



**Agent**Plus

## USER GUIDE

---

Version 2.2.0 · October 2026

# Contents

---

|                                                  |    |
|--------------------------------------------------|----|
| <b>Introduction</b>                              | 3  |
| <b>Application Setup</b>                         | 4  |
| Prerequisites and Installation                   | 4  |
| Security and Permissions                         | 7  |
| Configuration                                    | 8  |
| Agent Activation                                 | 14 |
| <b>Using AgentPlus</b>                           | 18 |
| Smart Navigation                                 | 18 |
| Visual Charts                                    | 18 |
| Data Tables                                      | 19 |
| AgentPlus Voice                                  | 21 |
| Open Lightning Web Component                     | 25 |
| <b>Troubleshooting</b>                           | 29 |
| Is It an Agentforce Issue or an AgentPlus Issue? | 29 |
| Common Issues                                    | 29 |
| Contact Support                                  | 30 |

# Introduction

---

AgentPlus is a comprehensive suite of enhancements designed to extend the core capabilities of Salesforce Agentforce, with a primary focus on improving speed, context, and natural interaction for users. By putting the agent's answers on screen — navigation, charts, tables and your own components — and adding voice, AgentPlus ensures a more intuitive and efficient experience when interacting with AI agents.

The application is built around six key pillars:

- **Smart Navigation:** Seamlessly move between applications and records.
- **Visual Charts:** Generate and filter data visualizations through chat or voice.
- **Data Tables:** Show filterable, sortable, paginated tables of the underlying records.
- **AgentPlus Voice:** A voice-first conversational interface using native browser technology.
- **Open LWC:** Open any Lightning Web Component dynamically with agent-provided parameters.

The AgentPlus user interface is available in eight languages: English, German, Spanish, French, Italian, Japanese, Korean, and Brazilian Portuguese.

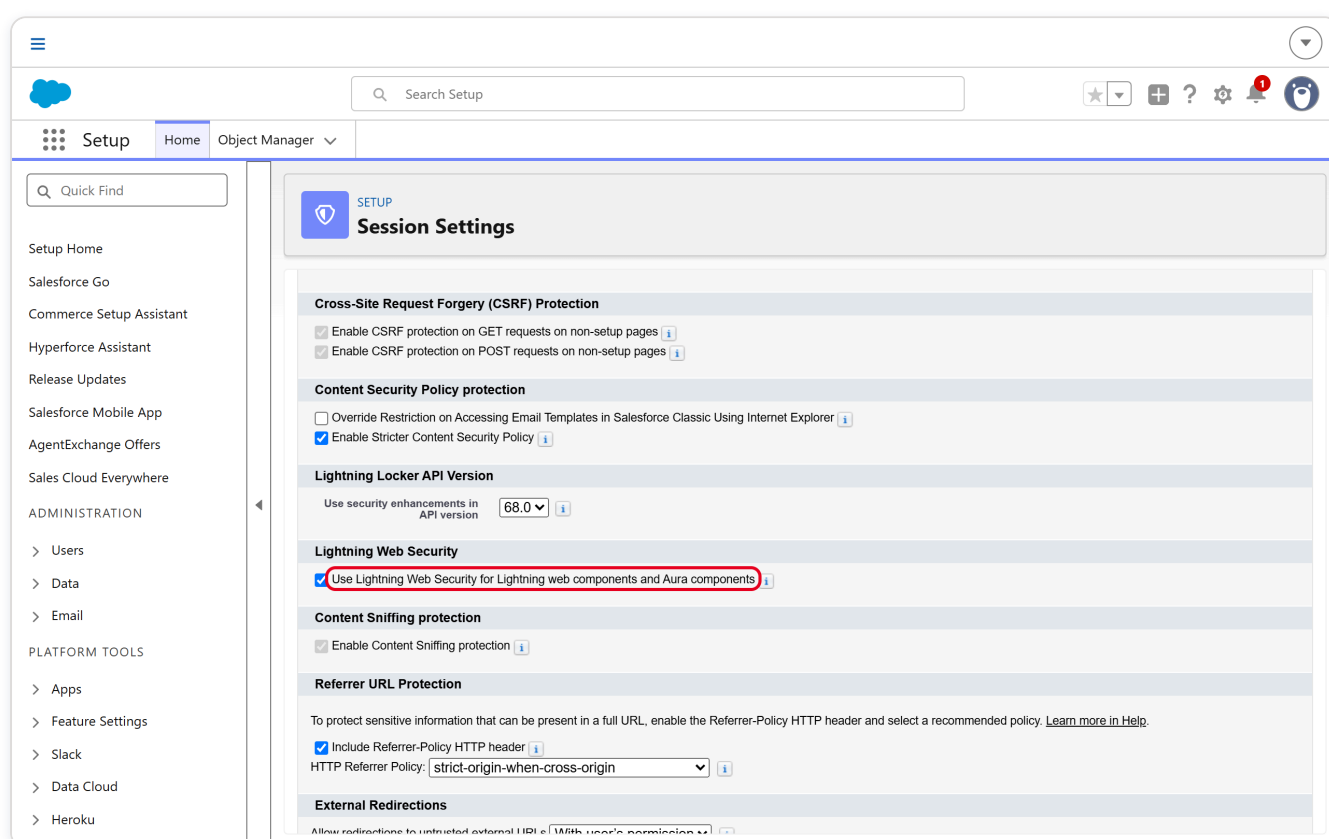
AgentPlus works in the utility bar of desktop Lightning apps, on Lightning pages of your own, in the Salesforce mobile app, and on Experience Cloud sites.

# Application Setup

## Prerequisites and Installation

To begin using AgentPlus, ensure your Salesforce organization has Agentforce enabled.

- **Install the application:** Go to the AppExchange listing to install AgentPlus in a sandbox or production org.
- **Agentforce enablement:** Make sure Agentforce is enabled for your org.
- **Enable Lightning Web Security (LWS):** This is a mandatory requirement for the components to function correctly. In Setup, go to **Session Settings** and confirm that **Use Lightning Web Security for Lightning web components and Aura components** is checked. See [Lightning Web Security](#) for details.



Setup — Session Settings with the Lightning Web Security option checked

## Utility Bar Configuration

Add the **AgentPlus Companion** component to the utility bar of every Lightning app where you want to use AgentPlus. The Companion contains AgentPlus Voice and shows the agent's actions on screen, so it is required for the Navigation, Charts, Data Tables, and Open LWC features, and it is the one component you place for AgentPlus Voice.

1. Navigate to **Setup > App Manager** and edit the chosen app.

SETUP  
Lightning Experience App Manager

23 items • Sorted by App Name • Filtered by All app menu items - TabSet Type, App Type

|    | App Name ↑                      | Developer Name        | Description                       | Last Modified ...     | App Type            | Vi... |   |
|----|---------------------------------|-----------------------|-----------------------------------|-----------------------|---------------------|-------|---|
| 1  | Agentforce Grid                 | AIWorkbench           | Access Agentforce Grid            | 10/15/2026, 2:40 A... | Lightning           | ✓     | ▼ |
| 2  | Agentforce Studio               | AgentforceStudio      | Agentforce Studio                 | 10/15/2026, 2:40 A... | Lightning           | ✓     | ▼ |
| 3  | AgentPlus                       | AgentPlus             | AgentPlus administration cons...  | 10/15/2026, 2:42 A... | Lightning (Managed) | ✓     | ▼ |
| 4  | AgentPlus Console               | AgentPlus_Console     | Console-navigation demo app ...   | 10/15/2026, 2:45 A... | Lightning           | ✓     | ▼ |
| 5  | AgentPlus Voice                 | agentplus voice       |                                   | 10/15/2026, 2:48 A... | Community           |       | ▼ |
| 6  | All Tabs                        | AllTabSet             |                                   | 10/9/2026, 6:27 AM    | Classic             |       | ▼ |
| 7  | App Launcher                    | AppLauncher           | App Launcher tabs                 | 10/9/2026, 6:27 AM    | Classic             | ✓     | ▼ |
| 8  | Approvals                       | Approvals             | Manage approvals and approv...    | 10/9/2026, 6:27 AM    | Lightning           | ✓     | ▼ |
| 9  | Bolt Solutions                  | LightningBolt         | Discover and manage business...   | 10/9/2026, 6:27 AM    | Lightning           | ✓     | ▼ |
| 10 | Community                       | Community             | Salesforce CRM Communities        | 10/9/2026, 6:27 AM    | Classic             | ✓     | ▼ |
| 11 | Content                         | Content               | Salesforce CRM Content            | 10/9/2026, 6:27 AM    | Classic             | ✓     | ▼ |
| 12 | CPQ Integration User Connect... | CPQIntegrationUserApp |                                   | 10/9/2026, 6:27 AM    | Connected           |       | ▼ |
| 13 | Digital Experiences             | SalesforceCMS         | Manage content and media fo...    | 10/9/2026, 6:27 AM    | Lightning           | ✓     | ▼ |
| 14 | Marketing CRM Classic           | Marketing             | Track sales and marketing effo... | 10/9/2026, 6:27 AM    | Classic             | ✓     | ▼ |
| 15 | McEvoe Journey                  | McEvoe                | Discover your customer's jour...  | 10/9/2026, 6:27 AM    | Lightning           | ✓     | ▼ |

App Manager — opening the edit menu on a Lightning app

2. In **App Settings > Utility Items**, click **Add Utility Item** and select **AgentPlus Companion** (under Custom - Managed).

Lightning App Builder | App Settings | Pages | Approvals | Help

**App Settings**

- App Details & Branding
- App Options
- Utility Items (Desktop Only)**
- Navigation Items
- Navigation Rules
- User Profiles

Utility Items (Desktop Only)

Give your users quick access to productivity tools and add background utility items to your app.

Add Utility Item

Utility Bar Alignment ⓘ Default

that opens components in docked panels. Available only when the app is viewed in Lightning Experience on a desktop.

Search: AgentPlus

- Standard (0)
- Custom (0)
- Custom - Managed (1)
  - AgentPlus Companion**

To enable the utility bar for this app, add a utility item.

Utility Items — adding the AgentPlus Companion component

3. Adjust the utility item properties. Make sure **Start automatically** is checked so the Companion is active as soon as the app loads. Our suggestion is to use the `robot` icon and a 340 × 450 panel size. Optionally set **Restrict to Agent** to lock this utility item to a single agent (see [Agent Selection](#)).

The screenshot shows the 'App Settings' interface in Lightning App Builder. The left sidebar lists 'App Settings' with sub-items: 'App Details & Branding', 'App Options', 'Utility Items (Desktop Only)' (selected), 'Navigation Items', 'Navigation Rules', and 'User Profiles'. The main area is titled 'Utility Items (Desktop Only)' and includes a description: 'Give your users quick access to productivity tools and add background utility items to your app.' There is an 'Add Utility Item' button and a 'Utility Bar Alignment' dropdown set to 'Default'. A preview of the 'AgentPlus Companion' utility item is shown on the left. To its right, the 'PROPERTIES' panel for 'AgentPlus Companion' is displayed. It has 'Utility Item Properties' and 'Component Properties' sections. In 'Utility Item Properties', the 'Label' is 'AgentPlus', the 'Icon' is 'robot', 'Panel Width' is '340', 'Panel Height' is '450', and the 'Start automatically' checkbox is checked and highlighted with a red rectangle. The 'Restrict to Agent' field is empty in the 'Component Properties' section.

Utility item properties — label, icon, panel size, Start automatically and Restrict to Agent

In a console app, the agent's charts, tables and components open in workspace tabs. In other apps, they open as pages of the app — unless a chart, table or component of the same kind is already on screen, in which case the agent updates it in place.

## Page Placement

You can also place the **AgentPlus Companion** on an app, record or home page in Lightning App Builder. The Companion then is the conversation: it fills its place on the page, and the agent's results are shown differently from the utility bar:

- Charts, tables and components open in a panel over the conversation, which you can minimize to its title bar or close without interrupting the conversation.
- A page the agent takes you to (a record, a list view, an app) replaces the conversation, so the Companion asks first.

The Salesforce mobile app has no utility bar, so it uses a page placement; the package includes one (see [AgentPlus Voice in the Salesforce Mobile App](#)). Set **Restrict to Agent** on the placement to tie the page to a single agent (see [Agent Selection](#)). Users still need the **AgentPlus User** permission set and a custom

permission that approves a voice agent (see [Voice Agents](#)).

## Experience Cloud Sites

An Experience Cloud site can offer AgentPlus to your internal users in two ways:

- **On a page:** The conversation fills a page of its own, like any other page placement. A page the agent opens replaces the conversation, after asking (see [Leaving the Conversation](#)).
- **Docked:** The conversation floats over every page of the site from a launcher at the bottom right, the AgentPlus orb in a white disc. It stays open while the user moves between pages, so the agent can take the user anywhere on the site without interrupting the conversation.

To set it up:

1. In **Experience Builder**, open the page that should host the conversation, or, for the docked placement, the theme layout's header or footer, so the conversation is part of every page that uses the layout.
2. Drag the **AgentPlus Companion** component (under Custom - Managed) into it.
3. For the docked placement, select **Docked**. Optionally set **Restrict to Agent** (see [Agent Selection](#)).
4. Click **Publish**. A site serves the component as it was when you last published, so publish again after any package update.
5. Make sure the users who will use it hold the **AgentPlus User** permission set and a custom permission that approves a voice agent (see [Voice Agents](#)).

Charts, tables and components open in a panel over the conversation, and they appear once the agent has finished its reply; the docked window widens to show them. A record or list the agent opens needs a page on the site that can show it: create the object pages for the objects your agent works with, or the site shows its error page.

**Note:** Only internal users are supported. AgentPlus Voice talks to the employee agents you approve in the [Voice Agents](#) card, and employee agents serve internal users, not the customers or partners who sign in to a site with an external license.

## Security and Permissions

Proper access control is essential for AgentPlus to interact with Salesforce data.

- **Users:** Assign the **AgentPlus User** permission set to all users who require access to the tool.
- **Administrators:** Assign the **AgentPlus Admin** permission set to the administrators who configure AgentPlus. It grants the AgentPlus app and admin tab, plus the custom permission required by the admin console.
- **Apex permissions:** Ensure the agent user has explicit permissions for all AgentPlus Apex classes so the agent can execute actions.

AgentPlus actions strictly enforce existing user permissions and record visibility.

## Configuration

### AgentPlus Admin Console

The AgentPlus app includes an administration console that centralizes the package configuration. To access it:

1. Make sure your user has the **AgentPlus Admin** permission set (see [Security and Permissions](#)).
2. Open the **AgentPlus** app from the App Launcher, or navigate to the **AgentPlus Admin** tab.

Users without the required permissions see an access denied message on this page.

The screenshot shows the AgentPlus Admin Console interface. At the top, there's a search bar and navigation icons. The main header shows 'AgentPlus' and 'AgentPlus Admin'. Below this, the 'Voice Agents' section is visible, with a description: 'Approve the agents offered by the Voice component, each gated by a custom permission. Users are offered an agent when they hold any of its permissions; the same agent can be approved under several permissions.' It includes a form to 'Add agent' with a dropdown for 'Select an agent' and a dropdown for 'Custom Permission' with a '+ Add' button. Below the form is a table with columns: Agent, Custom Permission, Speech, and Enabled. The table contains one row for 'AgentPlus Demo Agent' with 'AgentPlus Admin' as the custom permission and 'Recognition: Browser · Voice: Browser' as the speech configuration. The 'Enabled' column has a toggle switch set to 'Enabled'. Below this is the 'Lightning Web Components Access' section, with a '+ New' button. It has a description: 'Control which Lightning Web Components an agent can open. Each component is paired with a Custom Permission, and only users who hold that permission can open it.' It contains a table with columns: Title, Definition Name, LWC API Name, Agent Instructions, Custom Permission, and Enabled. The table has two rows: 'Account Summary' with 'accountSummary' as the definition name, 'c\_\_accountSummary' as the LWC API name, and 'AgentPlus Admin' as the custom permission; and 'Demo Greeting' with 'agpls\_demoCustomLwc' as the definition name, 'c\_demoCustomLwc' as the LWC API name, and 'AgentPlus Admin' as the custom permission. Both rows have 'Enabled' toggle switches.

AgentPlus Admin Console — Voice Agents and Lightning Web Components Access

### Voice Agents

The **Voice Agents** card manages the agents that appear in the Voice interface's agent selector, and controls which users can access each one.

To approve an agent:

1. Choose an agent from the **Agent** dropdown. The dropdown lists every internal employee agent in your org — including agents that are not yet active, which are marked as such, so you can approve an agent ahead of its activation. Users are only offered an approved agent once it is active.
2. Choose a [custom permission](#) from the **Custom Permission** dropdown.

3. Click **Add** to create the approval record.

Each approval record links one agent to one custom permission — a user is offered the agent when they hold that permission. To make the same agent available to users under a different permission, add a second record for the same agent with a different permission: a user holding any of an agent's gating permissions is offered it.

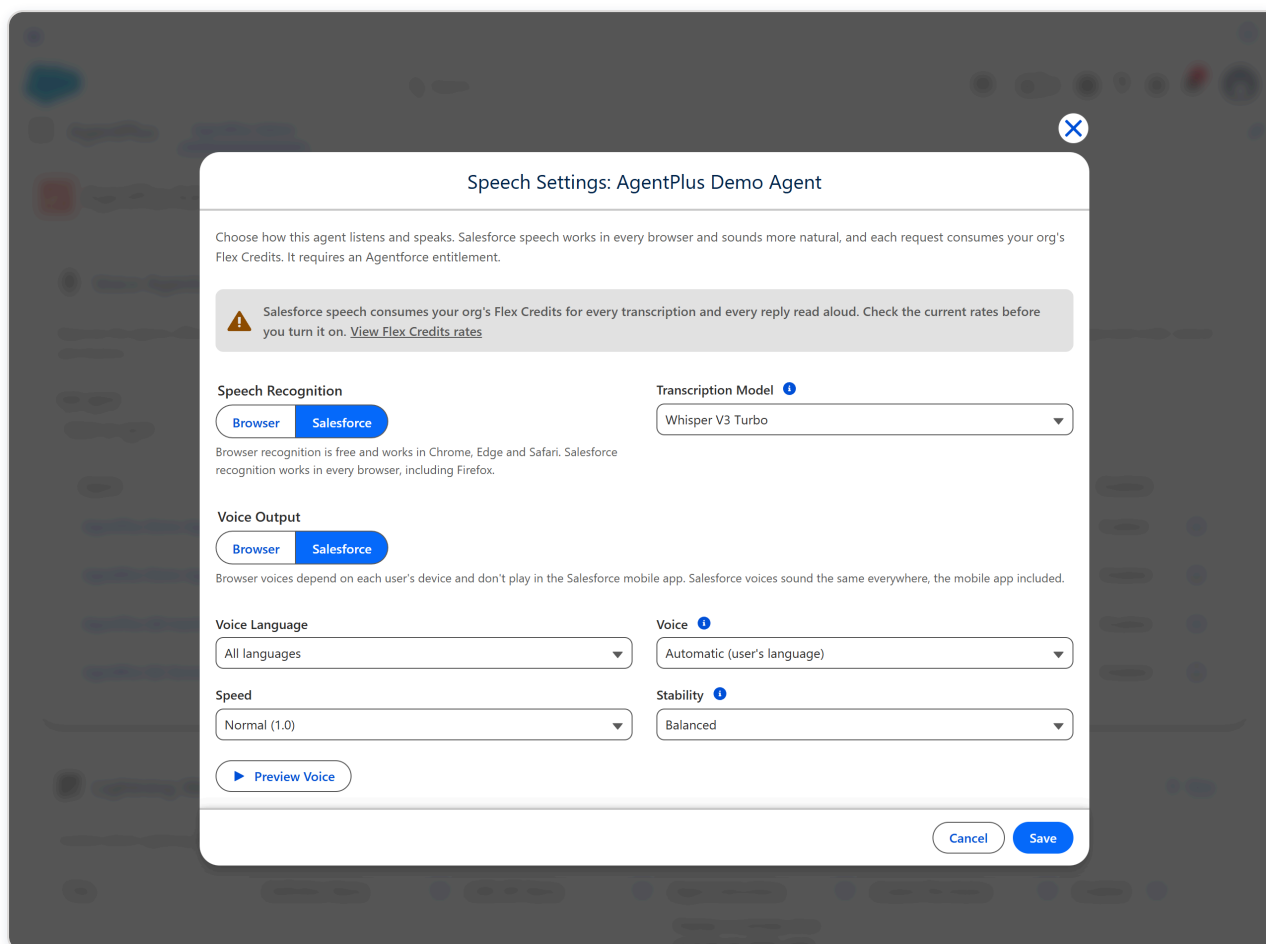
Each approval record appears in the table with the agent, the gating permission, and an **Enabled** column:

- Use the row menu to **Disable** a record (the agent remains in the table, but that permission no longer grants access) or **Enable** it again.
- The **Enabled** column describes the approval record, not the agent itself. An approved agent that is not currently active in Agentforce is flagged separately next to its name — an enabled record for an inactive agent still offers nobody anything until the agent is activated.
- Approval records are never deleted from this page, so a disabled one can be enabled again at any time.
- Adding an agent and permission pair that already exists re-enables the existing record instead of creating a duplicate.

### Voice Agent Speech Settings

Each agent listens and speaks with the browser's built-in speech until you choose otherwise. The **Speech** column shows each agent's engines. To change them, click the agent's name, or open the row menu and click **Edit Speech Settings**:

1. Under **Speech Recognition**, choose **Browser** or **Salesforce**. Browser recognition is free and works in Chrome, Edge and Safari; Salesforce recognition works in every browser, including Firefox. With Salesforce, choose the **Transcription Model**: Whisper V3 Turbo runs in Salesforce in your org's region; ElevenLabs Scribe v2 is a third-party model.
2. Under **Voice Output**, choose **Browser** or **Salesforce**. With Salesforce, pick a **Voice** (filter the list with **Voice Language**), a **Speed** and a **Stability**. **Automatic (user's language)** reads each reply in a voice of each user's own language. Choose Salesforce for any agent used from the Salesforce mobile app, where browser voices do not play. Click **Preview Voice** to hear it: the sample sentence is read in the voice's own language when AgentPlus is translated into it, and in your language otherwise.



Speech Settings — Salesforce recognition and voice, with the Flex Credits warning

3. Click **Save**. Settings apply to every approval record of the agent.

As soon as either engine is set to Salesforce, the window shows a warning with a link to the current Flex Credits rates in Salesforce Help.

**Important:** Salesforce speech consumes your org's Flex Credits for every transcription and every reply read aloud, and so does each voice preview. The rate for each usage type is published by Salesforce and changes between releases: check the [Flex Credits usage and rates](#) in Salesforce Help before you turn it on, and follow consumption in your org's Digital Wallet.

Salesforce speech requires an Agentforce entitlement and is not available in Government Cloud. Speech recognition and voice output are set separately, so you can, for example, keep free browser recognition and use a Salesforce voice.

### Lightning Web Components Access

The **Lightning Web Components Access** card manages which Lightning Web Components the agent is allowed to open, and which custom permission gates each one.

To register a component:

1. Click **New**.

2. Enter a **Title**, up to 40 characters. This is the name the agent uses for the component when it talks to the user — "Account Summary" rather than `accountSummary` . Write it the way it should be read aloud; the long explanation belongs in **Agent Instructions**.
3. Set the **Definition Name**. This is the identifier an agent script uses to open the setting directly (see [Wiring Open LWC into an agent](#)), and it is fixed once the setting is saved.
4. Enter the LWC API name and the **Agent Instructions**. The agent reads the instructions to decide whether and when to open the component, so use them to explain what it does and when to use it.
5. Select the gating [custom permission](#).
6. Choose how the component occupies tabs with **Update Existing Tab** (see below).
7. Optionally declare the input parameters the component accepts (see below).
8. Click **Add**.

The screenshot shows the 'Edit LWC Access' dialog box. It has a title bar with a close button. The form includes the following fields and controls:

- \* Title**: A text input field containing 'Account Summary'.
- Definition Name**: A text input field containing 'accountSummary'.
- \* LWC API Name**: A text input field containing 'c\_\_accountSummary'.
- \* Custom Permission**: A dropdown menu showing 'AgentPlus Admin (AgentPlusAdmin)'.
- Agent Instructions**: A text area containing 'Displays a summary of an account: key fields, open opportunities and recent activity. Open it when the user asks for an account overview.'
- Update Existing Tab**: A checked checkbox.
- Input Parameters**: A section with a '+ Add Parameter' button and a description: 'Declare the inputs this component accepts so the agent knows what to pass. Each parameter maps to an @api property on the component.'
- Enabled**: A checked toggle switch.
- Tab Key**: An unchecked checkbox.
- Parameter Name**: A text input field containing 'mode'.
- Type**: A dropdown menu showing 'Text'.
- Required**: A checked toggle switch.
- Agent Instructions**: A text area (partially visible).
- Buttons**: 'Cancel' and 'Save' buttons at the bottom right.

Lightning Web Components Access — component editor with parameters

To change a registered component, click its title in the table, or choose **Edit** from the row's menu. The same menu can **Clone** a setting or enable and disable it.

The API name may include a namespace prefix: a plain `accountSummary` (or the equivalent `c__accountSummary` ) refers to a component in your own org, while `ns__accountSummary` refers to one from an installed package. This lets you grant access to a specific component even when two share the same name.

The API name is a technical detail and never reaches the agent — it is the title the agent is given, so the title is what the user hears. Because a title belongs to the setting rather than to the component, registering one component twice lets you present it under two names, each describing what that mode is for.

The title is also the setting's record name under **Setup > Custom Metadata Types > AgentPlus LWC Access > Manage Records**, so a setting is called the same thing wherever you meet it. Renaming it in either place renames it in both. The **Definition Name** column in the table shows each setting's identifier, which is what an agent script needs to open that setting directly.

For each component you can declare the input parameters it accepts — each with a name, a data type (**Text**, **Number**, **Boolean**, **Date**, or **Record Id**), whether it is required, and its own agent instructions describing what to pass. A parameter you have already saved has an **Enabled** switch at the top of its box: switching it to **Disabled** stops the agent from passing the parameter without losing its configuration; a parameter you have just added and not yet saved can be removed outright.

## Choosing how a component uses tabs

**Update Existing Tab** controls what happens when the agent opens a component that is already on screen. Leave it on — the default — and the component keeps one tab, which updates in place each time it is opened. Turn it off and every request opens another tab, which suits a component the user is meant to line up side by side.

With the setting on, each parameter gains a **Tab Key** checkbox. Marking a parameter means its value tells one tab from another: the open tab is reused only while that value stays the same, and a request carrying a different value opens a separate tab beside it. Mark `recordId` on an account summary, for example, and the agent gets a tab per account, the way Salesforce itself keeps a tab per record — opening the same account twice returns to the tab already showing it. Leave every parameter unmarked and a single tab serves the component whatever the values are.

Mark only the parameters that genuinely identify what the user is looking at. Marking an incidental parameter — a display mode, a sort order — splits the tabs on a detail the user does not think of as a different thing to look at, and they end up with near-duplicate tabs of the same record.

The same component may be registered several times — each setting is a distinct **invocation mode**, with its own title, agent instructions and parameters. A single component can this way serve different purposes: for example, one setting invokes it as a one-record summary titled "Account Summary" ( `recordId` ), another as a two-record comparison titled "Account Comparison" ( `leftId` , `rightId` ). The agent is offered every mode the user is permitted for as a separate entry and picks the right one by its instructions, naming its choice when it opens the component so its parameters are checked against that mode and its tab behaviour applied. A request that names no mode opens nothing. Each mode can be gated by a different custom permission, so different audiences can get different modes. Give each mode a title that says what it is for — two entries called the same thing leave the agent choosing between them on their instructions alone.

The list leads with each setting's **Title**, followed by the API name, the gating permission's label and an **Enabled** column:

- Use the row menu to **Edit** a setting, **Clone** it, **Disable** it (the agent can no longer open the component), or **Enable** it again.
- **Clone** opens the editor prefilled from the selected setting and saves as a new one — the quickest way to grant the same component under a second permission, or to add another invocation mode with a different parameter set. Change the title, the API name or the permission before saving so the clone

isn't an exact duplicate — and give it a title of its own, since that is how the agent and the user will tell the two apart. The **Definition Name** arrives as the original's with `_copy` appended, so the clone saves without clashing with the setting it came from; edit it if you would rather it read differently.

- Settings are never deleted from this page, so a component can be turned off and back on at any time.

**Note:** Access settings are matched by API name only — AgentPlus does not verify that the component still exists in your org. If you delete or rename a Lightning Web Component, its enabled setting keeps offering it to the agent, and opening it fails in the user interface even though the agent is told the open succeeded. When you remove or rename a component, **Disable** its setting or update it to the new name.

**Note:** This page never deletes a record: disabling is how a setting is retired, and it keeps the configuration recoverable. To remove records permanently, go to **Setup > Custom Metadata Types**, click **Manage Records** next to **AgentPlus LWC Access** (or **AgentPlus LWC Parameter**, or **AgentPlus Voice Agent**) and delete them there. Delete a parameter before the access setting it belongs to.

**Note:** Saves on this page take a few moments to apply. When a change settles within a few seconds, the page shows a green success notification. If the change has been submitted but hasn't confirmed yet — usually under higher org load — a yellow warning toast appears instead: *"Saved. If the change doesn't appear, refresh in a moment."* If you see the warning, reload the page after a few seconds; the change will already be applied.

## Managing Configuration with CI/CD

Everything the admin console manages is stored as public Custom Metadata records —

`agpls__AgentPlusVoiceAgent__mdt` (voice agent approvals), `agpls__AgentPlusLwcAccess__mdt` (component access), and `agpls__AgentPlusLwcParameter__mdt` (component parameters). The configuration can therefore be versioned and promoted between orgs like any other custom metadata: build and test it in a sandbox using the admin console, then move the records to other orgs with your usual release tooling.

Notes for pipelines:

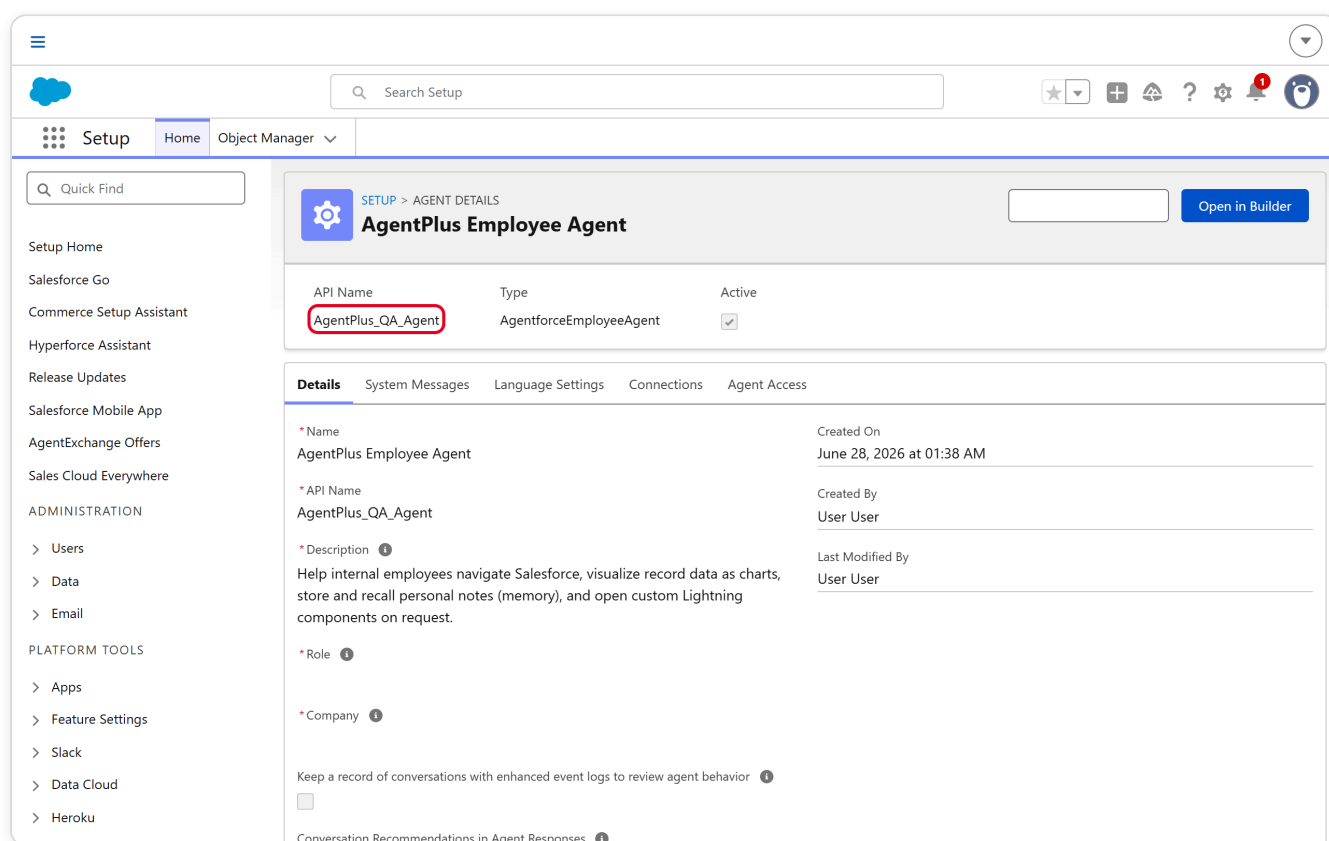
- In the record XML, field names carry the package namespace (for example `agpls__ComponentApiName__c`); record developer names do not.
- Parameter records reference their parent access setting by developer name in the `agpls__LwcAccess__c` field (the value is the record name only — the type prefix is added automatically).
- Custom metadata deployments complete asynchronously, like the console's own saves: allow for that in a pipeline rather than assuming the records are in place the moment a deployment is submitted.
- The admin console never deletes records (disabling preserves them); a pipeline can remove obsolete records with a destructive changes manifest.

## Agent Selection

AgentPlus Voice determines which agent to use from admin approval plus a single placement property:

- **Admin approval:** An agent is only available to a user when an admin has approved it in the [AgentPlus Admin Console](#) (Voice Agents card) and the user holds the custom permission linked to that approval.
- **Restrict to Agent:** Each placement of the **AgentPlus Companion** — a utility item, a Lightning page in App Builder, or a page of an Experience Cloud site in Experience Builder — has a **Restrict to Agent** property. Set it to an agent's API name to lock that placement to that one agent: the selector is not shown, and a user who isn't approved for that agent sees the no-agent message rather than being switched to a different one. Leave it blank — the default — and the user is offered every agent they are approved for, with a selector to switch between them when there are two or more.

To identify an agent's API name (for example, to set the **Restrict to Agent** property on a placement), navigate to **Setup > Agentforce Agents** and click the agent's name. Depending on the agent builder version, the API name is listed under **API Name** or **Developer Name**.

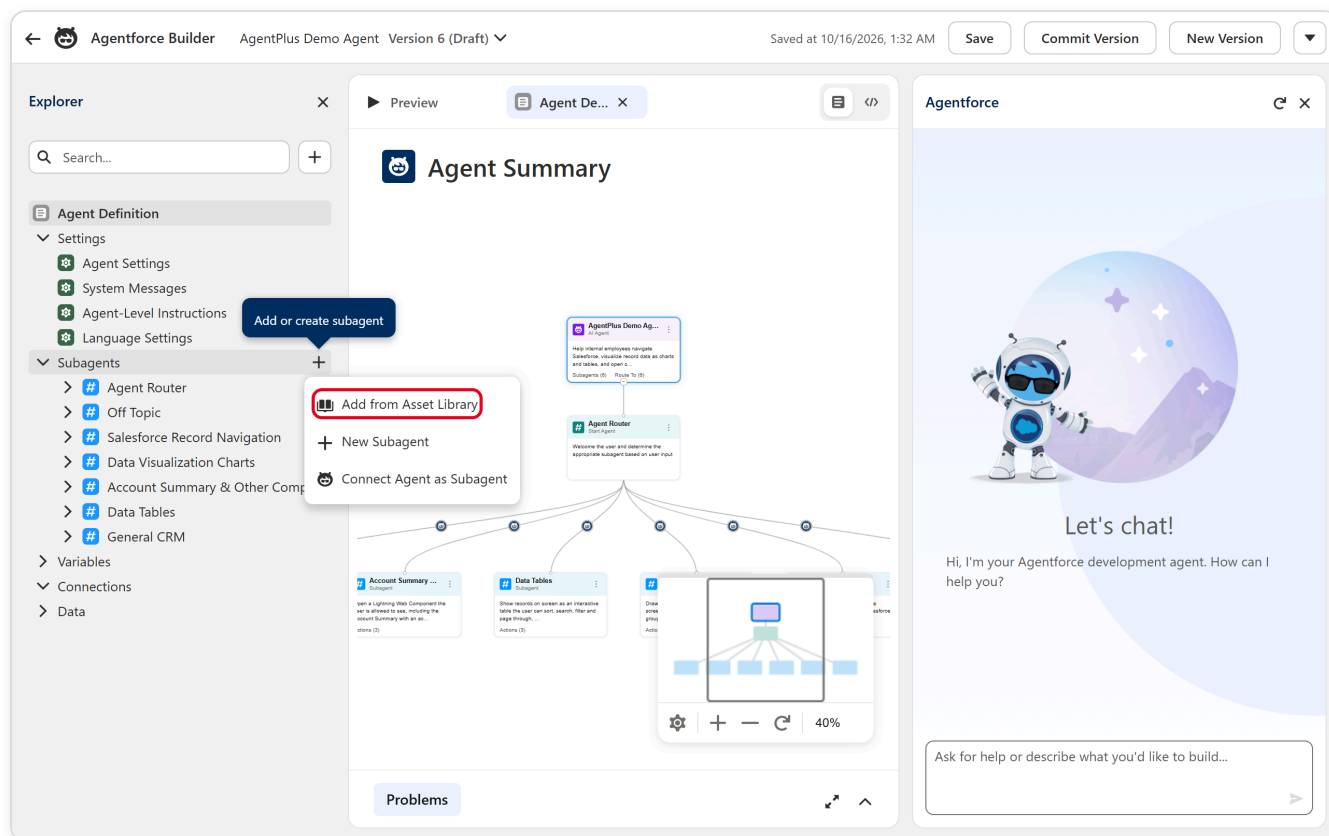


Agent detail page — locating the agent's API name

## Agent Activation

Open an internal agent and add the desired AgentPlus subagents — **Agent Charts**, **Agent Data Tables** or **User Navigation** — or add the individual actions, like **Open LWC**, to your own subagents. Open LWC ships as actions rather than a subagent, so it is always wired into a subagent of your own. When adding actions to your own subagents, don't forget to include instructions to ensure they are invoked correctly.

Use the **new builder (Agentforce Studio)** to maintain your agents: Salesforce is retiring the legacy builder, so new subagent and action wiring should be done in Agentforce Studio. Open **Agentforce Studio** and select your agent. If the builder opens the active version, click **New Version** to start a draft. In the Explorer, hover over the **Subagents** folder, click the **+** button, and choose **Add from Asset Library**. The active version keeps serving users while you prepare the draft, and you activate it when ready.



Agentforce Studio — adding AgentPlus subagents from the Asset Library

If you are still on the legacy builder, the same option lives under **New > Add from Asset Library** — but plan the move to Agentforce Studio. Unlike the new builder, the legacy builder requires deactivating the agent before editing and reactivating it afterwards.

### Known Issues When Importing AgentPlus Subagents

After importing the AgentPlus subagents in the new builder (Agent Script), the following adjustments may be necessary for the agent to compile:

- **Remove the `source` attribute** from custom actions:

```

Navigate:
  description: |
    Make the user interface navigate to a specific item, be it a record by its Id,
    an app by its DurableId or an object by its API name and action type.
  label: 'Navigate'
  require_user_confirmation: False
  include_in_progress_indicator: True
  progress_indicator_message: 'Navigating'
  source: 'agpls__Navigate' # <- remove this line
  target: 'apex://agpls__NavigateAction'

```

- **Give standard actions the `standardInvokableAction://` target.** Standard actions such as **Identify Record by Name** are supported in the new builder, but every Agent Script action needs an explicit `target`, and an imported bundle often arrives without a usable one — which is what makes these actions fail to compile after an import. Point the target at the action's API name:

```

IdentifyRecordByName:
  description: |
    Finds a record by its name.
  label: 'Identify Record by Name'
  target: 'standardInvokableAction://EmployeeCopilot__IdentifyRecordByName'

```

The compiler validates the target's *scheme*, not the action's existence, so a standard action must also be available in the org for the agent to actually run it. Verify the agent invokes it once, rather than trusting a clean compile.

- **Fix multi-level action outputs.** The new builder struggles with importing actions whose response has more than one class level, and the compiler rejects an `object` or `list[object]` output that does not declare a `complex_data_type_name`. This affects every AgentPlus discovery action — **Get Available Apps**, **Get Accessible Objects**, **Get Accessible Fields**, **Get Available List Views**, **Get Available Tabs**, **Get Available LWCs**. Each needs its output declared with the Apex class type behind it. **Get Available Apps** is shown here as the pattern; apply the same shape to the others, substituting the action's own class and inner class (beware of the indentation):

```

GetAvailableLightningApps:
  description: |
    Retrieves available apps for the current user.
  label: "Get Available Apps"
  require_user_confirmation: False
  include_in_progress_indicator: False
  progress_indicator_message: "Retrieving apps"
  target: "apex://agpls__GetAvailableLightningAppsAction"
  inputs:
    "excludedAppDeveloperNames": string
    description: |
      Developer names of apps to leave out of the result, copied from an
      earlier call's developerName. Never pass a display label here.
    label: "excludedAppDeveloperNames"
    is_required: False
    is_user_input: False
  outputs:
    "applist": list[object]
    description: |
      List of available Lightning apps with description, durable id, and label.
    label: "apps"
    is_displayable: True
    filter_from_agent: False
    complex_data_type_name: "@apexClassType/agpls__GetAvailableLightningAppsAction$AppResult"

```

`complex_data_type_name` names the Apex class the same way `target` does: with the namespace of whichever org owns that class. AgentPlus is installed as a managed package, so from your org's point of view the class belongs to the `agpls` namespace — the same prefix as `target`'s `apex://agpls__...`. (The `c__` form only applies to a class that lives in your own org — one you wrote yourself, with no namespace of its own — which is never the case for a class inside an installed package.) If you get the prefix wrong, publishing fails with a message that quotes the exact string to use, so let the validator settle it rather than guessing:

```

Validation failed for action 'GetAvailableLightningApps' due to the invalid value for
attribute 'complex_data_type_name' for the output parameter 'applist'. To fix, update
the 'complex_data_type_name' attribute value ... to
 '@apexClassType/agpls__GetAvailableLightningAppsAction$AppResult'.

```

Nothing is committed when that validation fails, so an attempted publish is a safe way to have the platform tell you the right value.

# Using AgentPlus

## Smart Navigation

AgentPlus allows you to navigate Salesforce entirely through voice or text commands, eliminating the need to click through multiple menus.

- **Targeted navigation:** Ask the agent to move between applications, object tabs, specific record pages, list views, custom tabs, a record's related lists, and standard pages such as Home, Chatter or Today.
- **Contextual awareness:** The agent can retrieve the Lightning apps, objects, list views and tabs available to you, ensuring navigation requests are valid for your user. For a related list it uses the object's relationships; for named pages only Home, Chatter and Today are available.
- **With the Companion on a page:** Going to another page leaves the conversation, so the Companion asks you first (see [Leaving the conversation](#)).

### Sample prompts:

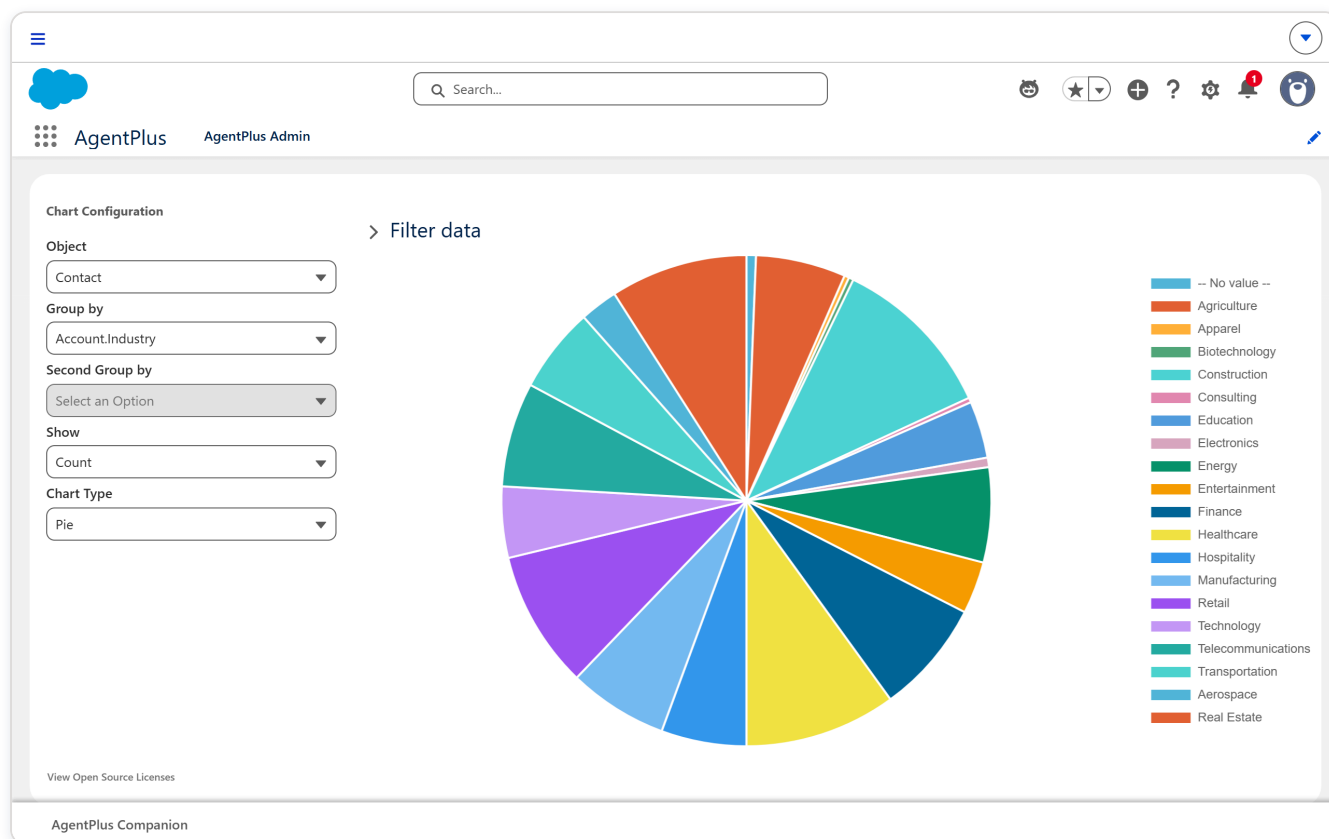
- *Take me to my most recent account*
- *Open my accounts list view "My Key Accounts"*
- *Show this account's contacts*
- *Open the AgentPlus Voice tab*
- *Take me home*

## Visual Charts

Generate real-time data visualizations directly within the interface.

- **Chart types:** Bar, line, pie, donut, area and radar. Pie and donut show a single grouping as a share of the total; if you ask for a second grouping on one of them, it's left off and the chart is drawn on the first grouping alone. Radar compares several dimensions; area emphasises volume over a trend.
- **Data organization:** Group data by up to two dimensions and apply aggregate functions such as count, count distinct, average, minimum, maximum, and sum. You can group and filter by a field one relationship away, written as `Relationship.Field` — for example, "contacts by account industry" groups on `Account.Industry`. (Lookups that can point to more than one kind of record aren't supported for this.)
- **Readable labels:** When you group by a lookup field such as Owner or Account, the chart shows the record names rather than raw IDs.
- **Dynamic filtering:** The chart includes a built-in filter builder that allows you to add multiple conditions with type-aware inputs (text, number, date, picklist, boolean). You can combine conditions using AND/OR logic with parentheses, and use operators including equals, not equals, greater/less than, IN, NOT IN, and blank checks. Filters update the chart in real time.
- **Live updates:** Once a chart is open, the agent can refine it — changing the chart type, grouping, aggregation, or filters — without closing and reopening the tab.

- **View-only placement:** Administrators can add the AgentPlus Chart Visualizer component to a Lightning app page and enable its **Hide Settings** option to present a chart without the configuration panel and filters.
- **On a phone:** Two buttons beside the chart's title open the chart's settings (object, grouping, aggregation, chart type) and its filters, each in a bottom sheet of its own, and the legend sits below the chart, so the chart keeps the full width and height of the screen.



Agent-generated pie chart: contacts grouped by their account's industry

### Sample prompts:

- *Show my open cases grouped by origin*
- *Show opportunities by account industry as a pie chart*
- *Break down accounts by owner*

## Data Tables

When you want to see the underlying records rather than a summary, ask the agent for a table. Where a chart aggregates, a data table lists the individual rows with the columns you ask for.

- **Columns you choose:** Name the fields you want as columns, including one field from a parent record (for example, a contact table with the account's industry). If you don't specify columns, the agent picks sensible ones.
- **Only what you can see:** The table shows only the columns and records your permissions allow. If a requested column isn't available to you, it's left out and the table notes how many were hidden, rather

than failing.

- **Built-in search, filter and sort:** Type in the search box to narrow the rows, use the filter builder to add conditions, and click a column header to sort. Lookup columns appear as clickable links to the record.
- **Open a row's record:** The record's name (for example, an opportunity's name or a case number) is a link to that record. It opens in a new browser tab, so the table keeps its search, filters and page.
- **Pagination:** Large results are paged; use the arrows to move one page at a time, or type a page number and press Enter to jump straight to it. For very large results, paging stops after the first couple of thousand rows — the table tells you how many records are reachable, and narrowing with search or filters brings the rest within reach. Counting stops at 10,000 records; beyond that the table shows "10,000+".
- **Live updates:** Once a table is open, the agent can refine it — adding a column or changing filters or sorting — without opening a new tab.
- **View-only placement:** Administrators can add the AgentPlus Data Table Viewer component to a Lightning app page and enable its **Hide Settings** option to present a table without the filter and search controls.

The screenshot shows the AgentPlus Admin interface. At the top, there's a search bar and navigation icons. Below, the 'Contacts' section is active, showing a table with columns: Full Name, Email, and Account Industry. The table contains 10 rows of contact data. Below the table, there's a pagination control showing 'Page 1 of 32 (320 records)'.

| Full Name      | Email                      | Account Industry   |
|----------------|----------------------------|--------------------|
| Andrew Allen   | andrew.allen@example.com   | Retail             |
| Andrew Brown   | andrew.brown@example.com   | Education          |
| Andrew Clark   | andrew.clark@example.com   | Agriculture        |
| Andrew Green   | andrew.green@example.com   | Finance            |
| Andrew Hall    | andrew.hall@example.com    | Manufacturing      |
| Andrew Harris  | andrew.harris@example.com  | Healthcare         |
| Andrew Jackson | andrew.jackson@example.com | Construction       |
| Andrew Johnson | andrew.johnson@example.com | Retail             |
| Andrew King    | andrew.king@example.com    | Telecommunications |
| Andrew Lewis   | andrew.lewis@example.com   | Education          |

Agent-generated data table with search, filter and pagination

### Sample prompts:

- *Show me the open cases from EMEA as a table*
- *List my opportunities with their stage and amount*
- *Give me a table of contacts with email and account industry*

## AgentPlus Voice

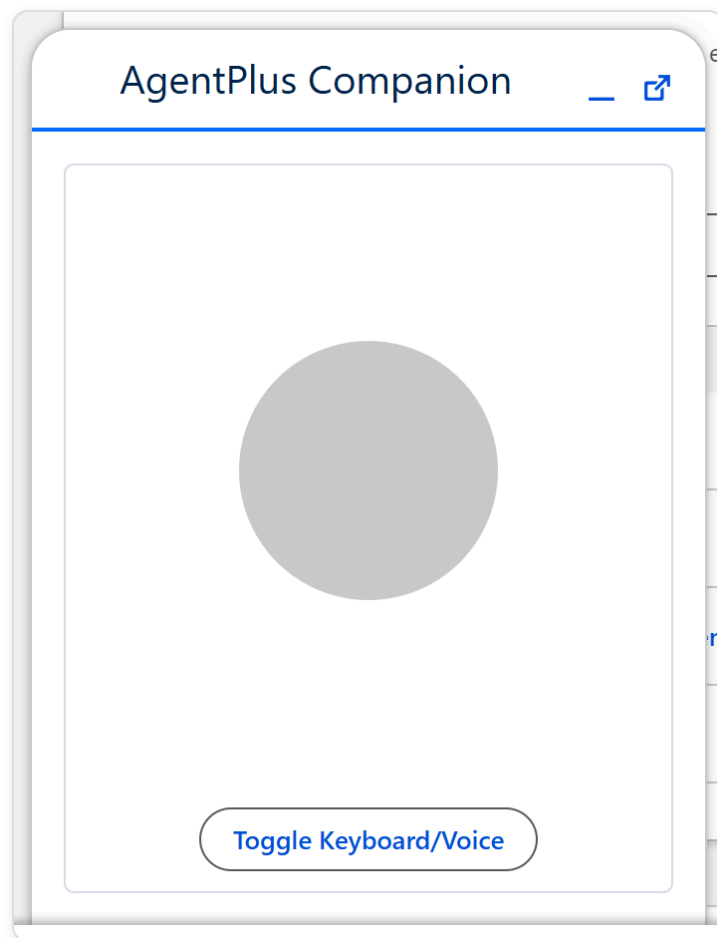
For a hands-free experience, use the AgentPlus Voice interface.

- **Voice control:** Each agent listens and speaks either with your browser's built-in speech or with Salesforce speech, as your administrator chose for that agent. Browser speech recognition is best supported in Chrome and Edge on desktop and Android; where it is unavailable, keyboard input keeps working. Salesforce speech works in every browser, including Firefox and phones, and reads replies in the same natural voice on every device, the Salesforce mobile app included.
- **Interaction:** Toggle between keyboard and voice input within the AgentPlus Companion to suit your current environment. AgentPlus remembers the mode you used last on each device, so the conversation opens the way you left it. While the agent prepares its reply, the conversation shows it typing.
- **One conversation listens at a time:** If AgentPlus is open in more than one place in the same browser window, only the conversation you last used listens and speaks. A conversation on a page you have left, which Salesforce keeps open out of sight so that Back is instant, stays silent.

Click the circle to activate and deactivate the voice chat. While the agent is speaking, clicking the circle stops the reply at once and switches the voice chat off; click it again when you want to talk.

While it is listening, the circle gently grows and falls with your voice, so you can see that you are being heard before you finish speaking, and a soft ring drifts out from it when you start to talk. It holds still, with no rings, when nothing is being picked up — a still circle means it is your turn. Your browser asks for permission to use the microphone the first time, and its recording indicator stays on only while the circle is listening.

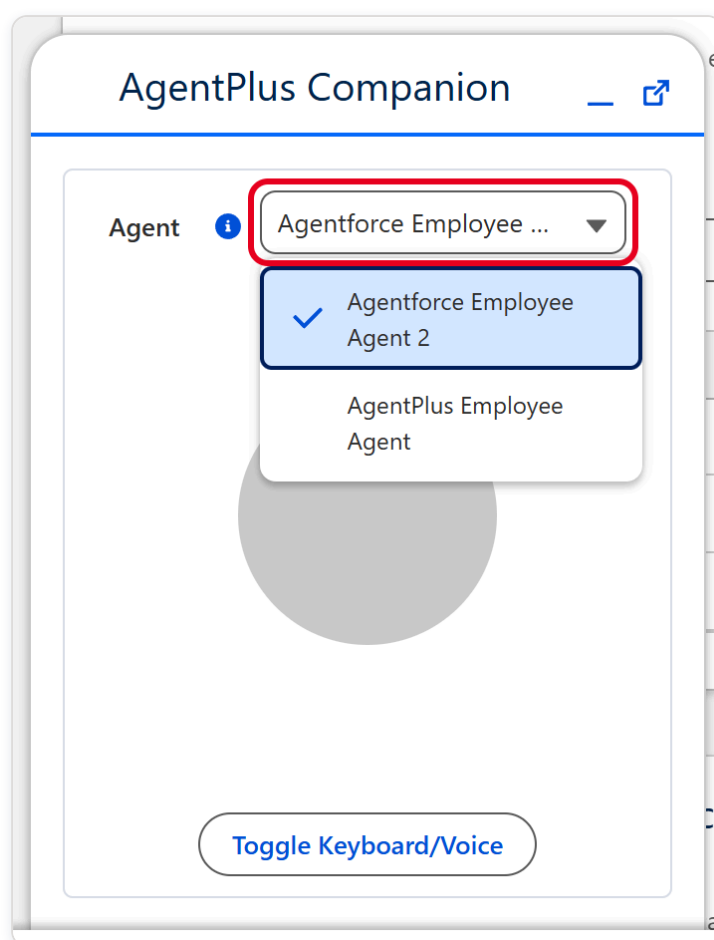
With Salesforce speech recognition, the agent waits until you pause for about a second before it takes what you said; the circle shows the thinking state while your words are transcribed, and they appear in the conversation once transcribed. With a Salesforce voice, the circle also moves with the agent's voice while it speaks. If Salesforce speech is unavailable for a moment, your browser's speech takes over for that turn and a message tells you so once.



AgentPlus Companion — the voice orb and keyboard/voice toggle

## Choosing an Agent

When two or more agents are available to you, an **Agent** dropdown appears at the top of the panel listing them. The info icon next to the label shows a tooltip reminding you that switching agents starts a new conversation.



AgentPlus Voice — agent selector dropdown

Only agents that have been approved by an admin, are currently active, and are gated by a custom permission you hold are shown. An agent approved ahead of its activation joins the list once it is activated. Select any agent from the list to switch to it. Switching starts a fresh conversation — the transcript resets because each conversation session belongs to a single agent.

The agent selected when the component loads depends on how the placement is configured:

1. When the placement is restricted to one agent, that is the agent you talk to and the dropdown is not shown. If you aren't approved for it, you get the no-agent message rather than a different agent.
2. Otherwise, the first agent in your available list, with the dropdown to switch.

If the dropdown is not visible, the placement is either restricted to a single agent or fewer than two agents are available to you in the [AgentPlus Admin Console](#).

If no agent is available to you at all — you aren't approved for the agent this placement is restricted to, or for any agent — the component shows the message *"No agent is available. Ask your administrator to enable one."* instead of the chat. Contact your administrator to approve an agent and grant you the required custom permission.

**Note:** The server re-validates your permission on every message, not only when the page loads. If your administrator removes your access to an agent after you have opened the panel, your next message returns a clear error rather than invoking an agent you are no longer entitled to. Permission changes take effect immediately — no page reload is required.

## Charts, Tables and Components While You Talk

Where the agent's charts, tables and components appear depends on where your administrator placed the AgentPlus Companion:

- **In the utility bar:** They open in a workspace tab in a console app, and as a page of the app otherwise. A chart, table or component of the same kind that is already on screen is updated in place instead.
- **On a page** — the Salesforce mobile app, a Lightning page or an Experience Cloud site: They open in a panel over the conversation. The conversation keeps running underneath, so you can keep talking while you look at the result. Use the minimize button to fold the panel down to its title bar, which rests below the conversation and stays in view at the bottom of the screen as you scroll, and the close button to dismiss it; the next result opens it again. On a phone the panel fills the screen. In the Salesforce mobile app and on sites, a result appears once the agent has finished its reply.

What you ask for in a conversation appears on the device where you spoke. If AgentPlus is also open on your computer, the chart you requested on your phone opens on your phone and leaves the computer alone.

## Leaving the Conversation

On a page placement, a page the agent opens — a record, a list view, an app — replaces the conversation, so the Companion asks first, naming the destination:

**Leave the conversation?** Opening *Acme Corporation* leaves the conversation. You can come back to it with Back.

Tap **Open** to go there, or **Stay** to keep talking. Voice stops when you leave, and when you come back with Back the conversation carries on where you left it, without asking whether to continue.

## AgentPlus Voice in the Salesforce Mobile App

AgentPlus runs inside the Salesforce mobile app. The mobile app has no utility bar, so the package includes an **AgentPlus Voice** app page and tab, which hosts the AgentPlus Companion, to add to the app's navigation menu.

To make it available (admin):

1. In **Setup**, enter **Salesforce Navigation** in the Quick Find box and open it.
2. Move **AgentPlus Voice** to the **Selected** list, place it where users should find it, and click **Save**.
3. Make sure your users hold the **AgentPlus User** permission set, which makes the AgentPlus Voice tab visible, and a custom permission that approves a voice agent (see [Voice Agents](#)).

Users then open **AgentPlus Voice** from the app's menu. The same tab can also be added to the navigation of any Lightning app in **App Manager**. To offer a page tied to a single agent, create your own app page in Lightning App Builder with the **AgentPlus Companion** component and set its **Restrict to Agent** property.

**Important:** Browser voices do not play in the Salesforce mobile app. For agents your users talk to from the mobile app, set **Voice Output** to **Salesforce** in the agent's [speech settings](#); otherwise replies appear as text but are not read aloud. Speech recognition works with either engine.

In the mobile app the conversation, voice and the agent's user-interface actions — Navigation, Charts, Data Tables and Open LWC — all work. A chart, table or component appears once the agent has finished its reply. It opens in a panel over the conversation that you can minimize or close, and a chart's settings and filters open in a bottom sheet of their own, opened from the buttons beside the chart's title. Before the agent takes you to another page, the app asks whether you want to leave the conversation (see [Leaving the Conversation](#)).

## Open Lightning Web Component

AgentPlus can open any Lightning Web Component (LWC) in a dedicated console tab — or in a panel over the conversation, where the AgentPlus Companion sits on a page — and pass parameters to it directly from the agent. No modifications are required to the target component.

- **Zero-modification compatibility:** The target component does not need any special boilerplate or awareness of AgentPlus. Existing and packaged components work out of the box.
- **Attribute passing:** The agent can supply any number of key-value pairs that map directly to the component's `@api` properties. The values are composed by the agent from the conversation — see the security note below.
- **Tab customisation:** The agent can set the tab label and icon for the opened component.
- **Live updates:** If the same component is already open in a tab, the agent can update its attributes without closing and reopening the tab.

To use this feature, add the **Open LWC** action to your agent and include the relevant instructions to ensure the action is invoked at the proper user prompts. Make sure you instruct the agent to pass the input parameters your component expects to receive.

**Access control:** Which components each user can open is governed by the **Lightning Web Components Access** settings configured in the [AgentPlus Admin Console](#). A component is only available to users who hold the custom permission mapped to it. The companion `apls__GetAvailableLWCs` action lets the agent retrieve the components available to the current user, each with its title and agent instructions. The Open LWC action then enforces the same permission: if a user asks to open a component they aren't mapped to, the request is denied regardless of what the agent attempts. The same check is applied again at the component tab itself — even if a user navigates directly to an open tab by URL, the component does not load until the permission is confirmed.

**Important:** Attribute values are not a security boundary. AgentPlus governs *which* component a user may open and *which* parameters it accepts, by name, type and whether they are required. It does not govern the *values*: those come from the agent, which composes them from the conversation, and a user can steer that conversation. Treat any attribute your component receives as untrusted input — the same way you would treat a URL parameter.

In practice, a record Id passed as an attribute proves nothing about the user's right to see that record. Your component must enforce access itself: query through Apex `WITH USER_MODE` (or `WITH SECURITY_ENFORCED`), use Lightning Data Service, or check permissions before rendering. A component that renders whatever Id it is handed will show that record to anyone who can open the component.

The rule of thumb: register a component for agent access only if it is safe to open with arbitrary attribute values, and put the real security check inside the component.

The action's properties are:

| Property                   | Description                                                                                                                                                                                                                                                                        |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LwcAccessDefinition</b> | <b>Required.</b> Identifies the registered setting to open. It is the only identifier the action takes: it determines the component that opens, the parameters the call is checked against, and the tab behaviour applied. A title or a component API name will not open anything. |
| <b>Attributes</b>          | JSON object containing key-value pairs to pass as parameters to the target LWC. Each key corresponds to an <code>@api</code> property on the component.                                                                                                                            |
| <b>TabLabel</b>            | Label displayed on the console tab. Defaults to "Opened by AgentPlus" if not provided.                                                                                                                                                                                             |
| <b>TabIcon</b>             | SLDS icon name for the console tab (for example, <code>utility:apps</code> ). Optional.                                                                                                                                                                                            |

**Note:** The agent cannot name a component directly — it can only open a setting registered on the Admin Console, identified by its `LwcAccessDefinition`. That is deliberate: the components an agent can open are exactly the ones registered there, and no instruction or conversation can widen that set.

Discovery does not tell the agent a component's API name at all. What it returns is the **Title** you gave the setting, which is what the agent says when it offers the component or confirms what it opened. Write titles the way you would write a menu item — "Account Summary", not "accountSummaryLwc".

## Wiring Open LWC into an agent

There are two ways to give a subagent access to a component, and the choice is about **who decides which component opens**: you, when you write the agent, or the agent, at conversation time. Both enforce the same permissions.

**Finding an `LwcAccessDefinition`.** Every setting shows its definition name in the **Definition Name** column of the **Lightning Web Components Access** table in the [AgentPlus Admin Console](#) — copy it from there. You choose the name yourself when you create the setting: leave the field blank and one is suggested from the title (a setting titled "Account Summary" becomes `account_summary`), or type your own. Once the setting is saved the name is fixed, because it is the record's identity and the setting's parameters hang off it.

**Option 1 — Pin the component.** The subagent always opens the same setting, so you write the definition name straight into the action binding and the agent never chooses. Use this when the subagent has one job: it is the simplest wiring, it costs no discovery round-trip, and the agent cannot open anything else even if the conversation pushes it to.

```
subagent account_summary:
  label: "Account Summary"
  description: "Show the account summary component for an account."

  reasoning:
    instructions: ->
      | When the user asks to see an account's summary, open it with open_component.
      | Pass the account's record Id in the attributes as recordId.
  actions:
    open_component: @actions.open_component
      with lwcAccessDefinition = "accountSummary"
      with attributes = ...
```

The agent still supplies `attributes`, because those depend on the conversation — but `lwcAccessDefinition` is fixed, so it is not something the agent can get wrong.

**Option 2 — Let the agent choose.** The subagent first calls `agpls__GetAvailableLWCs` to retrieve the settings the current user may open — each with its title, description and declared parameters — then picks the one that matches the request. Use this when several components are registered, or when admins add new ones over time and you do not want to revisit the agent each time.

```

subagent lightning_component_access:
  label: "Lightning Component Access"
  description: "Open custom Lightning Web Components the user is allowed to access."

  reasoning:
    instructions: ->
      | First call get_available_components to learn what the user is allowed to open,
      | with each entry's title, description and parameters.
      | Each entry carries an lwcAccessDefinition. The same component can appear more than
      | once, each entry a different way to use it, so pick the entry whose description fits
      | what the user asked for.
      | Then open it with open_component, passing that entry's lwcAccessDefinition exactly
      | as it was returned, along with any parameters that entry declares.
      | Refer to an entry by its componentTitle when you speak to the user – that is the name
      | they will recognize. The lwcAccessDefinition is an internal identifier: pass it, never
      | say it. Never invent one, and never pass a title in its place.

    actions:
      get_available_components: @actions.get_available_components
      open_component: @actions.open_component
        with lwcAccessDefinition = ...
        with attributes = ...

```

Here `...` means the agent fills the value in — for the definition it chose and for the attributes that entry declares. The instruction to pass the returned definition **unchanged** matters: an invented or reworded identifier matches no setting and the action refuses it.

Whichever option you choose, define the actions themselves once in the subagent's `actions:` block, pointing `open_component` at `apex://agpls__OpenLwcAction` and, for Option 2, `get_available_components` at `apex://agpls__GetAvailableLwcsAction`. Note that `GetAvailableLWCs` returns a list of objects, so its output needs a `complex_data_type_name` — see [Known Issues When Importing AgentPlus Subagents](#) for that pattern.

**Sample prompt:** *Open the Account Summary component for this account*

# Troubleshooting

---

## Is It an Agentforce Issue or an AgentPlus Issue?

Keep in mind that AgentPlus is the **interface layer**: it tells the agent what it can do, enforces access, and reflects the agent's actions in the user interface — all the reasoning (which action to invoke, with which values, how to phrase the answer) happens inside Agentforce. When something misbehaves, first identify which layer is failing. Open the agent's reasoning/plan trace in the builder — or enable conversation logs — and check whether the AgentPlus action actually ran:

- **The action never ran.** The agent answered on its own, picked a different action, or asked for more input. This is Agentforce configuration: review the subagent's instructions and the action descriptions so the agent knows when to invoke it, and confirm the action is attached to the subagent in the builder.
- **The action ran and returned an error.** AgentPlus actions return precise results — a missing permission, an unknown component, exactly which parameter was missing or mis-typed. Fix the access settings in the [AgentPlus Admin Console](#) or adjust the agent's instructions to pass the right values.
- **The action ran and reported success, but nothing happened on screen.** Confirm the AgentPlus Companion is present and started in the current app or on the current page, and that the tab you expect to react is the one you are looking at.

As a quick rule: wrong words or wrong action choice → Agentforce; access or parameter errors → AgentPlus configuration; success with no UI reaction → the Companion.

For the Agentforce side, Salesforce's own guidance is the best companion to this section: [Troubleshoot Agentforce](#) and [Tips to Help You Troubleshoot Agentforce with Confidence](#).

## Common Issues

- **AgentPlus Voice — no agent available:** If the component shows *"No agent is available. Ask your administrator to enable one."*, no agent has been approved for you in the Voice Agents settings. Ask your administrator to approve an agent in the [AgentPlus Admin Console](#) and grant you the custom permission linked to that approval. Make sure the approved agent is also active in Agentforce.
- **Voice agent selector — invocation error:** If sending a message produces an "agent not allowed" or invocation error, the selected agent may not be among your permitted agents. Reload the page to refresh the selector, or ask your administrator to verify that your user holds the custom permission gating that agent.
- **Navigation issues:** Page context is not supported for custom agent actions; the agent may not automatically know which record page you are currently viewing unless specified in the prompt.
- **The UI reacts in another tab, on another device, or not at all:** Only what you can see reacts — a background browser tab, or a page Salesforce keeps open out of sight after you left it, deliberately ignores agent actions so several open tabs don't all react. What you ask for in a conversation appears on the device where you spoke. Switch to the tab you want to act on before asking, and make sure it runs an app with the AgentPlus Companion in its utility bar or on its page.

- **The agent says a component opened, but the tab shows an error:** The Lightning Web Component was deleted or renamed while its access setting is still enabled. Disable the setting in the [AgentPlus Admin Console](#) or update it to the component's new name.
- **The agent says the interface can't be updated right now:** Your org has temporarily reached a Salesforce usage limit, so the agent says so rather than report an action you never saw on screen. Normal service resumes when the limit resets, usually within a day.
- **Permission errors:** If the agent fails to perform a task, verify that the agent user has the necessary Apex class permissions and the user has the AgentPlus User permission set.
- **The agent doesn't read its replies aloud in the Salesforce mobile app:** Browser voices do not play there. Set the agent's **Voice Output** to **Salesforce** in its [speech settings](#).
- **A chart, table or page doesn't appear in the Salesforce mobile app or on a site page:** There, the result appears once the agent has finished its reply, not while it is still talking. If it never appears, check that the page hosts the **AgentPlus Companion** — see [AgentPlus Voice in the Salesforce Mobile App](#) and [Experience Cloud Sites](#).
- **Debugging logs:** To investigate complex issues, activate conversation logs on the agent and use the platform integration user to review debug logs. Be aware that session logs may take some time to appear progressively.

## Contact Support

If you encounter any issues that are not covered in this guide, please contact your Salesforce administrator or our support team for assistance.