

Administrator Guide

Introduction

InnovatorEdge AI is the agent platform for Aras Innovator. It hosts governed AI agents that reason over PLM data, call tools to retrieve and change it, delegate work to specialist agents, and reach systems beyond Innovator through MCP.

Every agent reaches your data through Edge API. You decide which item types, properties and actions are exposed, and every access is logged. For the agents your people talk to, requests run under the identity of the signed-in user, so each agent is bounded by the permissions of the person using it.

Conversation is the primary surface. A user asks a question in plain language and receives an answer drawn from live Innovator records, each with a link back to the record it came from.

Agents grow with your needs. Add skills to teach an agent domain-specific patterns, connect it to systems beyond Innovator through MCP, and compose agents so that one delegates specialist work to another.

Agents act on data as well as read it. They can create records, update properties, change lifecycle state, and delete items, and every operation that changes data requires explicit human approval before it is committed.

InnovatorEdge AI hosts as many agents as you need. Each one carries its own instructions, tools, skills, and data connections. The Aras AI Assistant ships as the default governed agent in every tenant, and you can create additional agents for specific tasks or domains.

Your work as an administrator happens in the Agent Management Platform, where you configure agents and the resources they draw on: Access Points to Innovator data, model configurations, skills, and the tool registry. Who can do what is controlled through Centralized Identity Access Management (CIAM).

Who This Guide is For



This guide is for the **Edge AI Administrator**, the person responsible for setting up, configuring, and operating InnovatorEdge AI for their organization: agents, models, Access Points, skills, MCP Servers, and user access within the tenant.

It assumes familiarity with Aras Innovator administration and with your organization's Edge API configuration.

What This Guide Does not Cover

Topic	Where to find it
Using the assistant as an end user	User's Guide
Configuring Edge API itself, including exposing item types, properties, and actions	Building your API schema
Connecting an Aras Innovator instance to Edge API	Connect your Aras Instances
Creating and managing Edge API keys and access tokens	Managing API keys and Access tokens
CIAM administration beyond assigning Edge AI roles	CIAM Documentation
Aras Innovator administration	Aras Innovator release-specific documentation

Related Documentation

- [Frequently Asked Questions](#) for common questions on setup, security, capability, and behavior.
- [Appendix: Glossary](#) for terms specific to InnovatorEdge AI.
- [Edge AI Release Notes](#) for what changed in the current release.

Getting Started

This chapter takes your tenant from newly provisioned to a working assistant that your users can reach. Work through the steps in order. Several of them happen in services outside InnovatorEdge AI, and each points to the guide that covers it in detail.

What You Need

Before you begin, make sure the following are in place:

- An Aras Innovator instance, and administrator access to it.
- An active Edge account for your organization, with both an Edge API entitlement and an Edge AI entitlement. Both are required. See [Get Started With Aras InnovatorEdge API](#) .



- At least one Innovator instance connected to Edge API with a valid trusted domain certificate.
- The Edge AI CIAM audience value (`b832fca416424121bc28a7d3a3a832f2`) added to the **Allowed audience** field of each Edge API you intend to use with End User authentication. This value is fixed across all tenants. Omitting it produces authentication failures for every user on that Access Point, with no indication of why. See [Managing API keys and Access tokens](#) for where to register this value in Edge API.
- Tenant administrator access in Aras CIAM, so that you can assign Edge AI roles.

Important

The trusted domain certificate is what allows Edge API to identify the requester. Without it, an Access Point using End User authentication cannot verify who is asking, and identity-aware behavior is unavailable.

Setup Checklist

1. **Enable Edge API for your organization.** InnovatorEdge AI reaches your Innovator data through Edge API, so this establishes the layer everything else runs on. The designated tenant owner requests the Edge account; other people in the organization request user access from their tenant owner. See [Get Started With Aras InnovatorEdge API](#) .
2. **Confirm the CIAM tenant notification.** Aras shares the InnovatorEdge AI application with your tenant. Once shared, it appears in your CIAM Shared Applications list and the tenant administrator is notified automatically. Confirm the notification has arrived before continuing.
3. **Assign roles in CIAM.** Open CIAM user management and assign Edge AI User to the people who will use the assistant, and Edge AI Administrator to the people who will manage it. Users receive an email invitation when they are first granted access. If your organization uses Innovator user synchronization, configure it here. See **Roles and Access** on this page, and [CIAM user documentation / User Management](#) .

Important

A user with no assigned Edge AI role cannot reach the assistant or the Agent Management Platform. This step decides who has access, so complete it before you announce the service internally. We recommend managing access through dedicated groups rather than individual assignments; see CIAM Documentation for guidance on group and user assignment.



4. **Install the Aras Innovator AI Assistant.** The embedded assistant panel is delivered as a platform component application installed into your Aras Innovator instance. Download the package from your Edge account and install it following the Aras Innovator AI Assistant SaaS Installation Guide, which is provided in the Innovator AI Assistant folder inside the package. Once installed, the assistant is available to users as an embedded panel within Innovator.
5. **Connect your Innovator instance in Edge API.** Register your Aras Innovator instance as a data source, then expose the item types, properties and actions your agents will need. What you expose here sets the outer boundary of what any agent can reach, and it decides whether agents can change data at all: an agent can write only where the API exposes an Add, Edit or Delete action. See [Connect your Aras Instances](#) and [Building your API schema](#) .
6. **Add an Access Point in the Agent Management Platform.** Open the Agent Management Platform and register the Edge API connection as an Access Point: the endpoint URL and the authentication mode. With End User authentication, the signed-in person's own identity is used for every request, so there is no credential to store. If you choose API Key authentication instead, create the key in Edge API first and enter it here. See **Access Points** on this page.
7. **Configure and publish the Aras AI Assistant.** Open the Aras AI Assistant in the Agent Library. Select the Access Point you created, choose the tools you want the assistant to use, and save. Test it in Simulation Chat if you want to see how it behaves first. Then publish it. See **Agents** on this page.

Important

Publishing is always explicit. Registering an Access Point makes an agent publishable, because the agent can then be configured against at least one API. It does not select tools for you and it does not publish anything. An agent stays in Draft, reachable only by you in Simulation Chat, until you publish it.

8. **Verify it as a user would.** Open Aras Innovator, open the assistant panel, and ask one real question about data you know exists. Confirm three things: an answer comes back, it is scoped to what that account should see, and the link on a result opens the record in Innovator. If you have both an administrator and an ordinary user account available, try the same question with each and compare what comes back.

What Happens Next



Your Edge AI Users can now open the assistant in the embedded panel inside Innovator, or open the **Start a conversation** section at `edge.aras.cloud/ai` .

As you extend what Edge API exposes, agents pick up the change on their next request.

There is nothing to refresh in InnovatorEdge AI.

From here, this guide covers configuring the assistant further and creating agents of your own.

Navigating the Agent Management Platform

InnovatorEdge AI has a single entry point, at `edge.aras.cloud/ai` .

Everyone with an Edge AI role arrives at a home page. The page opens with a greeting that addresses the signed-in user by name and shows the name of their tenant. The **Start a conversation** section lists the agents available to the signed-in user; choosing one opens a conversation. Edge AI Users see this section only.

As an Edge AI Administrator, the home page also shows a **More for you** section with a **Manage agents** card. Selecting **Go to Agent Library** opens the **Agent Library**.

The Agent Management Platform is reachable only through **Manage agents**, and only by Edge AI Administrators. Keeping the platform in one browser tab and the assistant in another is the practical way to configure and test at the same time.

The platform is organized into two groups.

- **Agents.** The Agent Library, where you create, configure, publish, and manage agents.
- **Resources.** Access Points, Models, Skills, and MCP Servers. These are the building blocks an agent draws on, and each is shared across every agent in your tenant.

Aras provides some of these resources ready to use. Anything Aras manages is marked as such and presented read-only, so you can always tell what you inherit and what your organization owns. See **Skills**, and **Models** on this page.

Roles and Access

Access to InnovatorEdge AI is controlled through Aras CIAM, where Edge AI is registered as a single application. You assign each person one of two roles, and that role decides what they can reach and what they can do there.

Edge AI User



Converses with published agents, through the embedded panel in Innovator or the standalone chat. Cannot open the Agent Management Platform and cannot see or change any configuration.

Edge AI Administrator

Administers InnovatorEdge AI for one tenant. Creates, configures, publishes, archives and deletes agents, and manages Access Points, Models, Skills and MCP Servers. Can test draft agents in Simulation Chat, and can also converse with published agents.

Permissions Reference

Action	Edge AI User	Edge AI Administrator
Converse with published agents	Yes	Yes
Test draft agents in Simulation Chat	No	Yes
Open the Agent Management Platform	No	Yes
Create and configure agents	No	Yes
Publish and unpublish agents	No	Yes
Archive, restore and delete agents	No	Yes
Manage Access Points	No	Yes
Manage Models	No	Yes
Manage Skills	No	Yes
Manage MCP Servers	No	Yes

A role carries the same rights across every agent and resource in the tenant, so an Edge AI Administrator administers the whole tenant rather than a subset of it. Where a person holds more than one role, their access is merged.

What a Role Does Not Change

An Edge AI role decides what a person can configure. It has no effect on what data they can reach. Data access is decided by the Access Point an agent uses. Where that Access Point uses End User authentication, every query and every write runs under the signed-in person's own Innovator identity, so Innovator permissions decide the data and two people asking the same agent the same question receive answers scoped to their own access. See **Access Points** on this page.

Important

Granting Edge AI Administrator gives someone full control of agent configuration. It grants no additional access to Innovator data. If an administrator needs to see more data through an agent, that is a change to their Innovator permissions, not to their Edge AI role.



Users Without a Role

A person with no assigned Edge AI role cannot reach the assistant or the Agent Management Platform. They see an access denied message telling them to contact an administrator.

Assign roles as part of onboarding, before you tell people the service is available. See **Getting Started**, and [CIAM user documentation / User Management](#) for how to assign roles and manage them through groups.

Agents

An agent is a configured assistant. It carries instructions that shape how it behaves, a model that does its reasoning, the tools it is allowed to call, the skills that teach it your domain, and the Access Points that connect it to your data. Configuring an agent is deciding what it should be able to do, then confirming it behaves that way before your users meet it.

Your tenant starts with one agent, the Aras AI Assistant, and you can create as many more as you need.

The Agent Library

Every agent in your tenant is listed together with its publish state and whether Aras manages it or your organization does. Open an agent from here to configure it, or create a new one.

The Aras AI Assistant

Every tenant starts with a complete, production-ready agent that Aras builds, tunes and maintains.

The Aras AI Assistant arrives as a Deep Agent at high reasoning effort, with the full set of Aras-managed skills selected, the built-in Innovator tools available, and agent-to-agent delegation ready to use. Connect it to an Access Point, choose the tools you want it to use, publish it, and your organization has a capable PLM assistant without designing one.

Its real value is that it keeps getting better on its own. Aras refines its instructions, extends its skills, and adds tools with every release, and those improvements arrive in your tenant automatically. You are not maintaining a prompt library or tracking which techniques have improved since you deployed. That is also why Aras owns its core: the name, agent type and instructions stay under Aras control so the agent can keep advancing without breaking what you have built on top of it, and it cannot be deleted.



You extend it freely. Add your own skills, connect MCP tools, nominate sub-agents, and assign a different model where your organization requires one.

When you want an agent that is entirely yours, start from a template instead. The AI Assistant template gives you the same baseline, which you then own outright and adapt however your processes demand. See **Creating an Agent**.

Important

The two paths trade differently. The Aras AI Assistant stays current automatically and is the right default for broad PLM capability. An agent you build from a template is yours to shape without limits, and does not receive Aras updates. Many organizations run both: the assistant for general use, and purpose-built agents for specific processes.

Creating an Agent

Creating an agent takes two decisions, in order. First the agent type, which determines how the agent reasons. Then the template, which gives you a starting configuration for that type.

Choosing an Agent Type

Both types are general reasoning frameworks rather than PLM-specific behavior. They differ in how the agent approaches a question.

A **ReAct** agent works in a single loop. It reads the question, decides on one action, takes it, looks at the result, and decides what to do next. It keeps going until it can answer. This is direct and quick, and it suits questions that resolve in a few steps.

A **Deep Agent** plans before it acts. It writes out the steps it intends to take, then works through them, adjusting the plan as it learns. This suits investigative questions where the answer has to be assembled from several retrievals, and where the right next step is not obvious until the previous one returns.

The trade-off is straightforward. A Deep Agent handles harder problems, and it does so by running more loops, which means users wait longer for an answer and each answer consumes more tokens. A ReAct agent answers faster and more cheaply on work that does not need a plan.

Choose ReAct for focused lookups and reporting. Choose Deep Agent for analysis that spans several objects or relationships, and for agents that coordinate sub-agents.



Choosing a Template

A template is a starting configuration: instructions, tool selections and skill selections already filled in, so you are adjusting rather than starting from nothing.

Template	Agent type	Use it for
General Aras Assistance	ReAct	General-purpose PLM questions answered directly
AI Assistant	Deep Agent	The same baseline as the Aras AI Assistant, as an agent you own and adapt
A2A Orchestrator	Deep Agent	An agent whose job is to route work to specialist sub-agents

Templates are scoped to an agent type, which is why you choose the type first.

A template is a one-time copy. From the moment the agent is created it is independent, and later changes to the template do not reach it. That independence is the point: the agent is yours.

Configuring an Agent

The agent editor is where you decide what the agent is and what it may do.

- **Identity.** The name and a description. Users see the description when choosing between agents, so write it for them rather than for yourself.
- **Instructions.** Solution instructions define the agent's behavior, scope and tone. Say what it should focus on and what it should leave alone, for example to concentrate on change history and avoid surfacing unreleased parts. Platform instructions are authored by Aras and apply across all interactions.

Every agent also carries system guardrails, internal instructions that enforce security and data integrity. Instructions shape behavior within those boundaries. An instruction that conflicts with a guardrail is ignored, so you can write instructions freely without risking the platform's guarantees.

- **Model.** Choose from the model configurations available in your tenant. See **Models** on this page.
- **Tools.** Every tool an agent can call comes from a registered MCP server, and the editor groups them that way. The Aras-managed server carries the Innovator tools, subgrouped by the Access Point each one acts through; servers you register yourself carry everything else. Select the tools this agent needs. See **MCP Servers** on this page.



- **Skills.** Select the skills that teach this agent your domain patterns. Aras-managed skills are available to every agent; you can add your own alongside them, and disable an Aras skill on agents you own. See **Skills**.
- **Sub-agents.** Nominate the agents this one may delegate to. Where a sub-agent proposes a change to data, the approval surfaces in the user's own conversation rather than somewhere they cannot see it. See **Governed Data Changes** on this page.

Important

Tool selection is the boundary of what an agent can do. Instructions shape how an agent behaves; tools decide what it can touch at all. An agent can only call the tools it has been granted, and no instruction widens that. Grant changes take effect on the agent's next run.

Conversation Starters

Conversation starters are suggested prompts shown when a user opens a conversation with nothing in it. Selecting one places the text in the input, where the user can read and edit it before sending.

They are the cheapest way to teach a new user what an agent is good at. Write them as real questions someone in that role would actually ask.

Testing Before Publishing

Simulation Chat lets you use an agent while it is still in Draft, before any end user can reach it. It is available to Edge AI Administrators only.

Send prompts as a user would, then expand the reasoning steps on a response to see which data the agent queried and what it decided along the way. Simulation runs against live tenant data under your own identity and permissions, so what you see is what a user with your Innovator access would see.

Use it to confirm the agent is querying the right data and returning results in the shape you expect, before you expose it.

Reviewing Agent Data Access (Beta)



The agent Overview page includes a **Data access** row in its Configuration section. Opening it shows the schema of the Edge API connections the agent's tool bindings use: item types appear as nodes, and relationships appear as connecting lines between them. Selecting a node lists the properties available on that item type and the access levels available to this agent: read, list, add, edit, and delete.

A tabular view of the same information is also available.

Availability. The Data access row is active only when the agent has at least one enabled tool binding with a configured Edge API connection.

What the schema reflects. The schema shown reflects the Aras Innovator access of the credential used to connect: the stored API key for API Key connections, or your own account for Bearer Passthrough connections. It may not match what end users of this agent can access.

The schema graph is also reachable from the Access Points screen. Select **View Schema** from the actions menu on any connection to open the graph directly.

Use Data access to validate that the agent can reach the item types and relationships you expect before publishing.

Publishing, Archiving and Deleting

An agent has two independent states, and it helps to keep them apart.

- **Publish state** is **Draft** or **Published**, and it controls whether end users can reach the agent. Only Published agents appear to Edge AI Users. Publishing is an explicit action you take once the agent has an Access Point and its tools selected and saved. Three actions change an agent's publish state, and each shows a confirmation dialog before the change takes effect.
 - **Publish.** The dialog names the agent and states that Edge AI Users on its enabled surfaces gain access immediately. Confirm to publish; cancel to leave the agent in Draft.
 - **Move to Draft.** The dialog names the agent and states that it is removed from all end-user surfaces. Only Edge AI Administrators can reach it until it is published again. Confirm to move to Draft; cancel to leave the agent published.
 - **Save while published.** Saving edits to a published agent opens a dialog stating that the save will move the agent to Draft, removing it from end-user surfaces until it is republished. Confirm to save and move to Draft; cancel to leave the configuration form open with edits intact.



- **Activity status** is separate and describes the agent's availability: active, disabled, error, or archived. An agent is reachable by end users only when it is both active and published. Archive is a soft delete. An archived agent disappears from end users whatever its publish state, and Restore brings it back. Use Archive when an agent is out of service but you may want it again.
- Delete is permanent and cannot be undone.
- All of these actions are restricted to Edge AI Administrators. See **Roles and Access** on this page.

Governed Data Changes

Agents do not only answer questions. They create records, update properties, move items through lifecycle states, and delete items. They do none of it without a person saying yes.

The principle is simple: people act through AI within the rights they already have.

What an Agent Can Change

An agent with the right tools can:

- create a record
- update properties on an existing record
- change an item's lifecycle state
- delete an item

Two Layers of Governance

Before an agent can change anything, two decisions must have been made, in two different places.

- **In Edge API.** The API must expose a write action, Add, Edit or Delete, on at least one ItemType. An API that exposes only read actions cannot be used to change data by any agent, under any configuration. See [Building your API schema](#) .
- **On the agent.** When you activate an API for an agent, you select which of its tools the agent may use. A write action becomes available to that agent only once you assign the corresponding tool.

Both must be true. Either one on its own leaves the agent read-only.

The approval flow travels with the tools themselves. A tool that changes data carries the



human-in-the-loop validation, so the approval step cannot be separated from the operation it protects.

Important

The two layers give you controls at different scopes. Restricting write actions in Edge API applies to every agent and every user at once. Withholding a write tool from an agent applies to that agent alone. Use the API layer for organization-wide policy, and the agent layer for per-agent scope.

Approval Is Not Optional

Human approval is a platform invariant, not a setting. The agent reasons, retrieves and proposes freely. Before any operation that changes data, it stops and waits for the person it is working with.

This holds for every agent, every operation, and every user. There is no configuration that commits a change automatically.

How Approval Happens

The request for approval arrives inline in the conversation, as a card showing three things: the action the agent intends to take, the item it will act on, and the data it will write. The user approves or rejects it there, in the conversation, with no separate screen and no approval queue to visit.

Rejecting makes no change and ends the action cleanly. The conversation carries on, so the user can correct the instruction and let the agent propose something different.

What Approval Authorizes

Approval authorizes the attempt, not the outcome.

The write executes through Edge API, and Innovator decides whether it succeeds. Permissions, lifecycle maps and workflow rules all still apply. If Innovator refuses the operation, nothing is written and the agent reports the reason back in the conversation.

Where the Access Point uses End User authentication, the write carries the approver's own Innovator identity. Approving an agent's proposal is then no more permissive than that person making the



change by hand, and an agent cannot use approval to exceed the rights of whoever gave it.

Attribution and Audit

A change commits under the identity the Access Point authenticates with.

With End User authentication, the change is committed as the person who approved it and appears in Innovator's history as their change. With API Key authentication, it commits as the user the key belongs to, so Innovator records that account rather than the approver.

Write operations are logged with the same detail as reads: item type, item ID, user and timestamp. Your existing Innovator history and audit trail remain the record of what happened to your data, whether the change came through an agent or through the interface.

Important

Authentication mode decides attribution. Where changes must be traceable to a named person, and where a user should never be able to change more through an agent than by hand, bind write-capable agents to Access Points that use End User authentication. See Access Points.

Delegated Work

Where an agent hands part of a request to a sub-agent, and that sub-agent proposes a change, the approval surfaces in the user's own conversation.

Work moves between agents. Approval never moves away from the person.

Access Points

An Access Point is a registered Edge API connection. It is how agents reach your Innovator data, and what the connected API exposes is the outer boundary of what any agent bound to it can see or change.

Access Points are tenant-scoped and reusable. Register one and any number of agents can use it.

Registering an Access Point

You provide a name and description, the Edge API endpoint URL, and the authentication mode. Name and describe the connection for the colleague who inherits it, not for yourself: which Innovator



instance it reaches and what it is for.

Choosing an Authentication Mode

This is the most consequential decision on an Access Point, and it is worth deciding deliberately rather than accepting a default.

- **End User** propagates the identity of the signed-in person through to Edge API. Every request, for schema and for data alike, runs under that person's own Innovator permissions, so nobody can retrieve or change data through an agent that they could not reach directly in Innovator. Changes are attributed to them. Nothing is stored on the Access Point, so there is no credential to rotate and nothing that could outlive a person's access. Each Edge API used with this mode also requires the Edge AI CIAM audience value (`b832fca416424121bc28a7d3a3a832f2`) in its **Allowed audience** field; see [Getting Started](#) on this page for the prerequisite step.
- **API Key** authenticates with a static credential that you create in Edge API and enter here. Everything runs under the permissions of the user the key belongs to, and changes are recorded against that account rather than the person who approved them. Choose the key deliberately: it defines both what agents bound to this Access Point can reach and the account your audit trail will show. See [Managing API keys and Access tokens](#) for creating and rotating keys.

End User authentication requires the connected Innovator instance to have a valid trusted domain certificate, so that Edge API can identify the requester.

Important

Use End User authentication for any agent people talk to. It keeps Innovator's permission model intact end to end, it makes every change traceable to the person who approved it, and it leaves you no credential to manage. See [Governing Data Changes](#).

Testing a Connection

Where an Access Point uses API Key authentication, you can test the connection and the platform reports the result. Use it after registering the connection, and again whenever you rotate the key, to confirm the endpoint is reachable and the credential is accepted.



Before you publish an agent on a new connection, also send a few prompts in Simulation Chat. That exercises the connection the way an agent will, against the item types you actually expect people to ask about. See **Agents** on this page.

What the Connected API Must Expose

Some capabilities depend on specific item types, properties and actions being exposed in Edge API. Where they are missing, the agent tells the user it could not retrieve the data rather than failing silently.

Capability	Required in Edge API
Changing data of any kind	An Add, Edit or Delete action on the relevant ItemTypes
Version and generation history	The <code>GetItemAllVersions</code> action and the generation property on the relevant ItemTypes, and the History Container and History ItemTypes exposed with <code>config_id</code> , <code>item_config_id</code> , <code>item_keyed_name</code> , and <code>source_id</code> .
Queries filtered by owner, creator or assignment	The User ItemType with <code>first_name</code> , <code>keyed_name</code> , <code>last_name</code> and <code>login_name</code> , plus the Alias relationship; and the Identity ItemType with <code>keyed_name</code> and <code>name</code> , plus the Member relationship.

See [Building your API schema](#) for how to expose them, and **Configure Edge API for Item Version and History Queries** and **Configure Edge API for User and Identity Queries** in the [Get Started](#) guide for the detailed configuration.

Because agents read the schema at runtime, extending what an API exposes reaches your agents on their next request.

Removing an Access Point

An Access Point cannot be removed while an agent is using it, so retiring one is a sequence rather than a single action.

Deal with the dependent agents first. Either point each one at a replacement Access Point, or reconfigure it so it no longer needs this connection. Once no agent uses it, remove the Access Point.

Work in that order because your users are on the other end of those agents. Reconfiguring the agents first means nobody finds an agent unable to answer because the connection underneath it has gone.

Configuring the APIs Themselves



Access Points connect to APIs; they do not define them. The Access Points list links through to the API Management System, where the APIs and their schemas are configured. If you have not created an API yet, start there.

Models

A model configuration tells InnovatorEdge AI which language model to use for reasoning and how to reach it. Every agent uses one.

Aras-Managed and Tenant-Owned Models

Aras provides a model configuration to every tenant, maintained by Aras and available to you read-only. Agents use it unless you decide otherwise, so a new tenant has a working model from the first day with nothing to configure and nothing to procure.

Alongside it you can register your own. A tenant-owned configuration is yours to edit, test and remove.

Enabling and Disabling Models

The Models screen shows an enabled or disabled toggle for each model configuration. The toggle is visible in the list without opening the model, and also on the model detail page.

A disabled model is removed from the selection list in the agent editor and cannot be assigned to any agent.

Disabling a model that is already assigned to one or more agents produces a warning before the change takes effect. The platform moves each affected agent to **Draft** and clears its model assignment. Selecting a different model and republishing restores the agent's availability to end users.

Registering Your Own Model

Register a model from the Models screen. You provide a name, the provider, one or more deployment names, and the credential. Where the provider requires them, you also provide an endpoint and an API version. The first deployment you add is the default for that configuration.



Supported providers are Azure OpenAI and Microsoft AI Foundry, OpenAI, Anthropic, and Google. An OpenAI-compatible endpoint, such as a self-hosted deployment served through Ollama or vLLM, registers under the OpenAI provider type with a custom endpoint.

There are three common reasons to do this: your organization already has a model subscription it wants to use, a particular model suits your requirements for capability, cost or latency, or you want inference to happen on infrastructure you control.

Important

Registering your own provider is how you decide where inference happens. Prompts and responses are processed by the endpoint you nominate, inside the environment you run it in.

Testing a Model Configuration

Test the configuration to confirm the platform can reach the provider and that the credential is accepted. The result is reported back to you.

Before you publish an agent on a newly registered model, also send a few prompts through Simulation Chat. That exercises the model the way an agent actually uses it, including the agent's reasoning settings, which a connection test does not cover. See **Agents** on this page.

Assigning a Model

Set a tenant default so that every new agent starts on an approved model, and assign a different model to an individual agent in the agent editor where that agent has different needs. See **Agents**.

The Aras AI Assistant uses the Aras-provided configuration unless you assign it another one. If you do, it stops receiving Aras updates to its model configuration, while everything else it inherits from Aras continues to arrive. See **Agents**.

Skills

A skill is a reusable bundle of instructions that teaches an agent how to handle a particular kind of request: the query patterns your organization uses, the rules that apply to a domain, the conventions your data follows. Skills are how an agent learns your organization rather than PLM in general.



Skills are tenant-scoped. Write one and it is available to every agent in your tenant, so a pattern you get right once can be reused everywhere.

Aras-Managed Skills

Aras provides a set of skills covering common PLM work. They are available to every agent, read-only, and Aras extends and refines them with each release.

You can disable an Aras skill on an agent you own, and it stays disabled through later updates. Your decision is not undone the next time Aras ships an improvement.

Creating a Skill

A skill has three parts.

- **Name.** Required, and unique within your tenant. It is how you identify the skill when binding it to an agent, so name it for the situation it applies to rather than for the mechanism it uses.
- **Description.** A short summary of what the skill teaches. This is what you and your colleagues read when deciding whether an agent needs it, so be specific about when it applies.
- **Explanation.** The instruction text itself, up to 10,000 characters, and this is what the agent reads.

Writing a Skill That Works

The explanation is written for the agent, not for a person, and the difference matters.

Teach a pattern rather than a fact. A skill that says which property holds the approving engineer, and how your organization words that question, will keep working as your data grows. A skill that lists this quarter's programs will not.

Keep one skill to one concern. Small skills combine; a large one that covers three domains has to be bound to every agent that needs any of the three.

Use the room you have. Ten thousand characters is enough for worked examples, and examples teach an agent more reliably than abstract rules. Show a question your users actually ask and the shape of the answer you expect.

Say what to avoid as well as what to do. Naming a mistake you have seen an agent make is often the most valuable line in a skill.



Then test it. Bind the skill to a draft agent and work through real questions in Simulation Chat, expanding the reasoning steps to see whether the agent actually followed it. See **Agents**.

Important

A skill changes how an agent approaches a problem. It cannot widen what the agent can reach. Data access comes from Access Points and capability comes from tools, so a skill can never grant an agent access to something it was not given. Write skills freely; they cannot open a door.

Duplicating a Skill

Duplicate an existing skill when you need a variant. You get an independent copy, opened for editing, and changes to either one leave the other alone.

This is the practical way to adapt a skill for a second domain, or to try a rewrite while the original stays in service.

Removing a Skill

If a skill is bound to agents, you are shown which ones before the deletion goes ahead.

Read that list. Removing a skill changes how those agents behave the next time someone talks to them, and the agents will not tell you what they lost.

MCP Servers

MCP Servers is the tool registry for InnovatorEdge AI. Every tool an agent can call is registered here, whether Aras provides it or your organization adds it.

The Model Context Protocol, MCP, is how an agent talks to a tool. InnovatorEdge AI applies it uniformly: the Aras tools that reach your Innovator data are deployed as an internal MCP server and speak the same protocol as anything you register alongside them. There is no separate mechanism for built-in tools and third-party tools.

That uniformity is the reason this screen matters more than its name suggests. It is the complete inventory of what agents in your tenant are capable of doing, in one place, under one governance model.

The Aras-Managed Server



Aras deploys its Innovator tools as an internal MCP server, present in every tenant and maintained by Aras. These are the tools that retrieve items, search, walk BOM and where-used relationships, read lifecycle state, and, where you have enabled them, change data.

Within this server the tools are grouped by the Access Point each one acts through, so you can see which Innovator connection a given tool reaches. Aras extends this set with each release, and new tools appear in the registry without any work from you.

The Aras-managed server is also the only one that can act as the signed-in user. That is what allows Innovator's permission model to govern every query and every change an agent makes. See **Access Points**.

Registering an External MCP Server

Register a server to bring its tools into the registry. You provide a name, the server's URL, the authentication mode, and the credential where the server requires one.

Write a description. The next administrator needs to know which system this is and why an agent talks to it, and a URL alone will not tell them.

Authentication for External Servers

An external server is registered either with **no authentication**, for a public server that requires no credential, or with an **API key** that you supply.

Either way, the server is reached with what you configure on it rather than with the identity of the person using the agent. End-user identity is propagated to the Aras-managed server, and that is where Innovator's permissions apply.

Important

This asymmetry is the thing to understand before you register anything. Tools on the Aras-managed server run as the signed-in user, so Innovator's permissions govern them. Tools on an external server run with the credential you configured, so every person using that agent acts on the external system with those same rights. Grant external tools narrowly, and scope the credential to exactly what the agent legitimately needs.

Discovered Tools



Once a server is registered, the platform reads the set of tools it declares. That list belongs to the server and you do not edit it here. What you decide is which of those tools any given agent may use.

Testing an MCP Server

Test the server to confirm it is reachable, that the credential is accepted, and that tool discovery succeeds. Do this before binding any of its tools to an agent, so a discovery problem surfaces while you are configuring rather than while someone is asking a question.

Granting Tools to Agents

Tools are selected per agent, in the agent editor, where the registry appears grouped by server. An agent can call only the tools it has been granted, and a change to its grants takes effect on its next run. See **Agents**.

Reviewing the registry against your agents is worth doing periodically. It is the one view that answers what the agents in this tenant can actually do, which is a different question from what they were built to do.

Removing an MCP Server

Where agents use tools from the server, you are shown which agents are affected before the removal goes ahead.

Reconfigure those agents first, as you would when retiring an Access Point. An agent that loses a tool it relied on simply stops being able to do that part of its job, and it will not announce the loss.

Troubleshooting and Known Behaviors

Most reports that reach an administrator come down to configuration rather than to anything going wrong. This chapter starts with the checks that resolve the majority of them, then covers behaviors worth knowing about before a user reports them.

Start Here

Three questions settle most cases.

- **Is the agent published?** Only published agents are visible to Edge AI Users, and editing a published agent returns it to Draft. An agent that worked yesterday and is missing today has usually been edited and not published again. See **Agents**.



- **Does the user have an Edge AI role?** Without one they reach neither the assistant nor the platform. See **Roles and Access**.
- **Does the connected API expose what the question needs?** An agent can only work with the item types, properties and actions the Edge API exposes. This is the cause behind most "the agent cannot answer this" reports. See **Access Points**.

Common Symptoms

Symptom	Likely cause	What to do
A user cannot see an agent that exists	The agent is in Draft or archived, or the person has no Edge AI role	Publish or restore the agent, or assign the role. See Agents and Roles and Access .
An agent that was working is no longer available to users	It was edited, which returned it to Draft	Publish it again. See Agents .
The agent says it cannot find an item type you know exists	The item type is not exposed in the connected Edge API	Expose the item type and the properties the question needs. See Access Points .
Answers are missing records that should be included	On an Access Point using API Key authentication, the key's user cannot see those records	Use a key whose user can see the full schema, or switch the Access Point to End User authentication. See Access Points .
Two people get different answers to the same question	The Access Point uses End User authentication, so each answer is scoped to that person's Innovator permissions	This is working as intended. Compare their Innovator access. See Roles and Access .
Questions about who created, owns or modified an item do not resolve	The User and Identity item types are not fully exposed	Expose them with the properties and relationships listed in Access Points.
Version or history questions do not resolve	The history item types and the version action are not exposed	Expose them as listed in Access Points.
The agent will not change data even though the user approves	The API exposes no write action for that item type, or the agent has no write tool	Check both layers. See Governed Data Changes .
A connection test fails	The endpoint is unreachable or the credential is no longer valid	Correct the endpoint or rotate the key, then test again. See Access Points .
End User authentication fails for every user	The Edge AI CIAM audience value is not registered in the Allowed audience field of the Edge API	Add b832fca416424121bc28a7d3a3a832f2 to the Allowed audience field on each Edge API used with this Access Point. See Access Points and Managing API keys and Access tokens .
An agent cannot use an external tool	The tool is not granted to that agent, or the MCP server is unreachable	Grant the tool, or test the server. See MCP Servers .

Known Behaviors

- **Coverage varies by Aras application.** The assistant works against your schema without hardcoded item definitions, so any item type exposed through Edge API can in principle be queried. Item types belonging to some Aras applications, including Variant Management, Technical Documentation and Simulation Management, depend on schema exposure that is not always complete. Validate the questions you care about against those applications before you tell those teams the assistant covers them.
- **Results are bounded.** Retrieval is bounded per call, and large aggregations may be truncated. When a result is capped or truncated, the assistant says so and asks the user to narrow the



request, so a bounded answer is visible rather than silent.

- **Calculations report their inputs.** For a calculation across a set, the assistant states how many records it included, because records with incomplete data are excluded. Treat that count as part of the answer, and expect questions about it.
- **"History" is ambiguous in a short prompt.** Asking for the history of an item may return version and generation information rather than the full audit trail. Asking for the full history, or the action history, resolves it, and so does clarifying in the next message.
- **Some item types show an identifier in the context chip.** The chip above the chat input occasionally shows an item type and identifier instead of a readable name. The context itself is correct; only the label is affected.
- **Context follows the active item in Innovator.** When several item tabs are open, confirm the chip is pointing at the record the user means before relying on the answer. Removing the chip and asking again with the intended record open is the quickest way to be certain.
- **A manually stopped response ends that turn completely.** If a user stops the assistant mid-answer, any skill that had loaded during that turn is not carried into the next request in the same conversation. Starting a new conversation restores normal behavior.

Getting Support

If a problem persists, raise it through the Aras Subscriber Portal with:

- the release number, from the About panel
- your tenant ID, from the profile menu
- the name of the agent
- the name of the Access Point and the Edge API endpoint it uses
- the question the user asked, and what came back

Those details let support reproduce the case against your configuration rather than a generic one, which is usually the difference between one exchange and several.

Important

A support case leaves your organization. Review what you attach before you send it, and sanitize or anonymize anything covered by your intellectual property or confidentiality obligations. Conversation transcripts, error messages and screenshots often carry part numbers, program names, supplier names and specifications that support does not need. What support does need is the shape of the problem: the item type, the property, the action attempted and the response received. Replacing real identifiers with placeholders keeps the case diagnosable without exposing the design.

