_run`, `get_agent_run_artifacts`
- **Approvals**: `request_human_approval`, `create_approval_task`, `llm_approval`
- **Workspace**: `list_connectors`, `list_workspace_users`, `list_workspace_teams`, `fetch_saved_searches`, `notify_workspace`, `send_workspace_email`
- **Documents**: `upload_document`, `get_document_status`
- **Triggers**: `agent_run_completed`, `document_indexed`, `connector_sync_completed`, `connector_sync_failed`

37.4.2 Shell-mode allowlist

In shell mode, only vetted pieces are available (18 pieces total):

- `@activepieces/piece-sophea` (all Sophea actions and triggers)
- `@activepieces/piece-manual-trigger` (manual trigger)
- `@activepieces/piece-schedule` (time-based triggers)
- `@activepieces/piece-webhook` (HMAC-signed webhook trigger)
- `@activepieces/piece-tables` (data tables)
- `@activepieces/piece-forms` (forms and human input)
- `@activepieces/piece-file-helper` (file operations)
- `@activepieces/piece-csv` (CSV operations)
- `@activepieces/piece-data-mapper` (data transformation)
- `@activepieces/piece-data-summarizer` (data summarization)
- `@activepieces/piece-date-helper` (date utilities)
- `@activepieces/piece-delay` (delay step)
- `@activepieces/piece-math-helper` (math operations)
- `@activepieces/piece-pdf` (PDF operations)
- `@activepieces/piece-qrcode` (QR code generation)
- `@activepieces/piece-store` (key-value storage)
- `@activepieces/piece-text-helper` (text utilities)
- `@activepieces/piece-xml` (XML operations)

37.4.3 External connectors

External connectors (e.g., Slack, Google Drive, Confluence) are available outside shell mode. They are managed through the standard connector framework and respect tenant isolation at the connection level. Each connection is scoped to a workspace.

37.5 Security model

Layer	Mechanism
Transport	HMAC-signed HTTP between Automations API and Nous backend
Tenant	Workspace ID verified on every broker call
User	Runtime member verification (active workspace membership)
Data	Signed ACL principals propagated to RAG service
Execution	Sandboxed piece execution with vetted allowlist

Layer	Mechanism
Webhooks	HMAC signature verification on incoming webhooks

37.6 Related documentation

- How Sophea Automations Works
- Sophea vs Generic Automation Tools

Chapter 38

Sophea vs Generic Automation Tools

38.1 Overview

Generic automation tools like n8n and Zapier are designed for broad integration: connect hundreds of SaaS apps, route data between them, automate simple tasks. Sophea Automations is designed for a different problem: secure, AI-native workflows that operate on workspace knowledge with tenant isolation and permission enforcement.

This document compares the two approaches for enterprise B2B use cases.

38.2 Comparison

38.2.1 Security

Feature	Sophea Automations	Generic tools (n8n, Zapier)
Tenant isolation	Enforced on every request via HMAC-signed broker	Varies, often self-managed
ACL propagation	Signed principals propagated to RAG service	Not applicable
Webhook authentication	HMAC-signed, positive allowlist	Basic auth or none
Piece vetting	Shell mode allowlist restricts available pieces	All pieces available
Error detail leak prevention	Broker endpoints strip internal details	Varies
Audit trail	Every broker call logged with workspace, run, step	Limited

38.2.2 Multi-tenancy

Feature	Sophea Automations	Generic tools
Tenant model	Workspace equals tenant, native	Add-on or self-managed
Data isolation	Tenant-scoped database sessions	Varies
User isolation	Per-user ACL on knowledge base	Not applicable
Cross-tenant access	Impossible by design	Configuration-dependent

38.2.3 AI-native

Feature	Sophea Automations	Generic tools
Knowledge search	Native RAG pipeline with signed ACLs	External API call required
AI actions	21 built-in actions (knowledge, agents, approvals, documents)	Third-party plugins
Agent integration	Native agent-service lifecycle (start, status, cancel, artifacts)	Not available
Flow generation	AI generates flows from natural language	Limited or not available
LLM approval gates	Built-in `llm_approval` action with confidence thresholds	Custom or not available

38.2.4 RAG grounding

Feature	Sophea Automations	Generic tools
Knowledge base access	Direct, signed ACL principals	External API call
Document-level permissions	Enforced at query level	Not applicable
Citation support	Built-in, grounded answers with sources	Not available
Batch search	Native batch endpoint with RRF fusion	Not available
Chunk fetching	Native fetch-chunks endpoint	Not available

38.2.5 Template library

Feature	Sophea Automations	Generic tools
Domain templates	43 domain templates across 9 categories, plus 8 starter templates	Generic templates
Template categories	Knowledge, Reports, Research, Review, Approvals, Email, Data, Legal, HR	Generic
Setup hints	Each template lists setup requirements	Varies

Feature	Sophea Automations	Generic tools
AI-relevant	Templates use AI actions natively	Limited

38.3 When to choose Sophea Automations

Choose Sophea Automations when:

- You need workflows that search and reason over a workspace knowledge base
- You need tenant isolation enforced at the platform level, not the configuration level
- You need AI actions (knowledge search, agent runs, LLM approval gates) as first-class workflow steps
- You need audit trails for every automated action
- You need webhook triggers with HMAC authentication
- You need human-in-the-loop approval workflows integrated with AI

38.4 When generic tools may be sufficient

Generic tools may be sufficient when:

- You only need to route data between SaaS apps (no knowledge base)
- You do not need tenant isolation (single-tenant deployment)
- You do not need AI actions in workflows
- You need a connector that Sophea does not offer
- You are building simple, non-sensitive automations

38.5 Architecture comparison

38.5.1 Sophea Automations

```
Flow Step -> Sophea Piece -> Broker (HMAC) -> Nous Backend
                                           -> RAG Service (signed ACL)
                                           -> Agent Service
```

Every step goes through the broker. The broker enforces tenant isolation, verifies membership, and propagates ACLs. No step has direct database access.

38.5.2 Generic tools (n8n)

```
Flow Step -> Piece -> External API (direct)
                    Database (direct)
                    File system (direct)
```

Pieces have direct access to external systems. Security depends on the piece implementation and deployment configuration.

38.6 Conclusion

Sophea Automations trades breadth for depth. It does not try to compete on connector count. Instead, it provides a secure, AI-native automation platform where every action is auditable, every search is ACL-scoped, and every workflow is tenant-isolated by design.

For enterprise B2B teams that need to automate knowledge-intensive processes, this is the right trade-off.

38.7 Related documentation

- How Sophea Automations Works
- Enterprise Connectors
- Product positioning: `docs/product-06-sophea-automations.md`

Chapter 39

Automations Admin Guide

39.1 Overview

This guide covers how to enable and configure Sophea Automations for your workspace, including Portal setup, email configuration, and model routing.

39.2 Enabling Automations in Portal

39.2.1 Prerequisites

- Sophea platform deployed and running (Nous, Portal, RAG service)
- Portal admin access
- At least one workspace created

39.2.2 Steps

1. Log in to the Portal admin console
2. Navigate to the workspace settings
3. Enable the "Automations" feature toggle
4. The automations builder will be available at `/app/automations` in the Nous web UI

39.2.3 Shell mode

Shell mode restricts the available pieces to a vetted allowlist. This is recommended for production environments. To enable shell mode:

1. Set `NOUS_AUTOMATIONS_SHELL=true` in the Automations API server environment
2. Add `NOUS_AUTOMATIONS_SHELL` to `AP_SANDBOX_PROPAGATED_ENV_VARS` so pieces can read it inside the sandbox
3. Verify that only allowlisted pieces appear in the builder

The shell mode allowlist includes 18 vetted pieces: - `@activepieces/piece-sophea` (all Sophea actions and triggers) - `@activepieces/piece-manual-trigger` (manual trigger) - `@activepieces/piece-schedule` (time-based triggers) - `@activepieces/piece-webhook` (HMAC-signed webhook trigger) - `@activepieces/piece-tables` (data tables) - `@activepieces/piece-forms` (forms and human input) - `@`

activepieces/piece-file-helper (file operations) - @activepieces/piece-csv (CSV operations) - @activepieces/piece-data-mapper (data transformation) - @activepieces/piece-data-summarizer (data summarization) - @activepieces/piece-date-helper (date utilities) - @activepieces/piece-delay (delay step) - @activepieces/piece-math-helper (math operations) - @activepieces/piece-pdf (PDF operations) - @activepieces/piece-qrcode (QR code generation) - @activepieces/piece-store (key-value storage) - @activepieces/piece-text-helper (text utilities) - @activepieces/piece-xml (XML operations)

39.3 Configuring Workspace Email

Sophea Automations can send emails on behalf of workspace members. The send_workspace_email action uses the platform's email provider.

39.3.1 Prerequisites

- SMTP server configured in the Nous backend
- Workspace email settings enabled in Portal

39.3.2 Configuration

1. Configure the SMTP server in the Nous backend deployment
2. Set the following environment variables on the Nous backend:
    - SMTP_SERVER: SMTP server hostname
    - SMTP_PORT: SMTP server port (usually 587 for TLS)
    - SMTP_USER: SMTP username
    - SMTP_PASS: SMTP password
    - SMTP_TLS: whether to use TLS (defaults to enabled)
    - EMAIL_FROM: From email address (defaults to SMTP_USER)
3. Verify email delivery by running a flow with the send_workspace_email action

39.3.3 Email behavior

- Emails are sent as the workspace, not as individual users
- The recipientMode can be RUN_USER (send to the person who ran the flow) or WORKSPACE_MEMBERS (send to selected workspace members)
- Email body can be plain text or markdown (converted to HTML)

39.4 Setting Up Model Routing

Sophea Automations uses the same visible CHAT model and Workspace default configured for Nous. Runtime actions call the Nous broker directly. Generate and Refine use the Portal model proxy, but both surfaces enforce the same Workspace authority.

39.4.1 Workspace authority

1. A platform administrator makes W2, W3, and W4 available to the Workspace.
2. A Workspace administrator chooses one visible CHAT model as the default for that existing Workspace.

3. Account-level defaults affect future Workspace provisioning only, unless the administrator explicitly applies a change to existing Workspaces.
4. Hiding a model removes it from chat, action, Generate, and Refine selectors without physically deleting the catalog row.

39.4.2 Available models

The model selector lists `Workspace default` plus the visible explicit models available to the Workspace. `workspace_default` remains stored in the flow and resolves the live default every time the flow runs, so changing W3 to W4 updates unchanged flows without a restart.

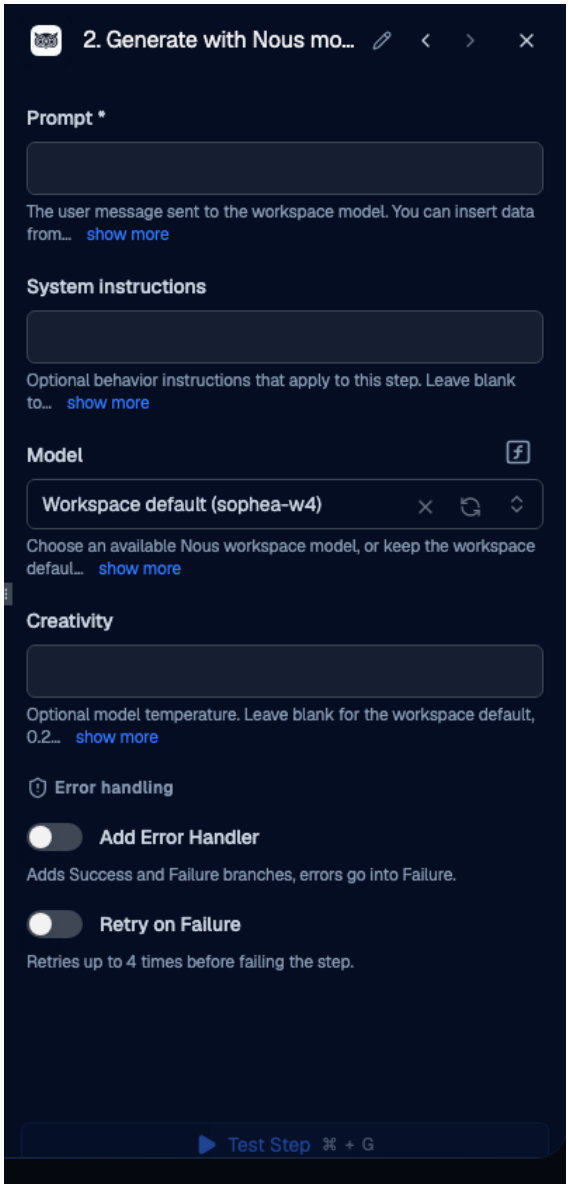

Figure 39.1: Automations action settings showing the live Workspace default model

A saved explicit model that is later hidden runs on the current Workspace default and records the requested model, executed model, and fallback reason in the run audit. A missing, deleted, malformed, non-CHAT, or different-Workspace model ID fails before a provider call.

Generate and Refine refresh the catalog when opened and again before submission. If an explicit selection disappears before the click completes, the operation stops, refreshes the selector, and waits for the user to choose or retry.

39.4.3 Model parameters

The `ask_workspace_model` action accepts: - `prompt`: The text prompt (required) - `systemPrompt`: System instructions (optional) - `modelId`: Model alias (default: `workspace_default`) - `temperature`: Sampling temperature (0-1, default varies by model)

39.5 Configuring Webhooks

39.5.1 HMAC-signed webhooks

The webhook trigger (`@activepieces/piece-webhook`) supports HMAC signature verification. In shell mode, only `HEADER` and `HMAC` auth types are allowed.

39.5.2 Setup

1. Add a webhook trigger to a flow
2. Set the auth type to `HMAC`
3. Configure the HMAC secret in the webhook piece settings
4. The webhook URL will be available at `/automations/webhooks/{flow-id}`
5. External systems must send the `X-Sophea-Signature` header with the HMAC-SHA256 signature

39.5.3 Verification

The webhook piece verifies: - The `X-Sophea-Signature` header matches the HMAC-SHA256 of the request body - The `X-Sophea-Timestamp` header is within the acceptable time window - In shell mode, `AuthType.NONE` and `AuthType.BASIC` are rejected

39.6 Managing Templates

39.6.1 Built-in templates

Sophea ships with 43 domain templates across 9 categories, plus 8 starter templates: - Knowledge (11) - Reports (11) - Research (10) - Review (8) - Approvals (5) - Email (4) - Data (3) - Legal (1) - HR (1)

Some templates appear in more than one category, so the per-category counts sum to more than 43.

The 8 starter templates cover common starting points: asking a workspace model, daily web research email, answering from indexed knowledge, daily knowledge digest email, structured extraction, content review approval, Excel-ready CSV report, and agent web research report.

39.6.2 Custom workspace templates

Workspace admins can create custom templates: 1. Build a flow in the builder 2. Save it as a template from the flow settings 3. The template will be available in the template gallery for all workspace members

39.6.3 Template categories

Templates are organized by category. The category carousel in the template gallery shows all available categories. Custom templates appear in the “My Templates” section.

39.7 Monitoring and Debugging

39.7.1 Run history

The run list sidebar shows all flow runs with: - Status (running, succeeded, failed, paused) - Duration - Start time - Error message (if failed)

39.7.2 Step inspection

Click any run to inspect step inputs, outputs, and errors: - Input tab: shows the input data for each step - Output tab: shows the output data and any error messages - Timeline tab: shows execution timeline for agent steps

39.7.3 Loading run data into tests

You can load run data into the test panel: 1. Open a run from the run list 2. Select the step you want to debug 3. Click “Load into test” to populate the test panel with the run’s step data 4. Modify the input and re-test the step

39.8 Related documentation

- User Guide
- How Sophea Automations Works
- Enterprise Connectors
- Local setup: `docs/tech-14-local-setup.md`

Chapter 40

Automations User Guide

40.1 Overview

This guide covers how to create, test, and publish automated workflows using Sophea Automations.

40.2 Getting Started

40.2.1 Opening the builder

1. Log in to the Sophea Nous web UI
2. Click “Automations” in the sidebar
3. Click “Create Flow” to open the builder

40.2.2 The builder interface

The builder has four main areas:

1. **Canvas**: The flow diagram showing triggers and actions
2. **Left sidebar**: Run history, flow settings, and flow list
3. **Right panel**: Step configuration (appears when a step is selected)
4. **Test panel**: Step testing and sample data (bottom right)

40.3 Creating Your First Flow

40.3.1 Step 1: Choose a trigger

Every flow starts with a trigger. Available triggers:

- **Manual Trigger**: Start the flow manually by clicking “Run”
- **Schedule**: Run the flow on a recurring schedule (daily, weekly, monthly)
- **Webhook**: Start the flow when an external system sends an HTTP request
- **Agent Run Completed**: Start the flow when a Nous agent finishes
- **Document Indexed**: Start the flow when a document completes indexing into the knowledge base

- **Connector Sync Completed**: Start the flow when a connector sync cycle completes successfully
- **Connector Sync Failed**: Start the flow when a connector sync cycle fails

To add a trigger: 1. Click the trigger node on the canvas 2. Select the trigger type from the dropdown 3. Configure the trigger settings in the right panel

40.3.2 Step 2: Add actions

Actions are the steps that run after the trigger. To add an action:

1. Click the "+" button below the trigger or any action
2. Browse the piece picker or search for a specific action
3. Select the action you want to add
4. Configure the action's inputs in the right panel

40.3.3 CSV and date defaults

Two common data actions start with supported defaults that you can change when your input requires a different format:

- **Convert CSV to JSON** sets **Delimiter Type** to **Comma**.

Figure 40.1: Convert CSV to JSON with Comma selected as the delimiter

- **Add/Subtract Time** sets both **From Time Format** and **To Time Format** to `DDD MMM DD YYYY HH :mm:ss (Sun Sep 17 2023 11:23:58)`.

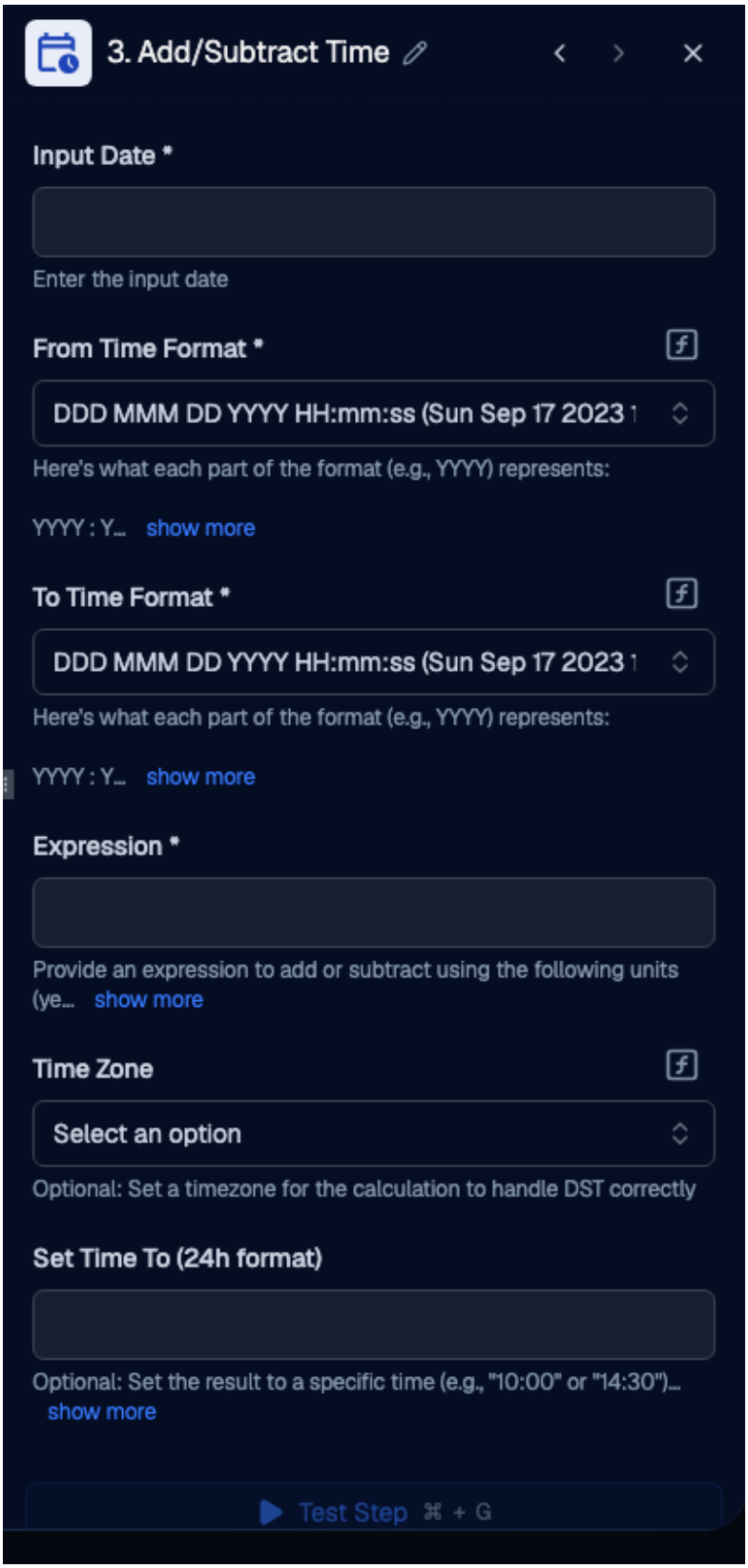

Figure 40.2: Add/Subtract Time with supported input and output format defaults

40.3.4 Available Sophea actions

Category	Actions
Knowledge	ask_nous_knowledge, search_knowledge, batch_search_knowledge, fetch_chunks
AI	ask_workspace_model, run_nous_agent
Agent lifecycle	start_agent_run, get_agent_run_status, cancel_agent_run, get_agent_run_artifacts
Approvals	request_human_approval, create_approval_task, llm_approval
Workspace	list_connectors, list_workspace_users, list_workspace_teams, fetch_saved_searches, notify_workspace, send_workspace_email
Documents	upload_document, get_document_status

40.3.5 Step 3: Reference previous steps

You can reference the output of previous steps using the `{stepName.output}` syntax. For example: - `{trigger.output.body.email}` to reference the email from a webhook trigger - `{step_1.output.answer}` to reference the answer from a knowledge search - `{step_1.output.chunks[0].content}` to reference the first chunk from a search

40.3.6 Step 4: Test the flow

Before publishing, test each step:

1. Select the step you want to test
2. Click “Test Step” in the test panel
3. The step will execute and show the output
4. Use the output to configure subsequent steps

40.3.7 Step 5: Publish the flow

1. Click “Publish” in the top right
2. The flow is now active and will run when the trigger fires
3. Published flows can be viewed in the “Runs” tab

40.4 Using Templates

40.4.1 Browsing templates

1. Click “Templates” in the sidebar
2. Browse by category using the category carousel
3. Click any template to see a preview

40.4.2 Template categories

Category	Count	Examples
Knowledge	11	FAQ answer, gap analysis, policy Q&A
Reports	11	Executive summary, postmortem, release notes
Research	10	Competitor analysis, market intelligence, regulatory monitoring
Review	8	Content review, code review, vendor evaluation
Approvals	5	Change request review, security review gate
Email	4	Alert notification, weekly knowledge summary
Data	3	Data enrichment, record summarization, deduplication
Legal	1	Legal clause summarizer
HR	1	Job description generator

Some templates appear in more than one category, so the per-category counts sum to more than 43.

40.4.3 Creating a flow from a template

1. Browse the template gallery
2. Click a template that fits your use case
3. Review the template description and setup requirements
4. Click “Use Template” to create a new flow
5. Configure any required inputs (API keys, email recipients, etc.)
6. Test and publish

40.4.4 Saving custom templates

1. Build a flow in the builder
2. Click “Save as Template” in the flow settings
3. Give the template a name and description
4. The template will be available in “My Templates” for your workspace

40.5 Testing and Debugging

40.5.1 Testing individual steps

1. Select a step on the canvas
2. Click “Test Step” in the test panel
3. The step executes and displays the output
4. Use the output to configure the next step

40.5.2 Loading run data into tests

If a step failed in a previous run, you can load the run data into the test panel:

1. Open the run from the run list sidebar
2. Select the step that failed
3. Click “Load into test” to populate the test panel with the run’s input data
4. Modify the input and re-test

40.5.3 Viewing run history

1. Click the “Runs” icon in the left sidebar
2. Filter by status (all, running, succeeded, failed, paused)
3. Click any run to inspect step inputs and outputs
4. Use the error search to find runs with specific error messages

40.5.4 Cancelling runs

- Running runs cannot be cancelled (they must complete or time out)
- Queued runs can be cancelled from the run list
- Paused runs (waiting for approval) can be cancelled

40.6 Using AI to Generate Flows

40.6.1 AI flow generation

Instead of building a flow manually, you can describe what you want in natural language:

1. Click **Generate with AI** in the builder.
2. Choose **Workspace default** or a visible explicit model.
3. Describe the workflow you want to create.
4. The AI generates a flow with appropriate triggers and actions.
5. Review the generated flow, test it, and publish.

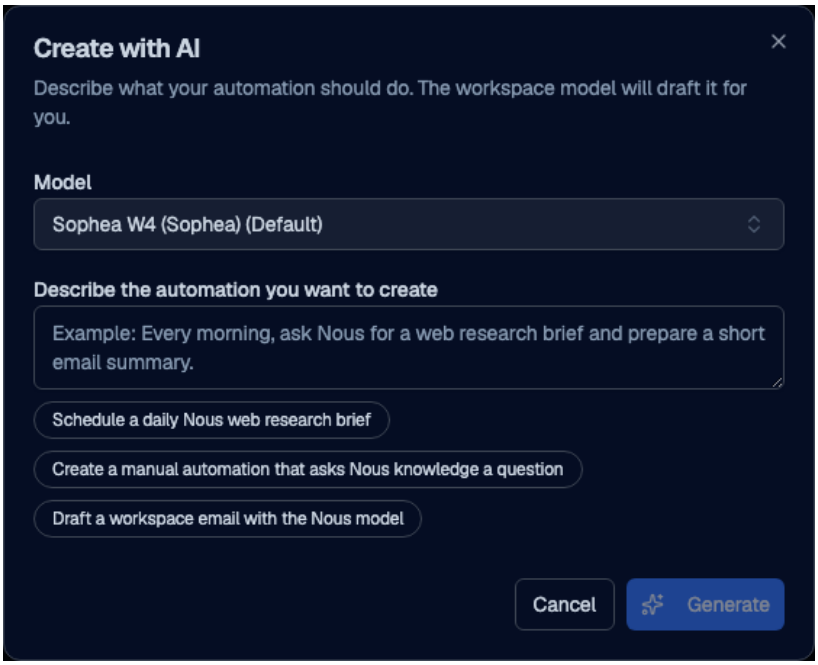

Figure 40.3: Create with AI dialog using Sophea W4 as the Workspace default

40.6.2 Tips for AI flow generation

- Be specific about the trigger (e.g., “Run daily at 9 AM” or “When a webhook is received”)
- Mention which Sophea actions you want (e.g., “Search the knowledge base and send an email”)
- Include the expected input and output format
- The AI uses the Sophea piece actions, so describe the workflow in terms of knowledge search, AI model, email, etc.

40.6.3 Refining a generated flow

After the AI generates a flow, you can iterate on it with follow-up instructions instead of starting over:

1. Click **Refine with AI** in the flow builder header.
2. Describe the change you want, for example `Add an email step after the knowledge search` or `Change the schedule to weekly`.
3. Confirm **Workspace default** or choose a visible explicit model, then click **Refine**.
4. Review the updated flow and continue refining or publish.

You can refine multiple times. Each refinement builds on the current state of the flow, so you do not need to restart when you want to make adjustments.

Generate and Refine refresh the available models when opened and again when you submit. If your selected model was hidden or removed in between, Automations stops the request and refreshes the selector. Review the new selection and click again; it does not change the model silently.

40.7 Choosing models in workflow steps

Model-capable Nous actions offer **Workspace default** and the visible explicit models available to your Workspace.

- **Workspace default** is recommended for most flows. The saved flow follows the live Workspace default on every run.
- An **explicit model** remains pinned while that model is visible. If an administrator later hides it, the saved flow uses the current Workspace default and records the requested model, executed model, and fallback reason in the run details.
- A deleted, malformed, or different-Workspace model ID stops the step safely. Open the step, refresh the model list, and make an explicit selection before retrying.

40.8 Approvals

40.8.1 Human-in-the-loop approvals

The `request_human_approval` action pauses the flow and waits for a human to approve or reject:

1. Add the `request_human_approval` action after an AI step
2. Configure the approval message and assignee
3. The flow pauses when it reaches this step
4. The assigned user receives an email with a link to approve or reject
5. If approved, the flow continues. If rejected, the flow stops.

40.8.2 LLM approval gates

The `llm_approval` action uses an AI model to approve or reject based on criteria:

1. Add the `llm_approval` action
2. Set the `criteria` (e.g., "Approve if the content is professional and accurate")
3. Set the `subject` (the content to evaluate)
4. Set the `confidenceThreshold` (0-1, default 0.8)
5. The AI evaluates the subject against the criteria and returns approved/rejected

40.8.3 Approval workbench

The approval workbench shows all pending approval tasks: 1. Click "Approvals" in the sidebar 2. View pending, approved, and rejected tasks 3. Click any task to approve or reject

40.9 Email

40.9.1 Sending workspace emails

The `send_workspace_email` action sends an email to workspace members:

1. Add the `send_workspace_email` action
2. Set `recipientMode` to `RUN_USER` (send to the flow runner) or `WORKSPACE_MEMBERS` (selected workspace members)
3. If `WORKSPACE_MEMBERS`, select the recipients from the workspace member list

4. Set the subject and body
5. Set `bodyFormat` to `markdown` or `plain_text`

40.9.2 Common email patterns

- **Daily brief**: Schedule trigger + ask_workspace_model + send_workspace_email
- **Alert notification**: Webhook trigger + ask_workspace_model + send_workspace_email
- **Weekly summary**: Schedule trigger + ask_nous_knowledge + send_workspace_email

40.10 Related documentation

- Admin Guide
- How Sophea Automations Works
- Product positioning: `docs/product-06-sophea-automations.md`

Chapter 41

Sample Workspace and Demo Flows

41.1 Overview

This guide describes how to set up a sample workspace with pre-configured knowledge, agents, and email so new users can try Sophea Automations immediately.

41.2 Sample Workspace Configuration

41.2.1 Prerequisites

- Sophea platform running locally or on staging
- At least one workspace with knowledge indexed
- Workspace email configured (see Admin Guide)
- At least one Nous agent configured (for agent-related flows)

41.2.2 Recommended setup

For a complete demo experience, configure the following:

1. **Knowledge base**: Index at least 10 documents covering:
    - Company policies (remote work, expense, security)
    - Product documentation (API guides, user manuals)
    - Internal wiki (team processes, onboarding checklists)
2. **AI model**: Ensure `workspace_default` model alias is configured in LiteLLM
3. **Agent**: Configure a web-search agent for research flows
4. **Email**: Configure SMTP so email-sending flows work end-to-end

41.3 Demo Flows

41.3.1 Demo 1: Create your first flow (5 minutes)

Goal: Search the knowledge base and display the answer.

1. Open the automations builder
2. Create a new flow with a Manual Trigger
3. Add the `ask_nous_knowledge` action
4. Set the question to: "What is our remote work policy?"
5. Test the step
6. Publish the flow
7. Run the flow manually

What this demonstrates: Knowledge search, AI grounding, citation support.

41.3.2 Demo 2: Generate a flow with AI (3 minutes)

Goal: Use AI to generate a flow from a description.

1. Open the automations builder
2. Click "Generate with AI"
3. Enter: "Every day at 9 AM, search the knowledge base for recent updates and email a summary to the flow runner"
4. Review the generated flow
5. Test each step
6. Publish the flow

What this demonstrates: AI flow generation, schedule triggers, knowledge search, email.

41.3.3 Demo 3: Set up an approval workflow (5 minutes)

Goal: Use AI to review content before publishing.

1. Open the automations builder
2. Create a new flow with a Manual Trigger
3. Add the `llm_approval` action
4. Set the criteria to: "Approve if the content is professional, accurate, and free of sensitive data"
5. Set the subject to a draft blog post or marketing copy
6. Set the confidence threshold to 0.8
7. Test the step
8. Add a `send_workspace_email` action after the approval
9. Configure the email to notify the runner of the decision
10. Publish and run

What this demonstrates: LLM approval gates, conditional flow, email notification.

41.3.4 Demo 4: Run a research agent (5 minutes)

Goal: Use a Nous agent to research a topic and summarize results.

1. Open the automations builder
2. Create a new flow with a Manual Trigger
3. Add the `start_agent_run` action
4. Set the agentId to your web-search agent
5. Set the message to: "Research the top 3 competitors in enterprise AI platforms"
6. Add the `get_agent_run_artifacts` action
7. Add the `send_workspace_email` action to email the results
8. Test each step

9. Publish and run

What this demonstrates: Agent lifecycle, async execution, artifact retrieval, email.

41.3.5 Demo 5: Batch knowledge search (3 minutes)

Goal: Search multiple queries in one step.

1. Open the automations builder
2. Create a new flow with a Manual Trigger
3. Add the `batch_search_knowledge` action
4. Set the queries to: ["current sprint goals", "open customer issues", "recent deployment notes"]
5. Set the limit to 5
6. Test the step
7. Add the `ask_workspace_model` action to summarize the results
8. Publish and run

What this demonstrates: Batch search, multi-query RAG, AI summarization.

41.4 Template Gallery Tour

The template gallery has 43 domain templates across 9 categories, plus 8 starter templates. For a quick tour:

1. **Knowledge**: Try the "FAQ answer generator" template
2. **Approvals**: Try the "Security review gate" template
3. **Research**: Try the "Competitor analysis" template
4. **Reports**: Try the "Executive summary generator" template
5. **Email**: Try the "Weekly knowledge summary" template

Each template lists setup requirements so you know what needs to be configured before use.

41.5 Troubleshooting

41.5.1 Flow does not start

- Check that the trigger is configured correctly
- For schedule triggers, verify the timezone and time
- For webhook triggers, verify the URL and HMAC configuration
- Check the run history for error messages

41.5.2 Knowledge search returns no results

- Verify that documents are indexed in the knowledge base
- Check that the user running the flow has access to the documents
- Try a broader search query
- Check the RAG service logs for errors

41.5.3 Email not sent

- Verify SMTP configuration in the Nous backend
- Check that the recipient is a workspace member
- Verify the email body is not empty
- Check the Nous backend logs for SMTP errors

41.5.4 AI model returns an error

- Verify the LiteLLM proxy is running
- Check that the model alias is configured
- Verify the model is available and not rate-limited
- Check the LiteLLM logs for circuit breaker trips

41.6 Related documentation

- Admin Guide
- User Guide
- How Sophea Automations Works
- Local setup: `docs/tech-14-local-setup.md`