

Oracle AI for Fusion Applications

How do I use AI Agent Studio?

26D



Oracle AI for Fusion Applications
How do I use AI Agent Studio?

26D

G59779-01

Copyright © 2025, 2026, Oracle and/or its affiliates.

Author: Soniya Mathew, Rebecca Nethala, Nilay Agambagis

Contents

Get Help	i
1 Overview	1
Overview of AI Agent Studio	1
Key Capabilities of AI Agent Studio	1
Components of AI Agent Studio	2
Large Language Models (LLMs) in AI Agent Studio	3
2 Access Requirements for AI Agent Studio	5
Access Requirements for AI Agent Studio	5
Provide Access to Configure AI Agents in all Products	7
Provide Access to Configure AI Agents in Oracle Fusion Cloud Human Capital Management	8
Provide Access to Configure AI Agents in Oracle Fusion Cloud Supply Chain & Manufacturing	12
Provide Access to Configure AI Agents in Oracle Fusion Cloud Procurement	17
Provide Access to Configure AI Agents in Oracle Fusion Cloud Financials	22
Provide Access to Configure AI Agents in Oracle Fusion Cloud CX	23
Provide Access to Configure AI Agents in Oracle Permitting and Licensing	23
3 Before You Begin	29
Choose How to Create Workflows and Agentic Applications	29
Get Started with AI Agent Studio	31
Add External REST Tool	33
Add MCP Tool	33
Add Your LLM	34
Add Email Tool	35
Add Document Tool	36
Add Email Accounts	36
Create Documents to Add to Document Tool	37
Add Connector Tool	39
4 Use Cases	41
Single Agent with Multiple Workers	41

Single Agent with Multiple Tools and a Human in the Loop	43
Multiple Agents With Multiple Tools and Delegations	44
Product-Specific Use Case Examples	47
5 Expressions in AI Agent Studio	49
Expressions	49
Variables Specific to Workflows	49
6 AI Agents	55
Create and Configure AI Agents	55
7 Use Workflow Builder	57
Create Workflows	57
Nodes in Workflows	63
Settings in Workflows	64
Configure Vector DB Writer and Reader Nodes in a Workflow	65
Configure Tool Nodes in a Workflow	68
Add a Policy Model to a Policy Node of Workflows	70
Add a Document Schema to a Document Processing Node of Workflows	72
Add a Reference Block to a Reference Node of Workflows	73
Add Multi Agent Nodes in AI Agent Studio	74
Configure a Return Node in a Workflow	74
Configure a Send Email Node in a Workflow	75
Configure Workflow Triggers	75
Add an Approval Process Channel to a Human Approval Node of Workflows	77
Add and Reuse Functions in AI Agent Studio	78
Configure an AI Guidance Node in a Workflow	79
8 Use Applications Builder	81
Overview of Agentic Applications	81
Anatomy of an Agentic Application	83
Create and Configure Workflows for Use in Agentic Applications	84
Build Agentic Applications Using App Builder	98
Run Agentic Applications from AI Agent Studio	100
9 Edit Agents, Workflows and Agentic Applications	103
Edit a Workflow	103

Edit a Supervisor Agent	104
Edit and Test Supervisor Agents Iteratively in Playground	104
Edit an Agentic Application	106

10 Debug and Test AI Agents **109**

Overview of the Debugger	109
Get Started with the Debugger	111
Create and Use Run Profiles	112
Inspect Node Results	114
Debug Switch Routing	116
Set and Use Breakpoints	117
Rerun, Step Through, or Resume a Workflow	118
Pin or Unpin a Node Output	120
Edit Node Output	121
Override Node Configuration	122
Validate Final Chat Output	123
Checklist for Publishing Workflow After Debugging	123
FAQs for Debugging and Testing AI Agents	123

11 Monitor and Evaluate AI Agents **129**

Monitor and Evaluate AI Agents	129
Monitor Agents	131
Evaluate Agents	131
Compare Evaluation Runs	132
View the Measured Value of Agents	133
Best Practices for Creating Evaluation Sets	133

12 Promote Published Agents **137**

Migrate Agentic Applications or Workflows from One Instance to Another	137
Migrate a Supervisor Agent from One Instance to Another	138

13 Use AI Studio Command Line Interface (CLI) **139**

Overview of AI Studio CLI	139
Set Up AI Studio CLI	141
Development Flow in AI Studio CLI	142
Code Assistant Usage and Examples	143

Test a Workflow or Agentic Application Created or Edited with AI Studio CLI	153
Recommended Usage Flow for AI Studio CLI	156
14 Integrate Applications and Workflows with Continuous Integration and Continuous Delivery (CI/CD)	157
Integrate Applications and Workflows with CI/CD	157
15 Allow External Access to Fusion Applications Agents	159
Enable Applications to Access Fusion Applications Agents	159
16 Collaborate with AI Agents Across Platforms Using Agent2Agent (A2A) Protocol	163
Collaborate with AI Agents Across Platforms using Agent2Agent (A2A) Protocol	163
Components in A2A	163
Authentication and Access for A2A	164
A2A API Overview	165
17 AI Agent Studio FAQs	171
Which large language models (LLMs) are currently used or supported for AI agents?	171
Which scheduled processes do I need to run to make sure the features of AI Agent Studio work as expected?	171
Can I use any agent in my workflow?	171
Can I migrate any workflow or supervisor agent from one environment to another?	172
How do I display a welcome message in a chatbot agent?	172
How do I make agents respond faster?	173
How do I send an email using AI Agent Studio?	173
How do I track the usage of AI Agents created in AI Agent Studio?	174
How can I best reduce the risk of my agent showing hallucinations?	174
How do I hide or restrict access to AI Agent Studio for specific users?	175
How can I provide access to an AI Agent for all employees?	175
Why can't I find the standard Employee role in the Security tab of Workflow Settings?	176
Can I rename the AI chat label to something other than Ask Oracle, or change the design of the chat panel?	176
What's the difference between editing a topic versus editing a system prompt?	176
What file types, limits, and processing capabilities are supported in AI Agent Studio?	176
Why do I see missing or misaligned structured document data in AI Agent Studio?	179
Why does the external REST tool only support HTTPS transactions?	180
Why is the Provider list in the LLM subtab empty?	180

Why is my agent not fetching data from the document added to it?	180
Why can't I see the imported workflows or supervisor agents in the target environment?	181
How can I move the AI Agents link to the top of the Quick Actions list?	181
Why is the Roles list in the Security tab empty?	181
How can AI agents be configured or instructed to respond in a user's preferred language?	181
Why is my agent not responding in my preferred language?	182

Get Help

In addition to the resources available in Oracle Help Center, you can explore training, connect with the Oracle community, and share your feedback.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at [Oracle University](#).

Join Our Community

Use [Cloud Customer Connect](#) to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest [ideas](#) for product enhancements, and watch events.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to oracle_fusion_applications_help_ww_grp@oracle.com.

Thanks for helping us improve our user assistance!

1 Overview

Overview of AI Agent Studio

AI Agent Studio is a design-time environment that empowers you to create, configure, validate, and deploy AI agents to meet your organization's needs.

With AI Agent Studio, you can easily extend preconfigured workflows, and build new workflows and agentic apps. AI Agent Studio is fully integrated into Oracle Fusion Cloud Applications, providing secure and seamless access to the knowledge stores, tools, and APIs of Fusion Applications. This integration enables agents to be deployed directly into the flow, ensuring an efficient process.

Key Capabilities of AI Agent Studio

Feature	Description
Agent template libraries	Use templates and natural language prompts to create or fine tune agents and workflows for common business scenarios, such as opportunity-to-quote processing and shift scheduling.
Workflow orchestration	Configure multiple agents to collaborate on multistep processes, integrating user approvals where necessary.
Agent extensibility	Change and extend existing agents in Oracle Fusion Cloud Applications by incorporating new data sources, prompts, and APIs to fit specific business needs or industry requirements.
Agentic apps	Use workflows and nodes to create agentic apps that can act as your decision-making partners.
Native integration with Fusion Applications	Directly access APIs and tools in Fusion Applications, ensuring seamless deployment of agents without complex modifications.
Third-party system integration	Connect with external systems and collaborate with third-party agents for end-to-end automation with secure API support.
Trust and security framework	Automatically applies the security configurations, policies, and access controls of Fusion Applications, ensuring compliance with enterprise security standards.
Validation and testing tools	Use built-in tools to make your agents reliable, repeatable, easy to explain, and secure by verifying AI-driven flows before deployment.

Components of AI Agent Studio

Here are the main components in AI Agent Studio:

Component	What It Does	Example
Workflow	Consists of a single agent or a group of agents that follow a structured sequence of steps or actions to accomplish a specific business task or answer a user query. The workflow is the component that can be deployed for use. - Consists of conversation logic, system integration, and user support flow. - Defines how the agent acts for a particular use case — what to do, when to do it, and how to respond based on user inputs or back-end system data.	A recruitment agent (that might consist of multiple agents and nodes) schedules interviews, screens resumes, calculates salary, and generates offers, based on policies and approvals.
Agents	Leverages a large language model to reason, create action plans, and interact with users to gather information and take direction. On behalf of users, the AI agent can do tasks that enhance productivity, efficiency, and the overall user experience. An agent must be added to a workflow, so that it can be deployed for use. AI agents can be categorized into various types. - User-proxy agent: Acts on behalf of a business user to provide input to another agent or group of agents. It's sometimes referred to as a conversational agent. - Supervisor agent: Orchestrates the use of agents within an agentic flow. - Specialist or utility agent: Focuses on a specific role or expertise and can be skilled in using a particular tool. AI agents can also have one or more of these characteristics: - Persona-based agent: Represents a specific role, such as benefits administrator, customer service representative, and finance administrator. - Tool user: Uses technology-related tools, such as calculators, web search queries, document embedding, and calendar schedulers. - Task-oriented agent: Understands their assignment or task, as a single agent or as part of a multiagent flow.	A scheduling or calendar agent that manages your workday by following your instructions. It can accept new calendar invites and propose alternative times when needed.
Topics	Defines the areas of expertise through instructions that set the boundaries and constraints for agent conversations and abilities.	An employee benefits agent can contain topics such as Health Savings Account (HSA), retirement benefits, and stock plans.
Tools	Defines the additional utilities an agent can use to accomplish a task. One or more tools are assigned to agents, and they're reusable among agents.	- Calculator - Connector - Email

Component	What It Does	Example
		• Business object • User query • Document retrieval tool for retrieval-augmented generation (RAG) • Deep Link • External REST
Instructions	Specifies the natural language rules that define the rules or conditions applied to a given topic. Instructions are part of the prompts that are sent to the underlying large language model. They can also contain guidelines and guardrails that set the parameters of an agent response.	Instructions for the payroll deduction topic: Make sure you've information regarding the number of dependents either by asking the user or querying the system. If you don't know the answer, don't make up a response.
Testing	Enables administrators to test the workflow design, ensure correct tool, topic, and instruction configurations, and validate reasoning and sources cited by the agent.	Provide example responses to a series of questions a user would likely ask an agent and details about how the agent arrived at its response.

Large Language Models (LLMs) in AI Agent Studio

AI Agent Studio supports LLMs for workflows and nodes.

You can select the model considering the model type that best matches the capability, performance, and cost requirements of your workload. AI Agent Studio provides the following model types:

- **Basic:** Supports standard workloads for which cost is more important than peak capability. Example workloads include summarization, question answering, retrieval-augmented generation (RAG), and lightweight reasoning.
- **Balanced:** Supports workloads that require a balance of quality, latency, and cost. Example workloads include RAG, tool use, multistep reasoning, and agent orchestration.
- **Frontier:** The highest-capability models, designed for complex and long-running workloads. Example workloads include autonomous execution, deep research, complex reasoning, and policy model creation.

Models Available in AI Agent Studio

You can review the models available in AI Agent Studio, including configured Bring Your Own LLM (BYOLLM) models, from the Credentials tab in AI Agent Studio. Open the **LLM** subtab to view the Oracle-provided models and any additional models that you have added. For more information, see [Add Your LLM](#)

Note: The models available for each model type can vary by release.

How to Define a Model in a Workflow

You can define a model at the workflow level and also at the node level for the LLM, Agent, and Document Processor nodes within the workflow. The model defined at the workflow level is used by all applicable nodes in the workflow unless a different model is specified in the node settings.

- To select a model at the workflow level, go to **Workflow Settings** and select the **LLM** subtab. Select the model from the **Model configuration** list.
- To select a model at the node level, go to the settings for an LLM, Agent, or Document Processor node and select the model from the **Model** list.

Deprecated Models

As newer models become available for selection in AI Agent Studio, some older models will be deprecated and eventually retired. When a deprecated model is configured for a workflow or node, a deprecation warning is displayed. The deprecation status is also shown on the LLM Card. You can continue to use a deprecated model, but it'll become unavailable when it's retired. If you're using a deprecated model, we recommend that you select one of the available models instead.

Supported Models

The following table lists the supported models by model type and status. The model identified as default is preselected by default when you configure a model.

Supported Models

Model Type	Model	Status
Basic	GPT-OSS 120B	Supported
Balanced	GPT-4.1 mini	Deprecated
Balanced	Gemini 3.1 Flash-Lite	Supported
Balanced	GPT-5 mini	Deprecated
Balanced	GPT-5.6 Luna	Default
Balanced	GPT-4o mini TTS	Supported
Balanced	GPT-4o Transcribe	Supported
Balanced	GPT-4o mini Transcribe	Supported
Balanced	GPT-4o Transcribe Diarize	Supported

2 Access Requirements for AI Agent Studio

Access Requirements for AI Agent Studio

You can give access to AI Agent Studio by assigning predefined duty roles to job roles. Also, make sure to complete these prerequisites:

- Run scheduled processes to import security data
- Run scheduled process to use the builder assistant
- (Optional) Assign privilege to use external REST API tools
- (Optional) Assign duty role to create Microsoft Teams and Slack client connectors
- (Optional) Assign role to schedule workflows
- (Optional) Assign permission group to view monitored agent records
- Give users access to AI agents



Run Scheduled Processes to Import Security Data

To import resources from LDAP, and transfer the necessary information into the security tables of Fusion Applications, run these two scheduled processes sequentially.

1. Import Resource Application Security Data
2. Import User and Role Application Security Data

You must run the processes one after the other.

1. Go to **Navigator > Tools > Scheduled Processes**.
2. Click **Schedule New Process**.
3. Leave the type as **Job**.
4. Search for and select the process.
5. Submit the process.

Run Scheduled Process to Use the Builder Assistant

You can get answers to questions about developing agents, workflows and agentic apps from the builder assistant agent integrated into AI Agent Studio. Using this conversational agent, you can get answers to questions about existing agents, search for agents, tools, and topics using natural language, and receive AI powered suggestions for relevant resources.

To use the builder assistant, run the **Index AI Agent Studio Assistant Objects and Attributes** scheduled process daily.

1. Go to Scheduled Processes, and select **Schedule New Process**.
2. Leave the type as **Job**.
3. Search for and select the **Index AI Agent Studio Assistant Objects and Attributes** process.

4. Submit the process.

Assign Privilege to Use External REST API Tools

To create External REST API tools in AI Agent Studio, the Create and Edit Backends for Visual Builder Studio (ORA_FND_TRAP_PRIV) privilege must be added to the custom role assigned to the user. You can add this privilege while creating or editing a custom role.

1. Go to the Security Console.
2. To use a new custom role, create it. To use an existing custom role, search for the custom role and edit it.
3. Go to the Function Security Policies page and select **Add Function Security Policy**.
4. Add the Create and Edit Backends for Visual Builder Studio (ORA_FND_TRAP_PRIV) privilege to the role and save it.
5. Save the custom role and assign to the user.

Assign Duty Role to Create Microsoft Teams and Slack Client Connectors

To create Microsoft Teams and Slack client connectors from the Connectors tab in AI Agent Studio, the Collab Tools Duty role (ORA_DR_FAI_GENERATIVE_AI_AGENT_COLLAB_TOOLS_DUTY) must be added to the custom role assigned to the user. You can assign the duty role while creating or editing a custom role.

1. Go to the Security Console.
 2. To use a new custom job role, create it. To use an existing custom job role, search for the custom job role and edit it.
- Note:** Make sure to enable permission groups.
3. Go to the **Role Hierarchy** page. From the Roles and Permission Groups tab select **Add Role**.
 4. Search for and add the Collab Tools Duty role (ORA_DR_FAI_GENERATIVE_AI_AGENT_COLLAB_TOOLS_DUTY) duty role.
 5. Save the custom role and assign to the user.

Assign Role to Schedule Workflows

You can call a workflow using a scheduled trigger. To schedule the workflow, the Fai Batch Job Manager Duty (ORA_DR_FAI_BATCH_JOB_MANAGER_DUTY) role must be added to the custom role assigned to the user.

1. Go to the Security Console.
 2. To use a new custom role, create it. To use an existing custom role, search for the custom role and edit it.
- Note:** Make sure that permission groups are enabled for the custom role.
3. Go to the Role Hierarchy page and open the Roles and Permission Groups tab.
 4. Add the Fai Batch Job Manager Duty (ORA_DR_FAI_BATCH_JOB_MANAGER_DUTY) role.
 5. Save the custom role and assign to the user.

Assign Permission Group to View Monitored Agent Records

To view records in the **Monitoring and Evaluation** tab of AI Agent Studio, an additional permission group must be added to the custom role assigned to the user.

1. Go to the Security Console.
2. To use a new custom role, create it. To use an existing custom role, search for the custom role and edit it.

Note: Make sure that permission groups are enabled for the custom role.

3. Open the **Permission Groups** page and select **Add Permission Groups**.
4. Search for and add the `read:Generative AI Workflow Execution` permission group.
5. Add security view for the permission group.
 - a. Select the permission group added.
 - b. In the **Details** section, open the Security Views tab.
 - c. Select **Add Security Views** and add the `AllRowsRestrictedFields` security view.
6. Save the custom role and assign to the user.

Give Users Access to AI Agents

After your agents are created and ready for use, provide access for users to interact with the AI agents. For information, see [How can I give users access to AI agents?](#)

Assign Predefined Duty Roles to Job Roles

Assign predefined product-specific duty roles to the appropriate job roles, and make sure permission groups are enabled. You can give people access to configure AI agents in all or specific products.

CAUTION: Using predefined roles might account for subscription consumption irrespective of whether you purchased the cloud service or not. See [Guidance for Assigning Predefined Roles](#).

- *Provide Access to Configure AI Agents in all Products*
- *Provide Access to Configure AI Agents in Oracle Fusion Cloud Human Capital Management*
- *Provide Access to Configure AI Agents in Oracle Fusion Cloud Supply Chain & Manufacturing*
- *Provide Access to Configure AI Agents in Oracle Fusion Cloud Procurement*
- *Provide Access to Configure AI Agents in Oracle Permitting and Licensing*

Provide Access to Configure AI Agents in all Products

1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.

2. On the Role Hierarchy page, open the Roles and Permission Groups tab and add these duty roles:
 - Fai Genai Agent CX Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_CX_ADMINISTRATOR_DUTY)
 - Fai Genai Agent FIN Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_FIN_ADMINISTRATOR_DUTY)
 - Fai Genai Agent GRC Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_GRC_ADMINISTRATOR_DUTY)
 - Fai Genai Agent HCM Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_HCM_ADMINISTRATOR_DUTY)

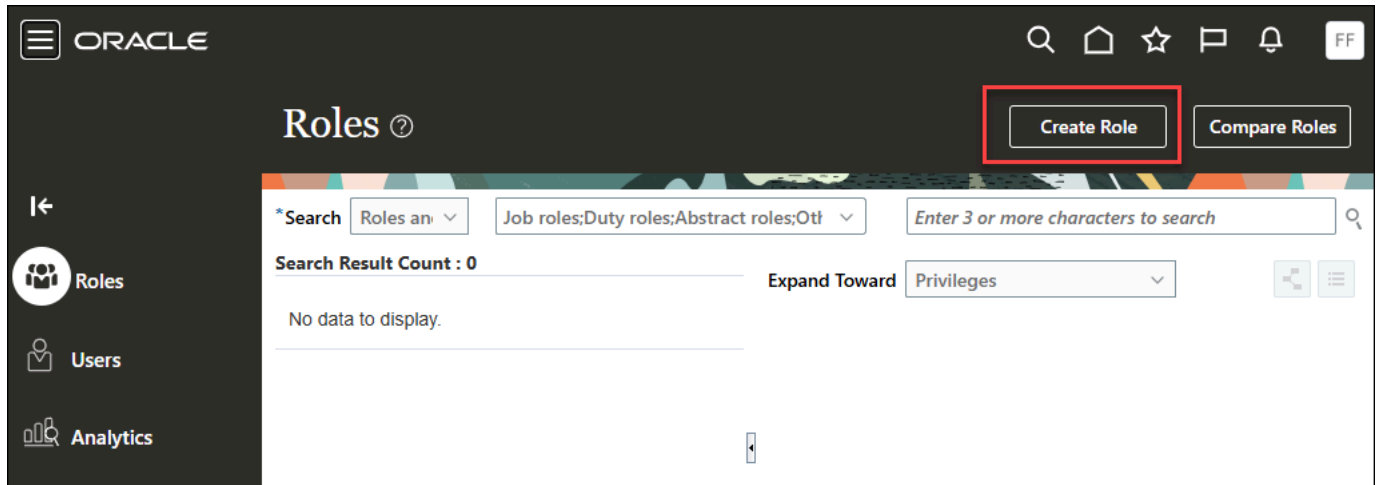
- Fai Genai Agent PRC Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_PRC_ADMINISTRATOR_DUTY)
 - Fai Genai Agent PRJ Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_PRJ_ADMINISTRATOR_DUTY)
 - Fai Genai Agent PSC Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_PSC_ADMINISTRATOR_DUTY)
 - Fai Genai Agent SCM Administrator Duty
(ORA_DR_FAI_GENERATIVE_AI_AGENT_SCM_ADMINISTRATOR_DUTY)
3. Open the Roles and Privileges tab and add the Manage All Intelligent Agents (ORA_FAI_MANAGE_ALL_AI_AGENTS) role.
 4. Save the custom role and assign it to users who want access.

Provide Access to Configure AI Agents in Oracle Fusion Cloud Human Capital Management

To give access to users without the Human Capital Management Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.



*Role Name

*Role Code

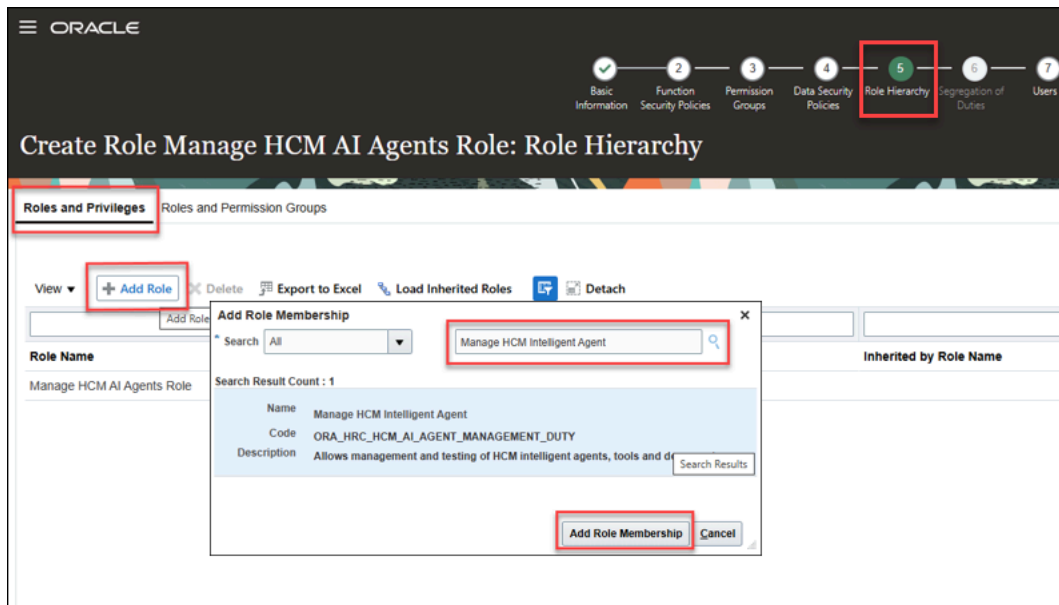
*Role Category

Enable Role for Access ☐

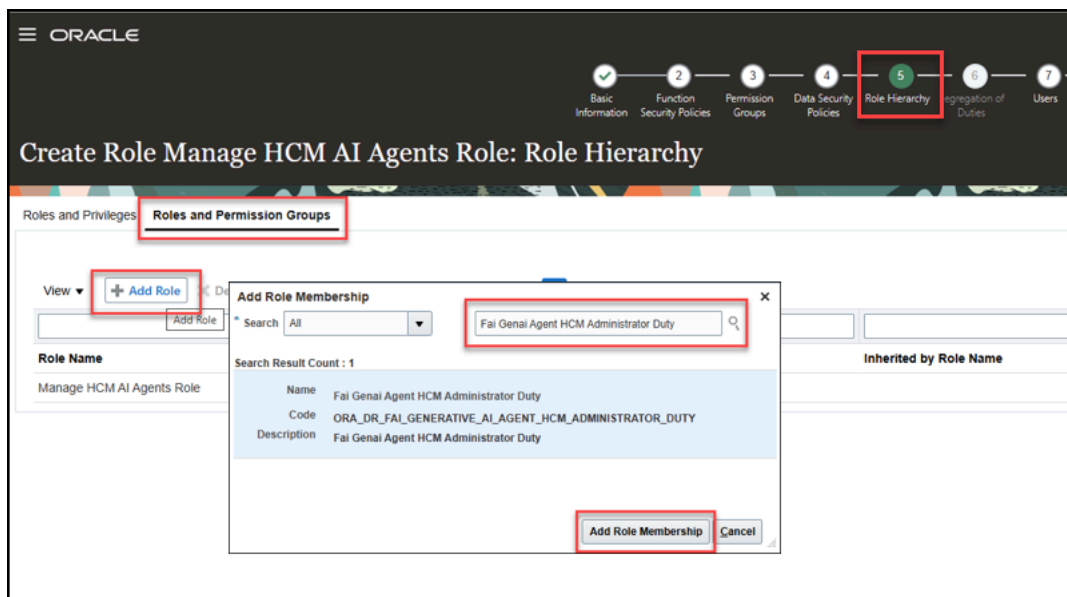
from All IP Addresses

Description

2. Go to the Role Hierarchy page.
 - Open the Roles and Privileges tab, and add the Manage HCM Intelligent Agent (ORA_HRC_HCM_AI_AGENT_MANAGEMENT_DUTY) duty role.



- Open the Roles and Permission Groups tab, and add the Fai Genai Agent HCM Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_HCM_ADMINISTRATOR_DUTY) role.

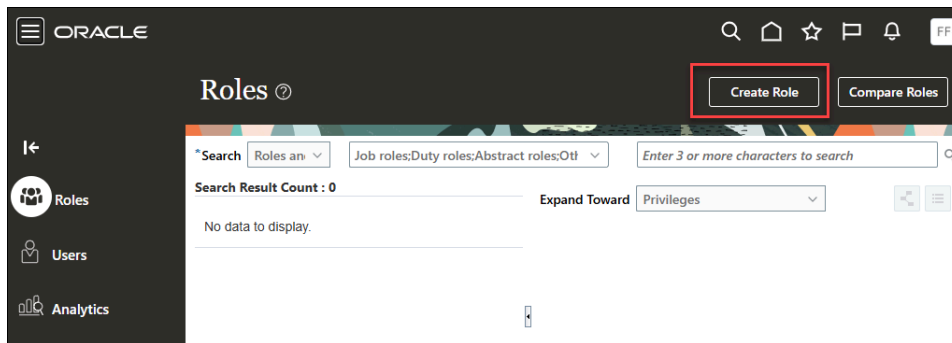


3. Save the custom role and assign to the appropriate job roles.

To give access to users with the Human Capital Management Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.



*Role Name

*Role Code

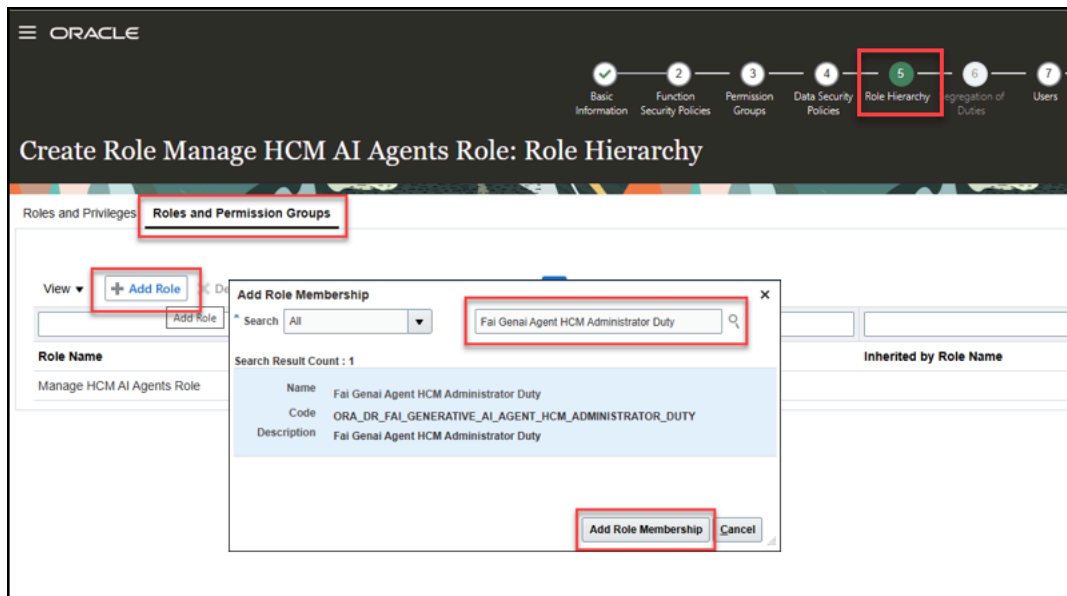
*Role Category

Enable Role for Access ☐

from All IP Addresses

Description

2. On the Role Hierarchy page, open the Roles and Permission Groups tab, and add the Fai Genai Agent HCM Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_HCM_ADMINISTRATOR_DUTY) role.



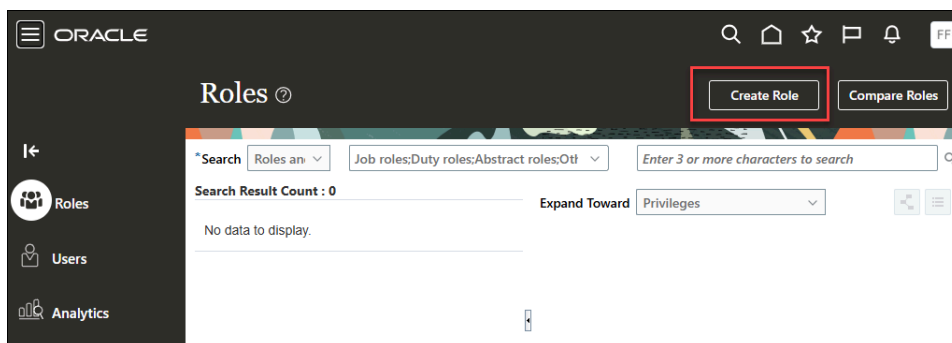
3. Save the custom role and assign to the appropriate job roles.

Provide Access to Configure AI Agents in Oracle Fusion Cloud Supply Chain & Manufacturing

To give access to users without the Supply Chain Application Administrator Job Role:

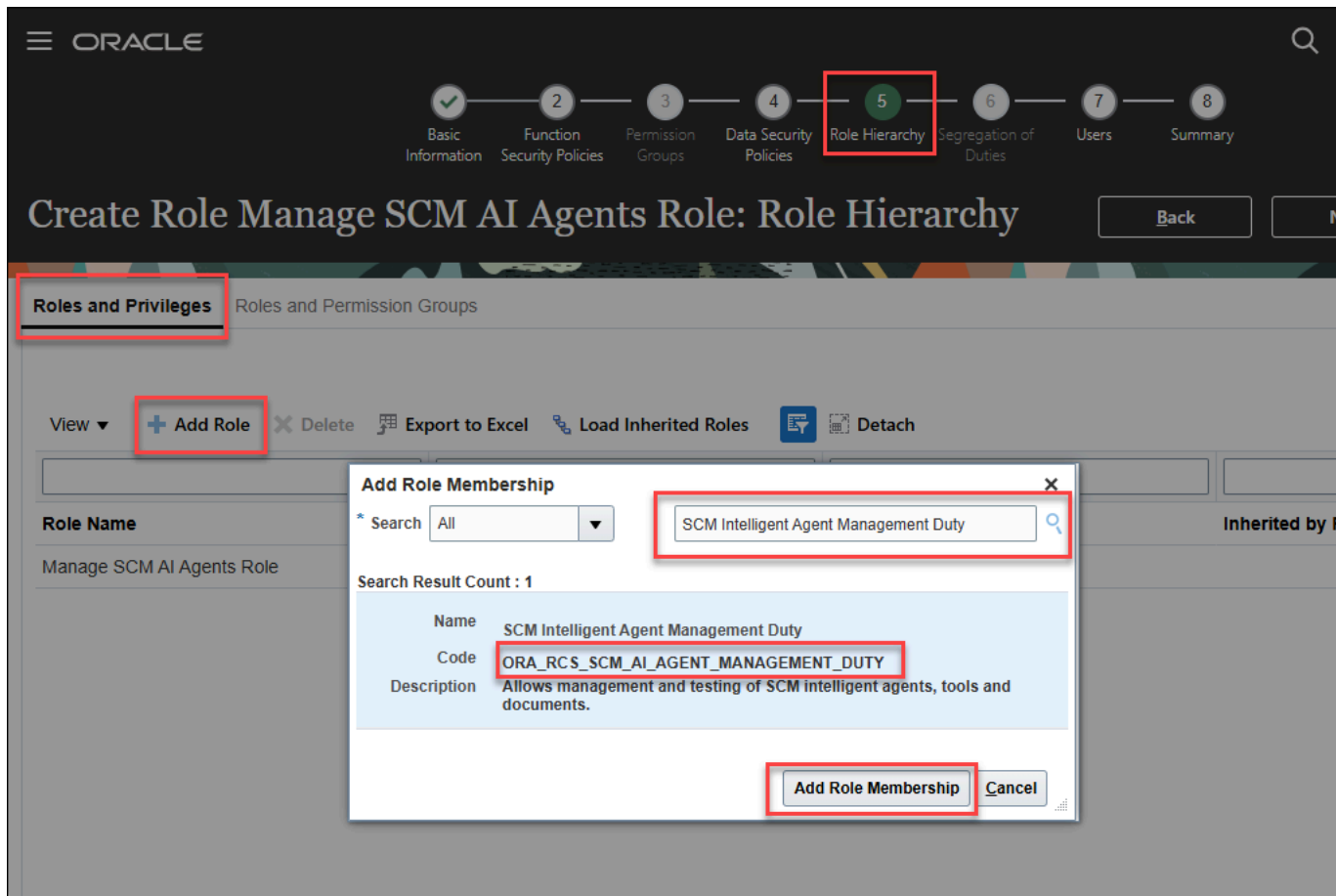
1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.

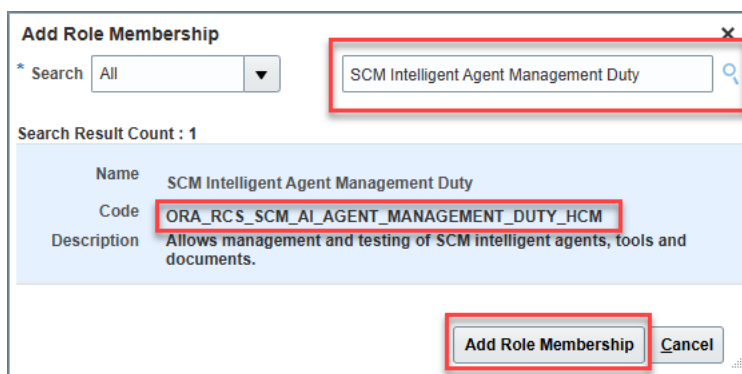


*Role Name	Manage SCM AI Agents Role
*Role Code	MANAGE_SCM_AI_AGENTS_ROLE
*Role Category	Common - Job Roles ▼
	Enable Permission Groups
Enable Role for Access	☐
from All IP Addresses	
Description	

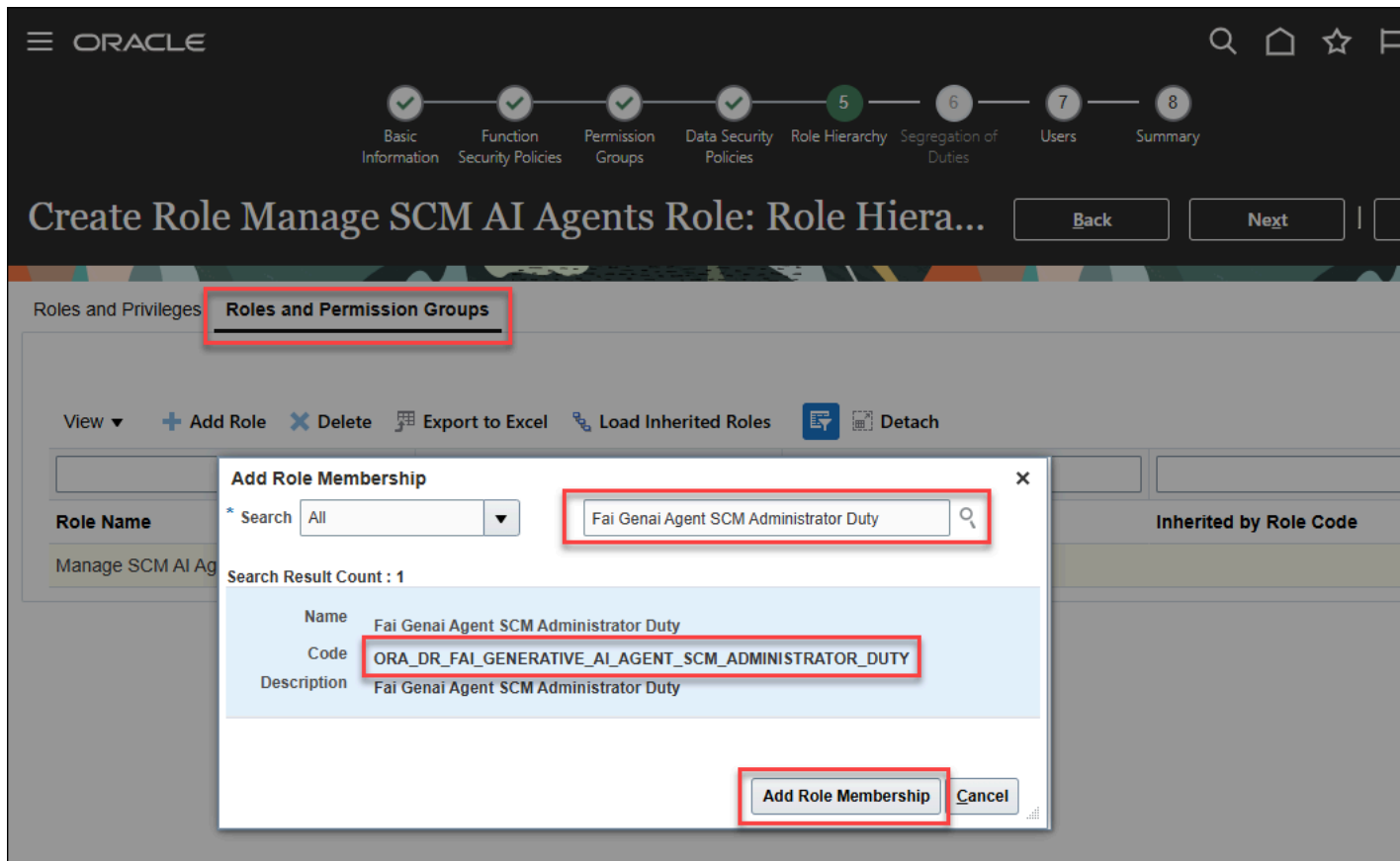
2. On the Role Hierarchy page, open the Roles and Privileges tab and add these roles:
 - SCM Intelligent Agent Management Duty (ORA_RCS_SCM_AI_AGENT_MANAGEMENT_DUTY)



- SCM Intelligent Agent Management Duty (ORA_RCS_SCM_AI_AGENT_MANAGEMENT_DUTY_HCM)



3. Open the Roles and Permission Groups tab and add the Fai Genai Agent SCM Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_SCM_ADMINISTRATOR_DUTY) duty role.

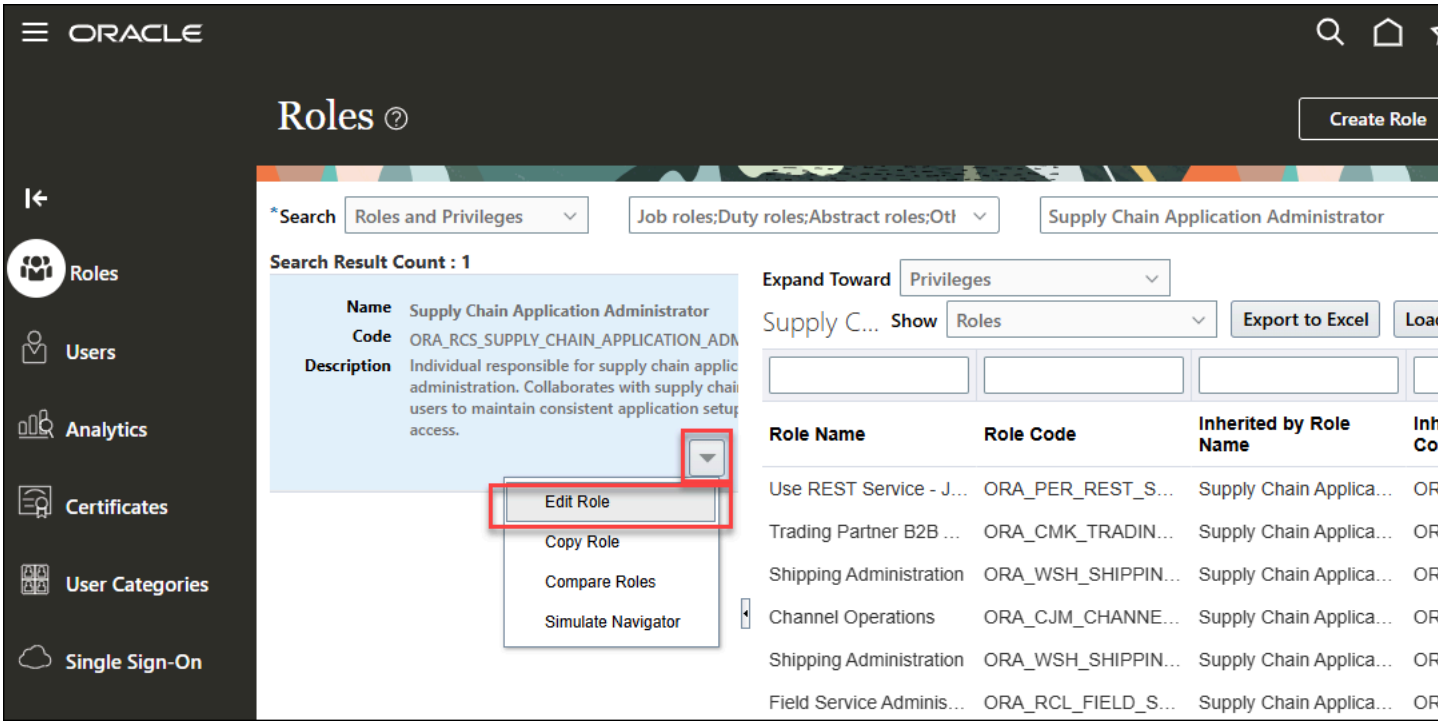
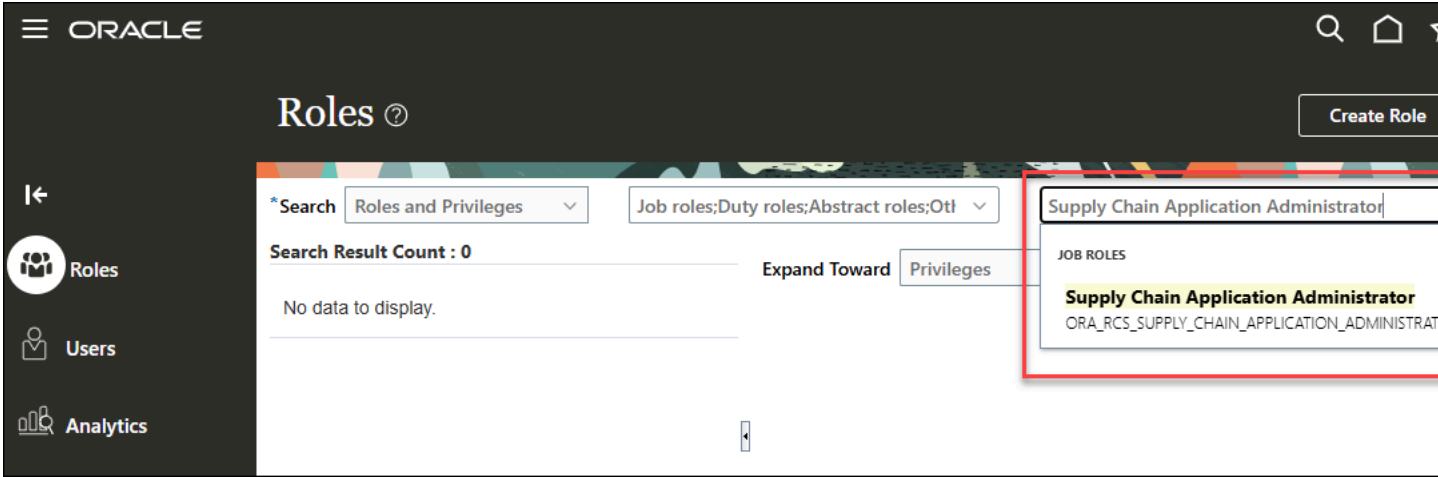


4. Save the custom role and assign this custom role to the appropriate job roles.

To give access to users with the Supply Chain Application Administrator Job Role:

1. Go to the Security Console.

2. Search for the Supply Chain Application Administrator (ORA_RCS_SUPPLY_CHAIN_APPLICATION_ADMINISTRATOR_JOB) job role, and edit it.



3. Enable permission groups and save the job role.

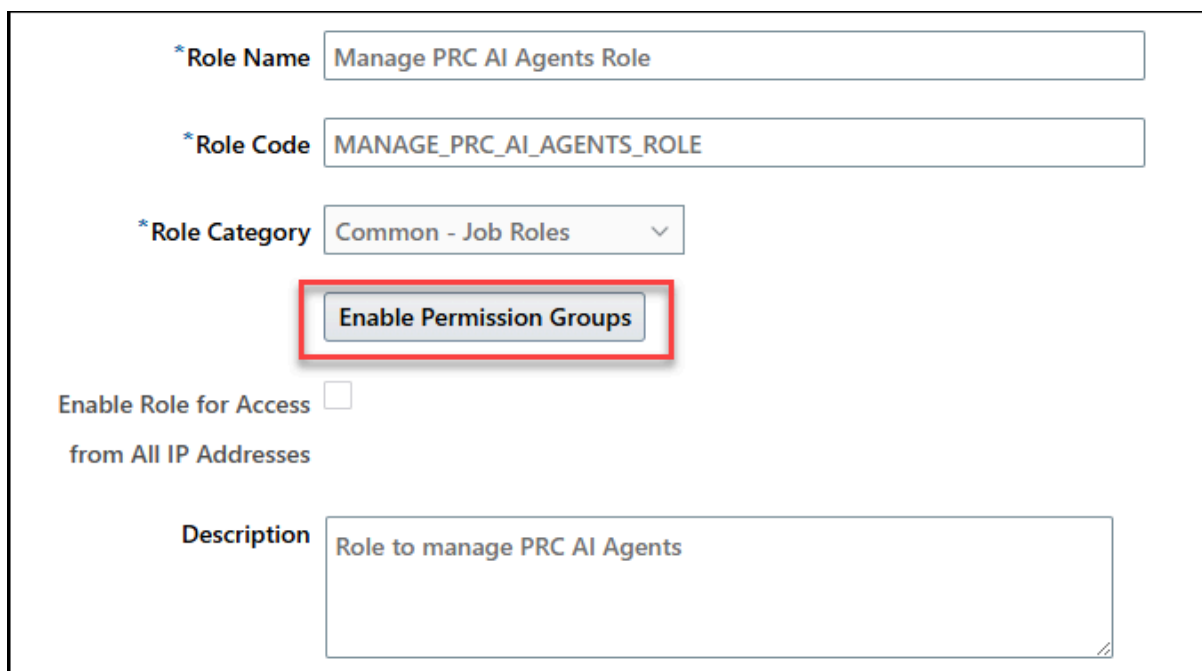
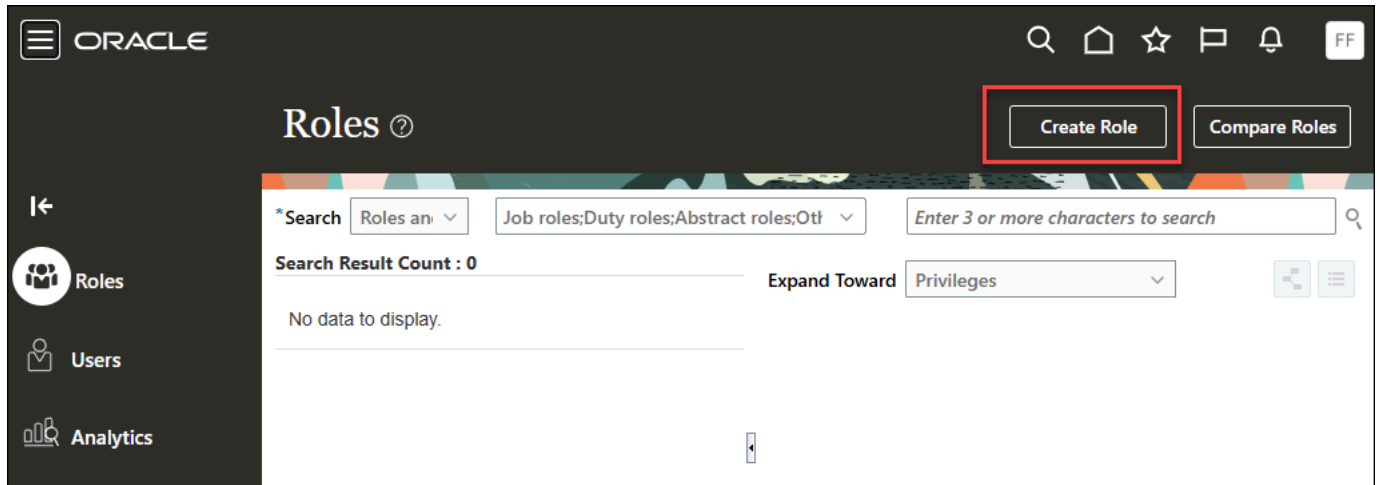
The screenshot shows the Oracle AI Agent Studio interface for editing a role. At the top, there is a navigation bar with the Oracle logo and a progress indicator with five steps: 1. Basic Information, 2. Function Security Policies, 3. Permission Groups, 4. Data Security Policies, and 5. Role Hierarchy. The main heading is 'Edit Role Supply Chain Application Administrator: Basic'. Below this, the role details are displayed: Role Name: Supply Chain Application Administrator, Role Code: ORA_RCS_SUPPLY_CHAIN_APPLICATION_ADMINISTRATOR_JOB, and Role Category: SCM - Job Roles. A red box highlights the 'Enable Permission Groups' button. Below this button, there are checkboxes for 'Predefined Role' (checked) and 'Enable Role for Access from All IP Addresses' (unchecked). At the bottom, there is a 'Description' field with the text: 'Individual responsible for supply chain application administration. Collaborates with supply chain application users to maintain consistent application setup, rules, and access.'

Provide Access to Configure AI Agents in Oracle Fusion Cloud Procurement

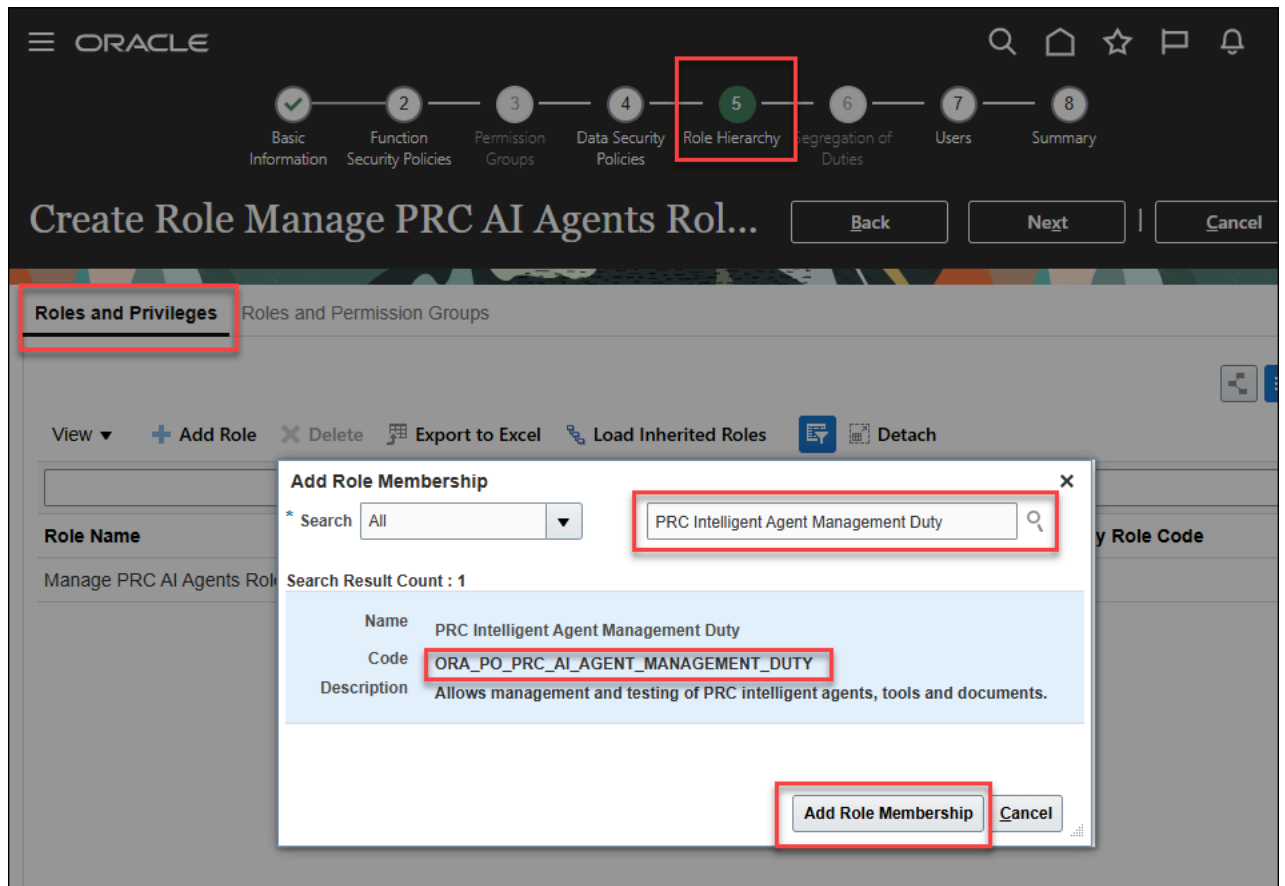
To give access to users without the Procurement Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.

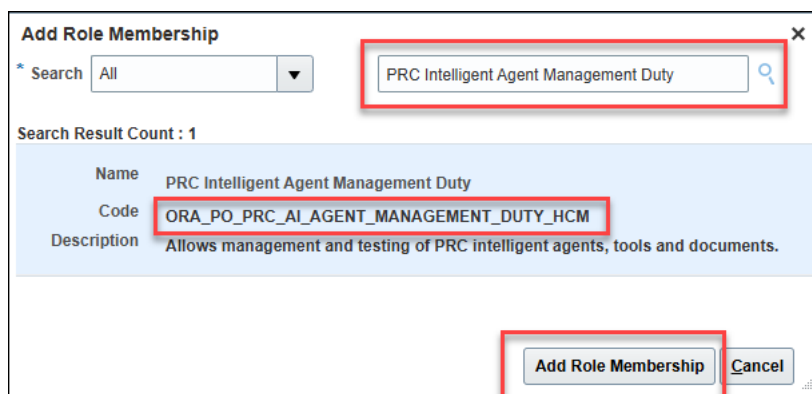
Note: Make sure to enable permission groups.

A screenshot of the 'Create Role' form in the Oracle Security Console. The form contains the following fields: '*Role Name' with the value 'Manage PRC AI Agents Role', '*Role Code' with the value 'MANAGE_PRC_AI_AGENTS_ROLE', and '*Role Category' with a dropdown menu showing 'Common - Job Roles'. Below these fields is a button labeled 'Enable Permission Groups' which is highlighted with a red box. Further down, there is a checkbox for 'Enable Role for Access from All IP Addresses' which is currently unchecked. At the bottom, there is a 'Description' field with the value 'Role to manage PRC AI Agents'.

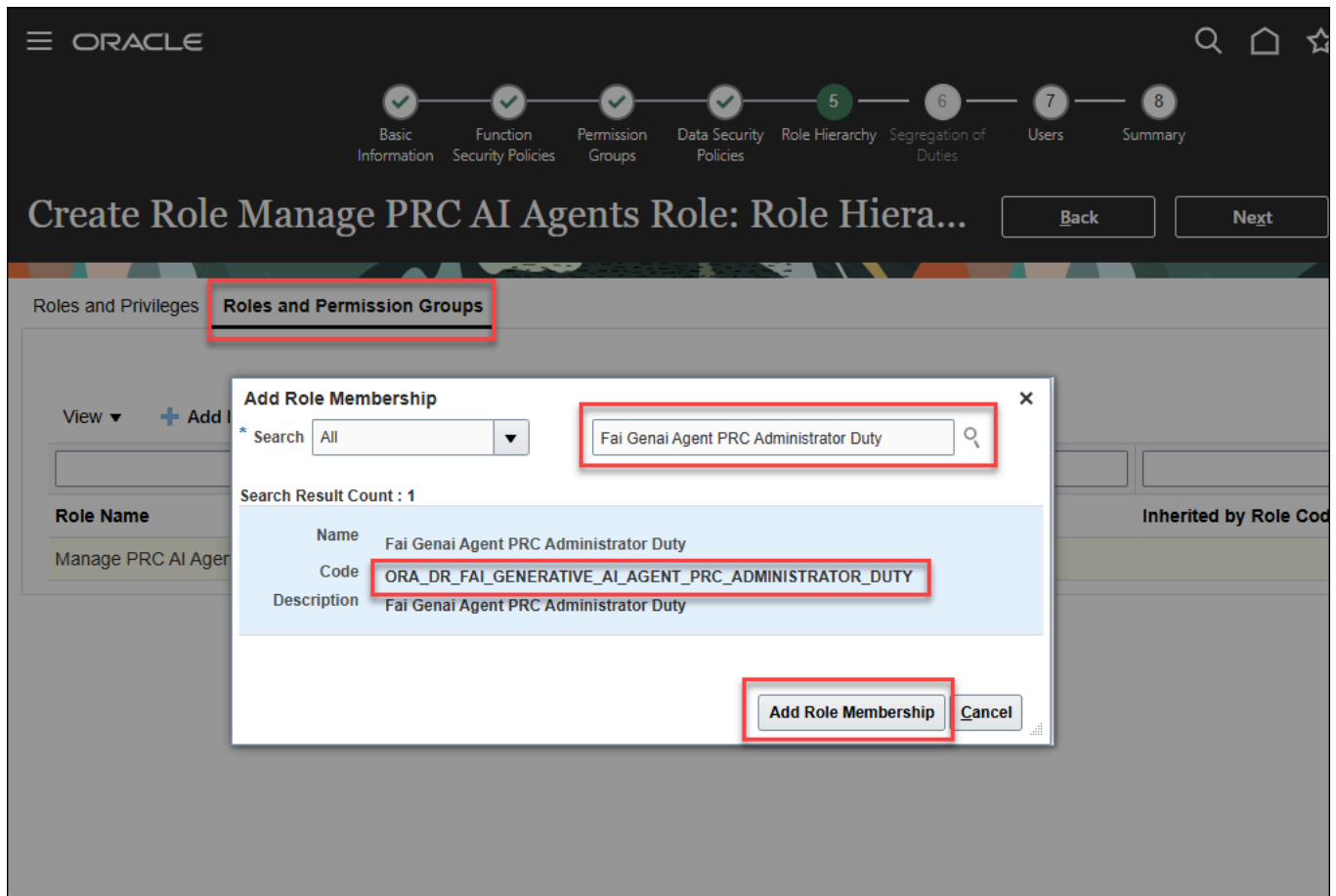
2. On the Role Hierarchy page, open the Roles and Privileges tab and add these roles:
 - o PRC Intelligent Agent Management Duty (ORA_PO_PRC_AI_AGENT_MANAGEMENT_DUTY)



- o PRC Intelligent Agent Management Duty (ORA_PO_PRC_AI_AGENT_MANAGEMENT_DUTY_HCM)



3. Open the Roles and Permission Groups tab and add the Fai Genai Agent PRC Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_PRC_ADMINISTRATOR_DUTY) duty role.

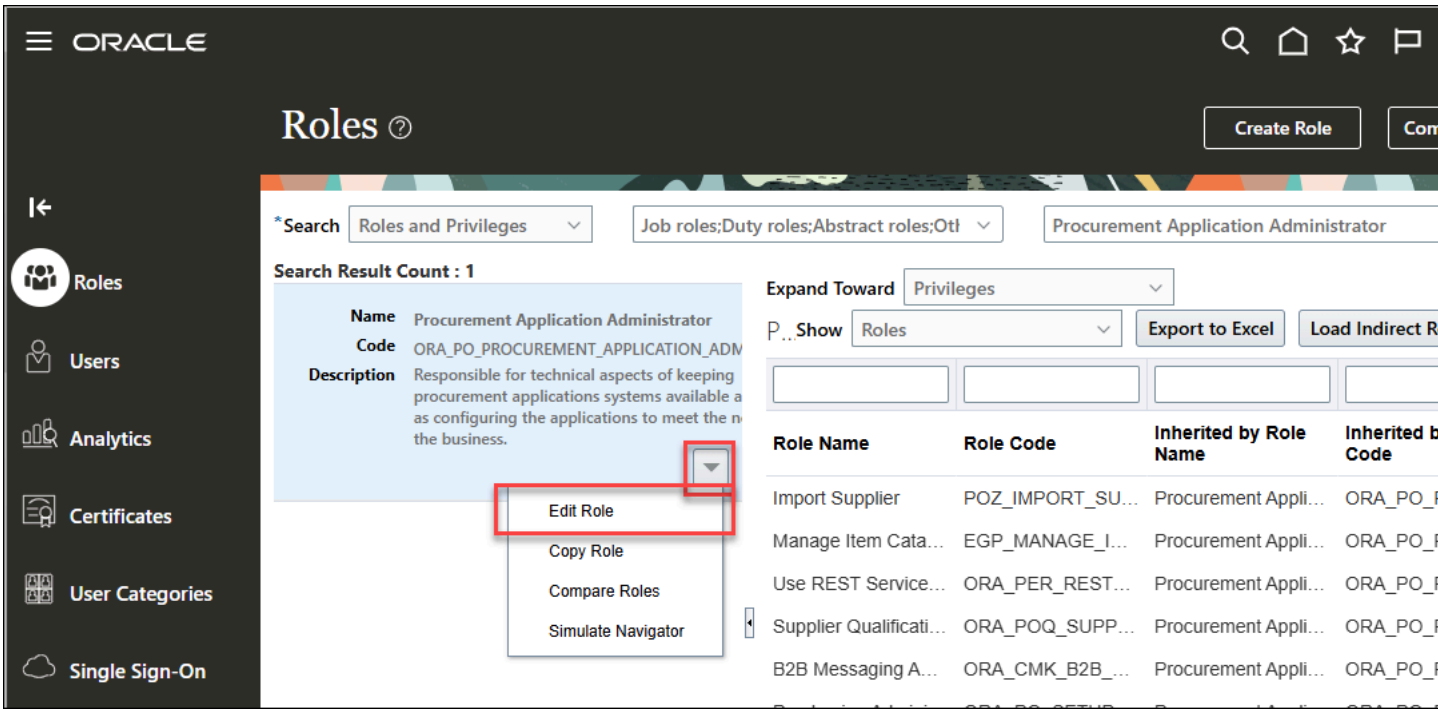
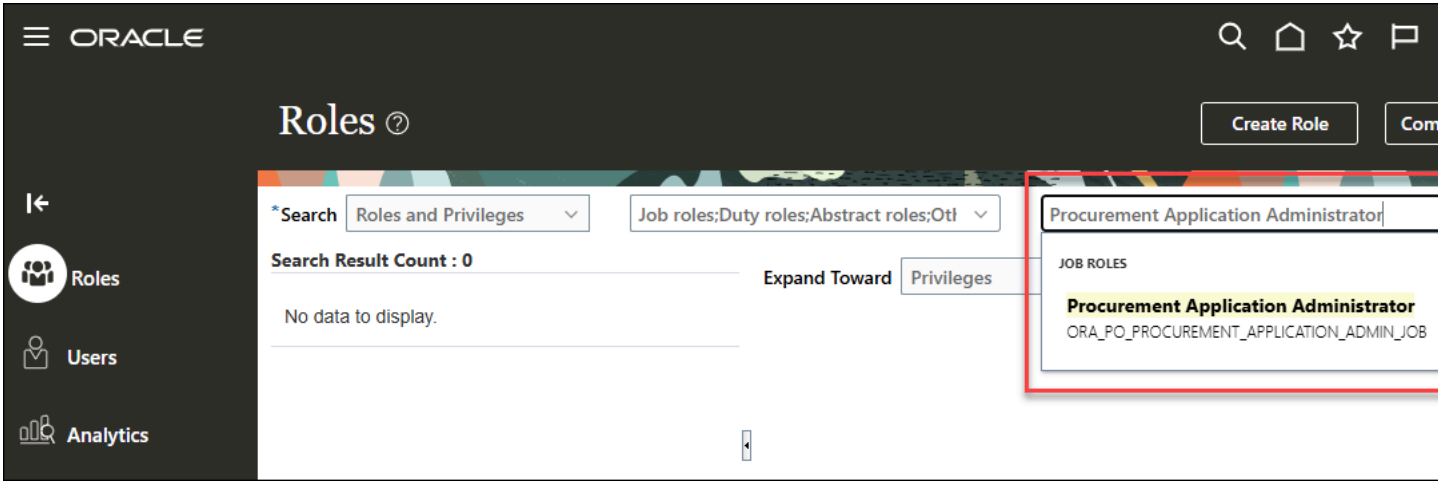


4. Save the custom role and assign this custom role to the appropriate job roles.

To give access to users with the Procurement Application Administrator Job Role:

1. Go to the Security Console.

2. Search for the Procurement Application Administrator (ORA_PO_PROCUREMENT_APPLICATION_ADMIN_JOB) job role, and edit it.



3. Enable permission groups and save the job role.

The screenshot shows the Oracle 'Edit Role' page for the 'Procurement Application Administrator' role. At the top, there is a navigation bar with five steps: 1. Basic Information, 2. Function Security Policies, 3. Permission Groups, 4. Data Security Policies, and 5. Role Hierarchy. The 'Edit Role' title is prominently displayed. Below the title, the role details are listed: Role Name (Procurement Application Administrator), Role Code (ORA_PO_PROCUREMENT_APPLICATION_ADMIN_JOB), and Role Category (Procurement - Job Roles). A red rectangular box highlights the 'Enable Permission Groups' button. Below this button, there are checkboxes for 'Predefined Role' (checked) and 'Enable Role for' (unchecked). The 'Access from All IP Addresses' section is also visible. The 'Description' field states: 'Responsible for technical aspects of keeping procurement applications systems available as well as configuring the applications to meet the needs of the business.'

Provide Access to Configure AI Agents in Oracle Fusion Cloud Financials

To give access to users without the Financial Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.
 - Note:** Make sure to enable permission groups.
2. Go to the Role Hierarchy page.
 - o Open the Roles and Privileges tab, and add these duty roles:
 - Manage Financials Intelligent Agent (ORA_FUN_MANAGE_FIN_AI_AGENT_HCM)
 - Manage Financials Intelligent Agent (ORA_FUN_MANAGE_FIN_AI_AGENT)
 - AI Agent Connector Management(ORA_CSO_CONNECTORS_AI_AGENT_MANAGEMENT_DUTY)
 - o Open the Roles and Permission Groups tab, and add the Fai Genai Agent FIN Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_FIN_ADMINISTRATOR_DUTY) duty role.
3. Save the custom role and assign it to the appropriate job roles.

To give access to users with the Financial Application Administrator Job Role:

1. Go to the Security Console.
2. Search for the Financial Application Administrator (ORA_FUN_FINANCIAL_APPLICATION_ADMINISTRATOR_JOB) job role, and edit it.
3. Enable permission groups and save the job role.

Provide Access to Configure AI Agents in Oracle Fusion Cloud CX

To give access to users without the Sales Administrator Job Role:

1. Go to the Security Console and create a new custom job role.
Note: Make sure to enable permission groups.
2. On the Role Hierarchy page, open the Roles and Privileges tab and add these roles:
 - Manage CX AI Agents (ORA_ZCA_MANAGE_CX_AI_AGENTS)
 - Manage CX AI Agents (ORA_ZCA_MANAGE_CX_AI_AGENTS_HCM)
3. Open the Roles and Permission Groups tab and add the Fai Genai Agent CX Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_CX_ADMINISTRATOR_DUTY) duty role.
4. Save the custom role and assign this custom role to the appropriate job roles.

To give access to users with the Sales Administrator Job Role:

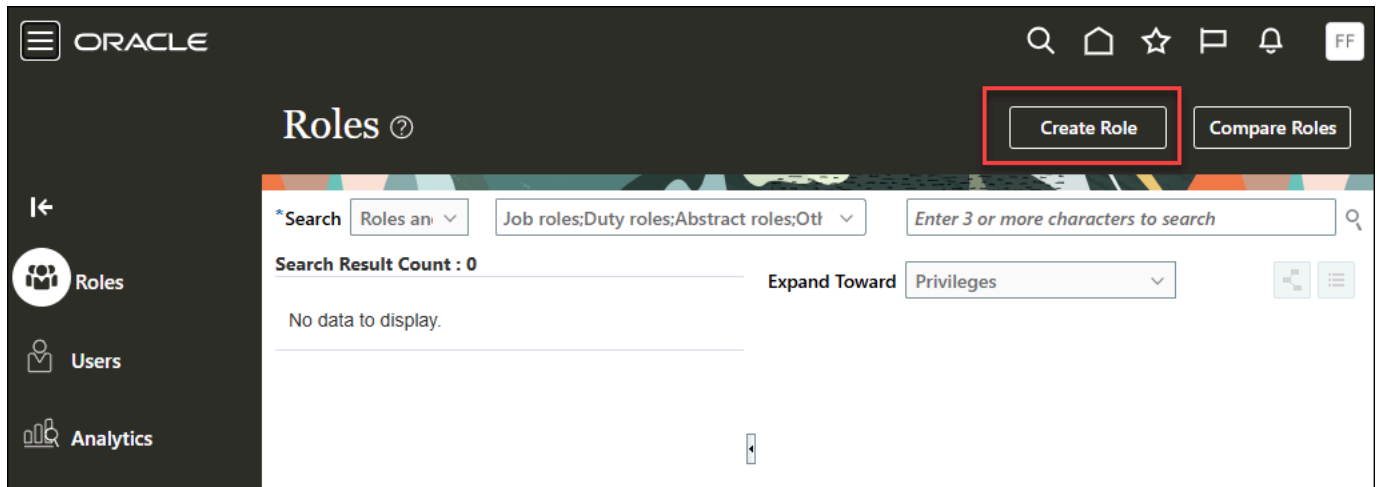
1. Go to the Security Console and search for the Sales Administrator (ORA_ZBS_SALES_ADMINISTRATOR_JOB) job role, and edit it.
2. Enable permission groups and save the job role.

Provide Access to Configure AI Agents in Oracle Permitting and Licensing

To give access to users without the PSC Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.



*Role Name

*Role Code

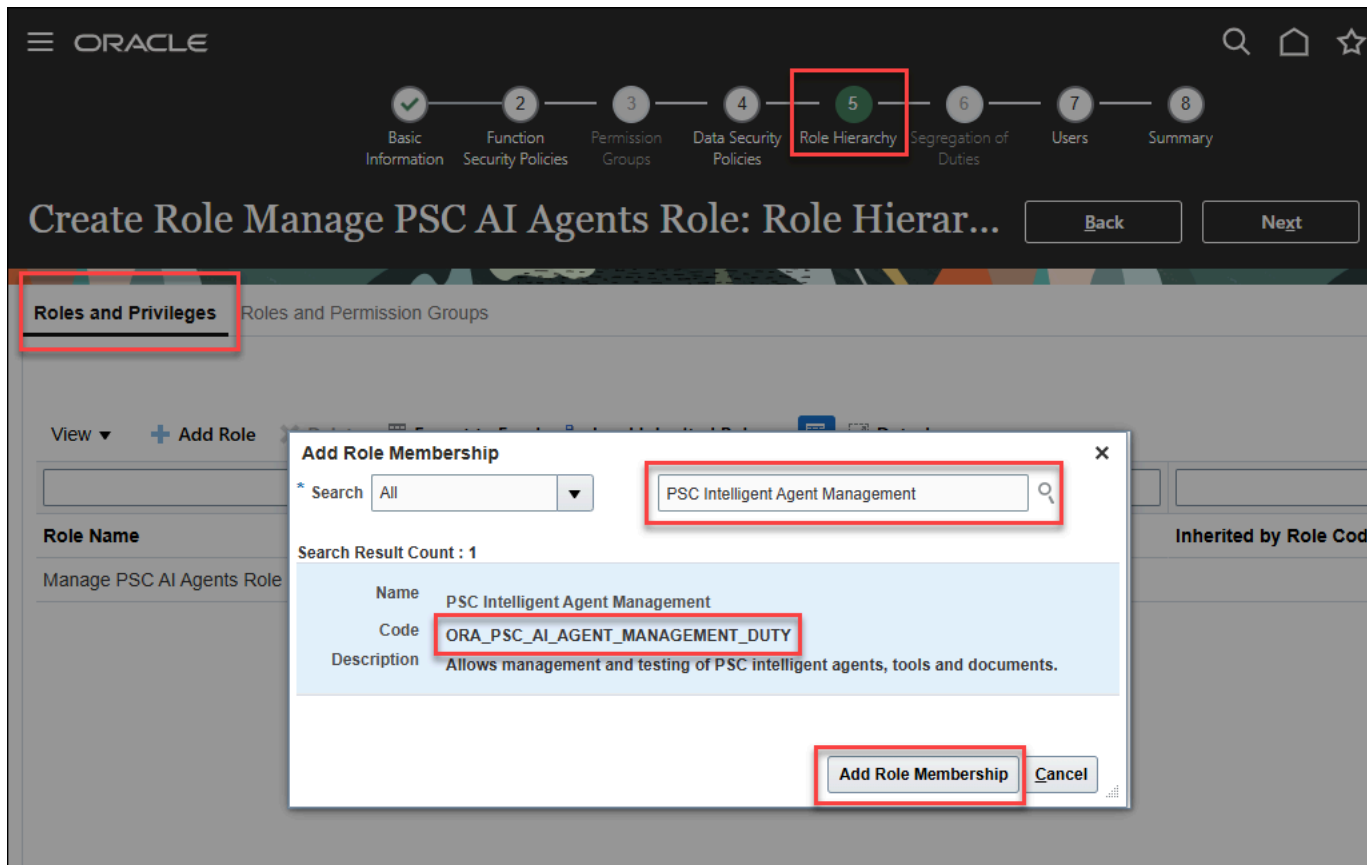
*Role Category

Enable Role for Access ☐

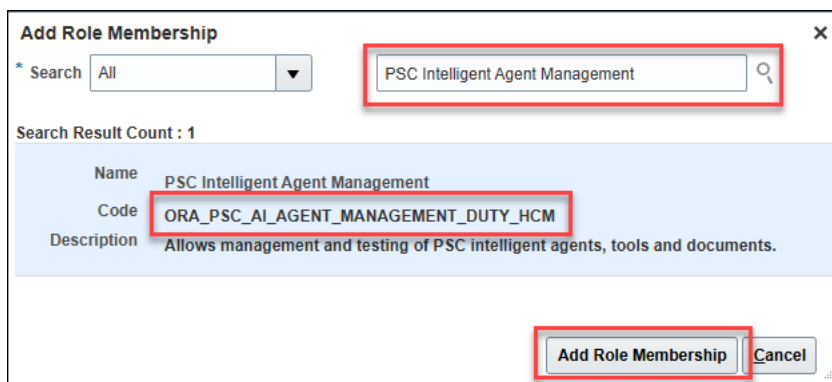
from All IP Addresses

Description

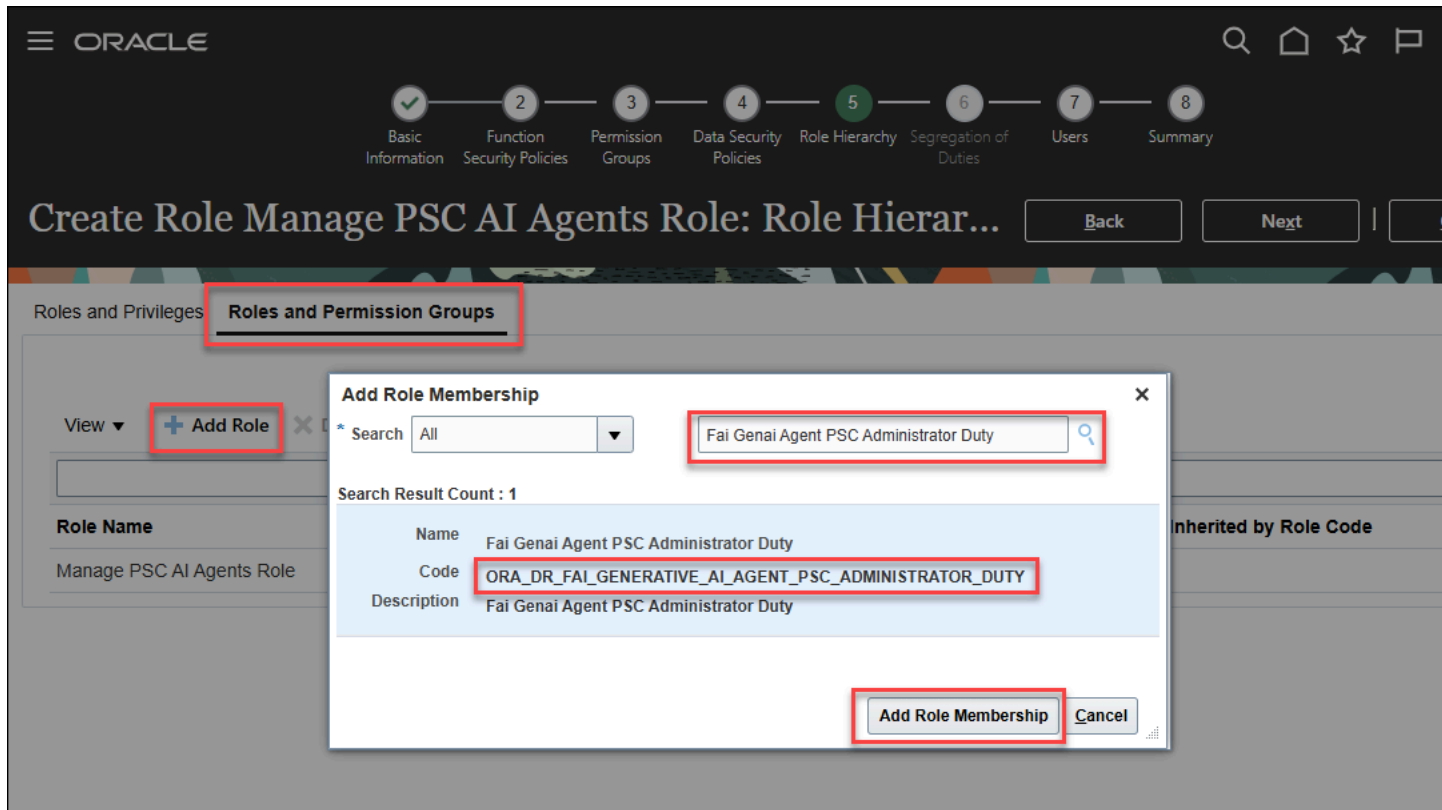
2. On the Role Hierarchy page, open the Roles and Privileges tab and add these roles:
 - o PSC Intelligent Agent Management (ORA_PSC_AI_AGENT_MANAGEMENT_DUTY)



- o PSC Intelligent Agent Management (ORA_PSC_AI_AGENT_MANAGEMENT_DUTY_HCM)



3. Open the Roles and Permission Groups tab and add the Fai Genai Agent PSC Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_PSC_ADMINISTRATOR_DUTY) duty role.

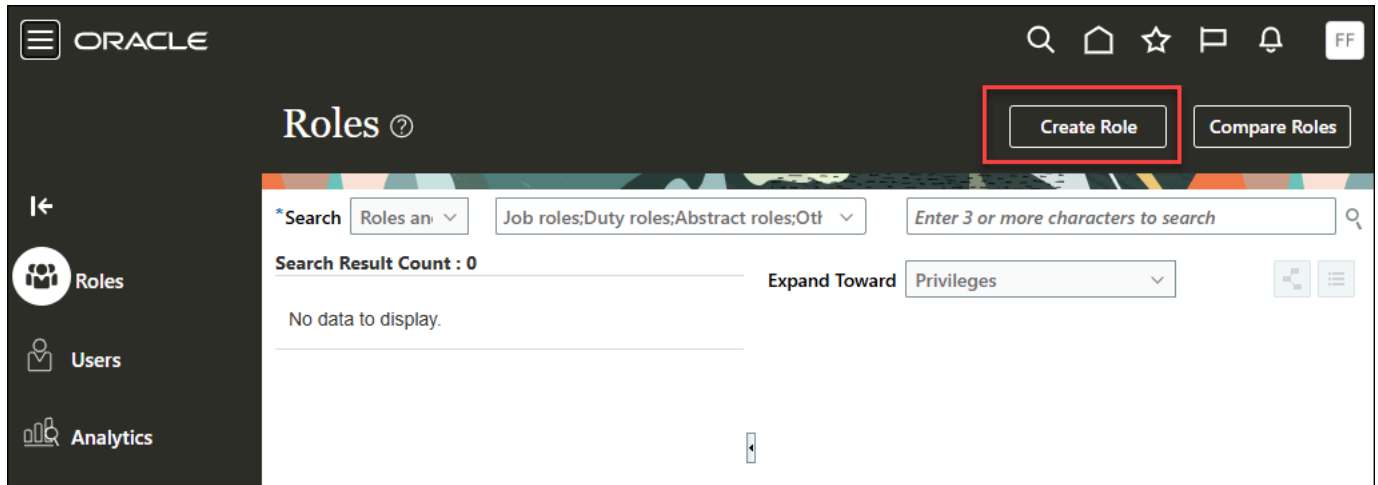


4. Save the custom role and assign this custom role to the appropriate job roles.

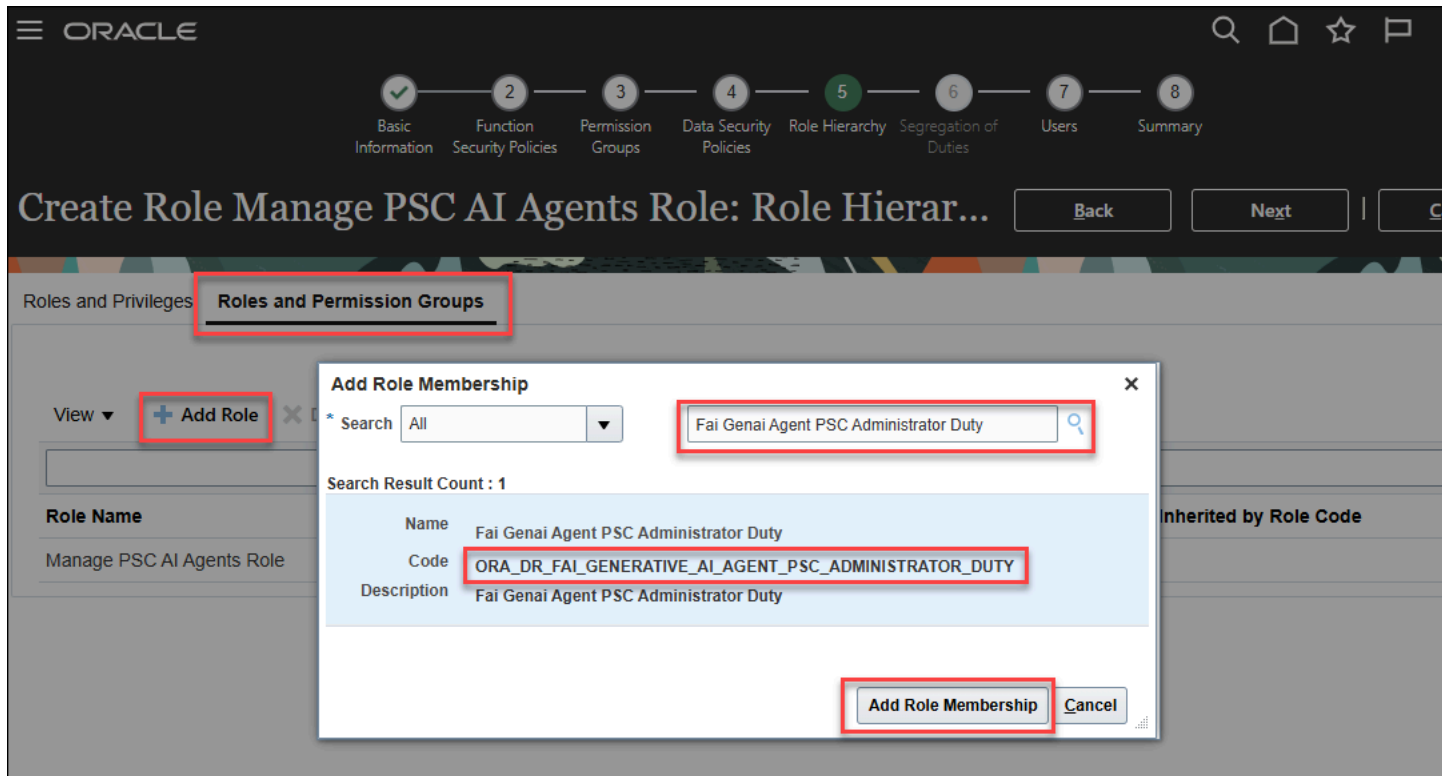
To give access to users with the PSC Application Administrator Job Role:

1. Go to the Security Console and create a new custom job role.

Note: Make sure to enable permission groups.

A screenshot of the 'Create Role' form. It contains several input fields: '*Role Name' with the value 'Manage PSC AI Agents Role', '*Role Code' with the value 'MANAGE_PSC_AI_AGENTS_ROLE', and '*Role Category' with a dropdown menu set to 'Common - Job Roles'. A button labeled 'Enable Permission Groups' is highlighted with a red box. Below these fields, there is a checkbox for 'Enable Role for Access from All IP Addresses' and a 'Description' text area.

- On the Role Hierarchy page, open the Roles and Permission Groups tab and add the Fai Genai Agent PSC Administrator Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_PSC_ADMINISTRATOR_DUTY) duty role.

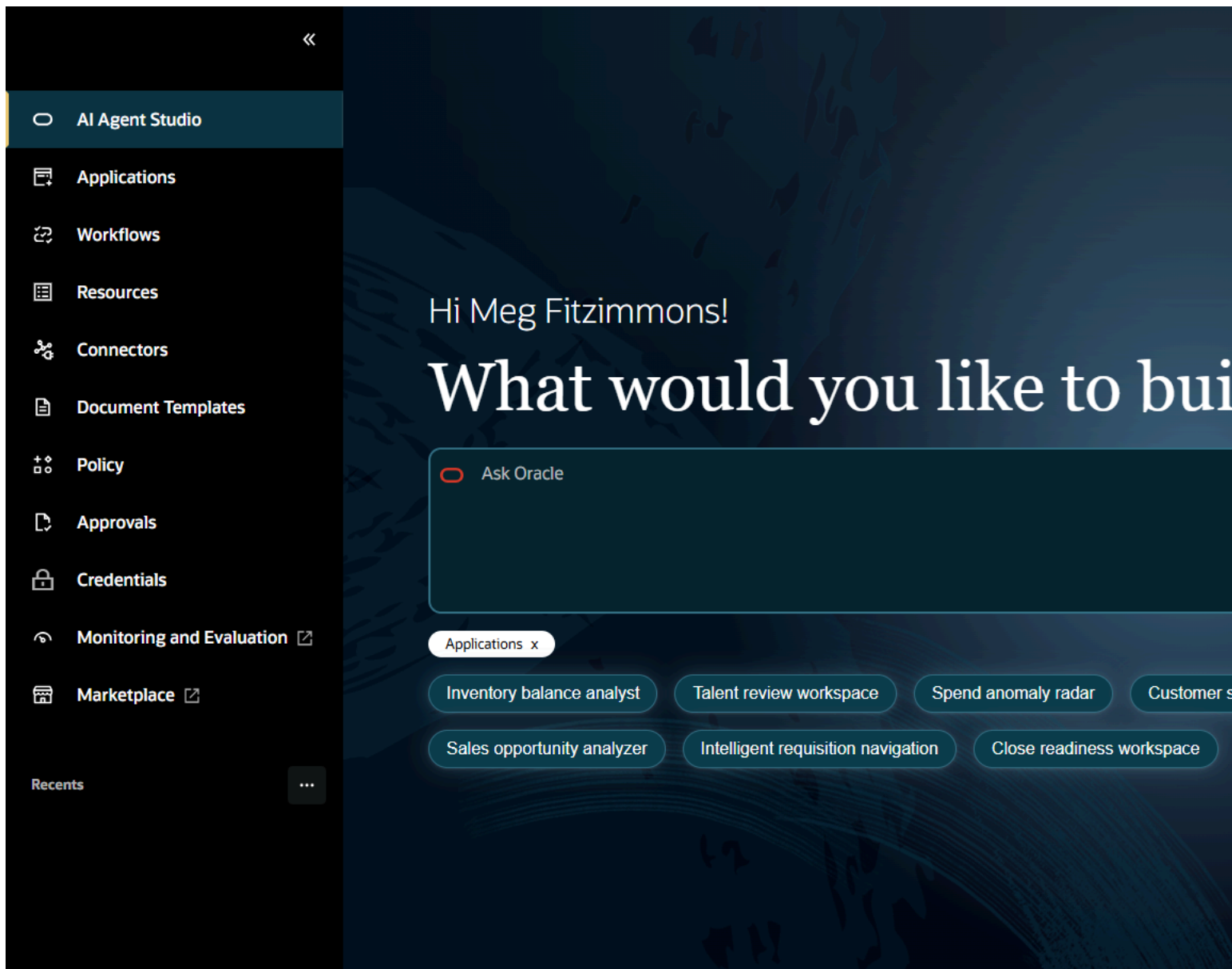


- Save the custom role and assign this custom role to the appropriate job roles.

3 Before You Begin

Choose How to Create Workflows and Agentic Applications

Using AI Agent Studio, you can create workflows and agentic apps from scratch, or modify preconfigured and existing ones.



Create Workflows

You can create workflows in either of these ways:

- **Using Ask Oracle**
 - a. In the **Ask Oracle** field, describe the workflow you want to create.

Example: **I want to create a Supplier Quote Assistant workflow to process quotes from multiple suppliers. The workflow should extract and map the required attributes from each quote document, create a draft requisition, and submit it for approval. A workflow agent is required to execute these tasks in the specified order.**
 - b. Close the **Applications** tag displayed below the **Ask Oracle** field, and then select the **Workflows** tag.
 - c. Select .
- **From the Workflows Tab**
 - a. From AI Agent Studio, select the **Workflows**  tab.
 - b. Select **Add**.
 - c. Specify all details for the workflow and publish it. For more information, see [Create Workflows](#).

Create Agentic Apps

You can create agentic apps in either of these ways:

- **Using Ask Oracle**
 - a. In the **Ask Oracle** field, describe the agentic app you want to create.

Example: **I want to create a Contract Assistant Agentic Application to generate a new contract with auto-filled fields. This assistant will be provided with multiple contract templates to generate the new contract. The app must also review existing contracts to perform risk analysis, recommend changes and route the contract for final approvals. For these activities that need distinct expertise (template management, legal validation, recommendation, and routing), the app will need to a workflow that collaborates with other worker agents.**
 - b. Make sure the **Applications** tag below the **Ask Oracle** field is selected.
 - c. Select .

The page to create a new agentic app is displayed where you can continue creating the agentic app with the help of the displayed instructions.
- **Using the Applications tab**
 - a. From AI Agent Studio, select the **Applications**  tab.
 - b. Select **Add**.
 - c. Specify all details for the agentic app and publish it. For more information, see [Build Agentic Applications Using App Builder](#).

Modify Existing Workflows and Agentic Apps

You can edit unpublished (draft) agentic apps and workflows.

- To modify agentic apps, go to the **Applications**  tab and select **Edit**  .

Preconfigured agentic applications are identified by the **Seeded** tag. Select **Duplicate**  and create a copy of the app to edit.
- To modify workflows, go to the **Workflows**  tab and select **Edit**  .

Preconfigured workflows are identified by the **Seeded** tag. Select **Duplicate**  and create a copy of the workflow to edit.

For more information, see [Edit a Workflow](#).

Get Started with AI Agent Studio

Use AI Agent Studio to create workflows consisting of one or more agents. Each agent can include multiple tools and topics, and workflows can also incorporate additional nodes available in the workflow builder. You can also create agentic apps that contain multiple specialized workflows. You can start with a preconfigured workflow or create a new workflow from scratch.

Here's a broad outline of the tasks involved in creating a workflow.

Task	Details
Define resources	Before creating agents, workflows, and agentic apps, you can create the resources that are to be used within them. Resources can include topics and tools required by the agent or workflow to answer questions, to connect to external sources, to send and receive emails, and so on. You can also specify the credentials to connect to external resources.
Build new agents	Define the capabilities and scope of the agent. - Agent name - Product area the agent will work in - Natural language instructions to allow the agentic flow, or other agents, to understand the capabilities of this particular agent - Tools and topics the agent will need to use. In addition to the tools and topics you created, you can also add the predefined ones.
Add a user (human) in the loop	If required, add an approval step for some actions that your AI agent will perform. The review step can be added at any point in the process for oversight and control over key actions, such as sending an email or updating a record.
Build a workflow	Create a workflow and add the necessary artifacts to it, including agents, LLMs, and other nodes.

Task	Details
Test the workflows	Make sure to test the workflows before deploying to production. Ask a test query, and determine the accuracy and relevance of the agent's response. You can also see the instructions the agent is following, and the actions the agent has taken to arrive at the response.
Deploy the workflow	After defining and testing your workflow, you can deploy it directly from AI Agent Studio. • Embed the agent conversation chat experience into any website or application. • Trigger the agent from an external resource using Webhooks, or seamlessly embed the chat experience into HTML and React web pages.

Here are some of the resources and artifacts you can create before creating your agents and workflows:

Tools:

- **Oracle Fusion Cloud Applications business object:** Business Object tools allow AI agents to retrieve, update, create, or delete business object records within Fusion Applications. Using this tool, agents can securely access information or call specific functions in the application. You can control what data the agents can access, by selecting the business objects and fields to use or ignore. AI agents adhere to the native security and role-based access controls of Fusion Applications, ensuring protection and privacy for your enterprise data.
- **Connector:** You can use this tool to connect your AI agents to various data sources such as SharePoint, web pages, and other custom content source.
- **Document tool:** You can upload specific documents to be used by the AI agent, and provide natural language instructions on how the agent should use these documents. The agent can then search for information in the documents to provide a more exact answer to a user's question.
- **Email:** This tool can access the email client to send emails that include summaries of interactions or details pulled from a knowledge store.
- **Deep link:** A deep link will send a user directly to the part of Fusion Applications where they can update underlying information. For example, if the user moved and wants to update their home address in the HR system, a deep link can quickly route the user to the page for making that update.
- **External REST:** You can connect to internal and external SaaS applications or public APIs using External REST tool. To connect to internal and third-party services, add the authorization parameters. For example, you can retrieve the real-time weather information for a specific location by creating an External REST tool that connects to the appropriate external API.
- **MCP:** You can use the Model Context Protocol (MCP) tool and securely connect to external MCP servers, without building additional REST wrappers or plugin logic.

Topics

Create topics with precise instructions that increase the effectiveness of your agents.

Connectors

Create connectors to connect your workflows to various data sources such as Microsoft Teams, Slack, SharePoint, web pages, and other custom content source.

To view the Microsoft Teams and Slack channels created before release 26C, go to AI Agent Studio, open the Credentials tab and then the Channels subtab. To use these channels in release 26C, create them from the Connectors tab.

For more information, see *How do I use connectors in AI Agent Studio?*

Add External REST Tool

You can enable your agents to securely connect to internal or external SaaS applications and public APIs by adding External REST tools in AI Agent Studio. Make sure you have a role with the Create and Edit Backends for Visual Builder Studio (ORA_FND_TRAP_PRIV) privilege assigned to it.

1. Go to **AI Agent Studio**.
2. Select the **Resources**  tab and then select the **Tools** subtab.
3. Select **Add**.
4. Add a new tool of type **External REST**.
5. Enter the name, code, and description for the tool, and select the appropriate family and product.
6. Select **Require Human Approval** to ensure that a person reviews and approves any action before the tool runs.
 - a. Message: Enter instructions for the reviewer.
 - b. Actions: Select the required action.
7. In the External REST section, add connection details for the third-party service.
 - a. Instance URL: Enter the base domain of the external API, with no paths, parameters, or trailing slashes. For example, <https://api.weather.gov> is correct.
 - b. Authentication: Define how your agents will authenticate with the external system.
 - c. Description: Provide details about the tool, including what the tool does, and when the agent must call the tool.
8. Select **Add Endpoint** to provide the endpoint details.
 - a. Name: Assign a unique name to this endpoint. For example, **getWeatherByCoordinates**.
 - b. Operation Type: Select the HTTP method required to process the REST call against the target endpoint.
 - c. Resource Path: Specify the relative resource path to be appended to the Instance URL. Don't include the protocol or base domain. For example, **/points/{latitude},{longitude}** is correct.
 - d. Description: Provide a clear summary of the endpoint's function and the specific task it performs within the workflow.
9. In the Parameters subtab, select **Add Parameter**, and define each dynamic variable in the path, including data type and purpose. For example, you can define latitude as the first parameter and longitude as the second.
10. In the Sample Queries subtab, select **Add Sample Query**, and list example user queries that should trigger this tool.
11. In the Headers subtab, select **Add Header**, and specify any required metadata and security credentials for the external server.

The tool now appears in the Tools subtab, part of the Resources tab. You can select it when creating agents, nodes, or workflows.

Add MCP Tool

Using Model Context Protocol (MCP) tool, you can securely connect to external MCP servers, and use their capabilities inside agents and nodes, without building additional REST wrappers or plugin logic.

1. Go to **AI Agent Studio**.
2. Select the **Resources**  tab and open the **Tools** subtab.
3. Add a new tool of type **MCP**.
4. Enter the name, code, and description for the tool, and select the appropriate family and product.
5. Add MCP server connection details.
 - Instance URL: Enter the endpoint URL of the MCP server your agent will connect to.
 - Transport Type: Select the communication method:
 - Server Sent Events (SSE): Enables real-time, one-way streaming of data from a server to a client.
 - StreamableHTTP: Enables servers to independently handle multiple client connections using the HTTP POST and GET requests.
 - Credential Type: Select the authentication method required by the MCP server.
6. Update the details and create the tool.

Your tool now appears in the Tools tab, and you can select it when creating agents, nodes, or workflows.

Add Your LLM

Oracle provides a set of LLMs that you can use when creating an agent, node, or workflow. You can also use other LLMs by adding their credentials in AI Agent Studio.

If the model you want to use isn't listed for a supported provider, you can submit a request through [My Oracle Cloud Support](#) to have the model added.

Note: Oracle updates the list of Bring Your Own LLM options on a monthly basis and may remove models that have been deprecated, based on the information provided by the respective model providers. You're responsible for obtaining and managing all the required third-party provider credentials, associated service costs, and custom model configurations. You must also monitor model lifecycle changes, keep configurations up-to-date, and complete any required migrations to supported model versions within the timelines specified by the model provider.

You can add an LLM from any of these supported providers:

- OpenAI (through Microsoft Azure)
- Gemini Enterprise Agent Platform (through Google Vertex AI)
- Google Gemini (through direct API)
- Anthropic (through Google Vertex AI)
- Oracle Cloud Infrastructure
- OpenAI (through direct API)

To add the credentials for available LLMs, do these steps:

1. Go to AI Agent Studio and open the **Credentials**  tab.
2. In the LLM subtab, select **Add LLM**.
3. Add the details and save it.

- **Provider:** Select the provider you need from the available options.
- **Model:** Choose the model you need from the available options.
- **Name:** Enter a name.
- **Code:** Enter a code.
- **Description:** Describe the configuration details.
- **Basic URL:** Enter the endpoint URL that will receive requests for the LLM.
- **API Key:** Add the API key for the selected model.

Your LLM now appears in the LLM tab, and you can select it when creating agents, nodes, or workflows.

To request a new LLM model, do these steps:

1. Create a service request through *My Oracle Cloud Support*.
2. Add the details for the approver.
 - Supported provider name shown in AI Agent Studio
 - Requested model and version
 - Business use case
 - Required timeline
3. Submit your request.

The time required to make a requested model available in AI Agent Studio depends on the complexity of the model integration. In most cases, requests are completed within two to four weeks. Each request is reviewed, and an estimated availability timeline is provided as part of the service request process. The model will appear in the Models list, where you can add LLMs.

Add Email Tool

Using the Email tool, you can securely enable agents to send, receive, and manage emails within automated workflows, without building additional email integrations or custom communication logic.

1. Go to **AI Agent Studio** and open the **Resources**  tab.
2. From the Tools subtab, select **Add**.
3. From the **Tool Type** list, select **Email**.
4. Enter the tool name, description, and select the appropriate family and product. The tool code will be automatically populated based on the tool name.
5. (Optional) Select **Require Human Approval** to ensure that a person reviews and approves any action before the tool runs.
 - a. **Message:** Enter instructions for the reviewer.
 - b. **Actions:** Select the required action to be done by the reviewer.
6. In the **Email** section, select **Add delivery option** and configure email notification details.
7. From the **Delivery Option** list, select **Email Notification**.
8. Enter the recipient information and email subject.
9. Create the email template as needed. You can also include tokens in your template.
 - a. From the **Tokens** list, select a token.
 - b. Select **Insert Token** to add the token to the template.
10. Select the option to enable attachments to include attachments in the email.

11. Save and create the email tool.

Your email tool now appears in the Tools tab, and you can select it when you create workflows.

Add Document Tool

Using the Document tool, you can upload specific documents for your agents to use. You can also include natural language instructions on how the agent should use these documents. The agent can access the information in the uploaded documents to formulate answers for user queries.

1. Go to **AI Agent Studio** and open the **Resources**  tab.
2. From the Tools subtab, select **Add**.
3. From the **Tool Type** list, select **Document**.
4. Enter the tool name, description, and select the appropriate family and product. The tool code will be automatically populated based on the tool name.
5. Enable **Require Human Approval** to ensure that a person reviews and approves any action before the tool runs.
 - a. **Actions:** Select the required action to be done by the reviewer.
 - b. **Message:** Enter instructions for the reviewer.
6. From the Document section, select **Add Document**.
7. Enter the name and description for the document.
8. Add files to the tool. Drag and drop the files, or select **Browse files** to select the files.
9. Select Status as **Ready to Publish**.
10. Select **Create**.

Selecting the Ready to Publish status initiates the **Process Agent Documents** scheduled process. After the scheduled process completes, your document status is automatically changed to Published. You can now choose the document tool from the Available Tools section when creating or editing an agent.

For information about creating documents to be uploaded to the Document tool, see [Create Documents to Add to Document Tool](#).

For information about supported file formats, see [What file types, limits, and processing capabilities are supported in AI Agent Studio?](#)

Add Email Accounts

To use an email trigger or a human approver in your workflows, you must register an email account with AI Agent Studio.

1. Go to **AI Agent Studio** and open the **Credentials**  tab.
2. In the **Email Accounts** subtab, select **Add Email Account**.
3. Add either a **Microsoft** or **Google** email account.
4. Select **Inbound** or **Approver** to designate the purpose of the email account.
 - Inbound: Receives emails used to initiate a workflow

- Approver: Receives emails to either approve or reject requests

These email account designations are configured in the same way and they differ only in where they appear in the UI. You can't use the same email account for both the inbound and approver accounts.

5. Enter the account information.

Field	Description
Email Folder	Folder name in the email account that's used for monitoring requests
Polling Interval	Interval at which AI Agent Studio checks the folder in the email account

Microsoft and Google accounts require different authorization details to manage the account. You can obtain the required information from the vendor's cloud console.

Account	Authorization Fields
Microsoft	◦ Tenant ID ◦ Client ID ◦ Client Secret
Google	◦ Service Account Email ◦ Private Key

Create Documents to Add to Document Tool

You can upload files to your document tools for agents to query from. Each document is a logical collection of attachments (files). You can also upload multimodal files.

For information about the types of files supported, see [What file types, limits, and processing capabilities are supported in AI Agent Studio?](#).

The content and structure of a document are key aspects to fetch the correct answer for your query. Here are some best practices for attachments:

- Make sure that a single document doesn't contain more than five images or scanned pages. This applies to the following types of files:
 - Fully scanned PDFs
 - Image-only documents
 - Multimodal files combining text and images.

If your content includes more than five images, split it into multiple smaller documents before uploading.

- Avoid heavy files. To ensure reliable processing and avoid latency or timeouts, make sure that your attachments follow these standards:
 - Avoid documents larger than 25 MB

- Use image compression where possible
- Prefer optimized PDFs over high-resolution raw scans
- Remove unnecessary pages, blank pages, or duplicated content before uploading

Document Structure

It's highly recommended to structure your document using headings, subheadings, and so on.

Headings must clearly describe the meaning of the content in the subsequent section. Each section (under a heading) must contain semantically related concepts. If you find that a section contains a mix of concepts, it's advisable to create sections. Also, the various heading levels must be clearly distinguishable. For example, use font size 12 for Heading 1, font size 10 for Heading 2, font size 8 for Heading 3, and so on.

A well-constructed document structure enables the agent to split the document into semantically meaningful chunks. This improves the quality of the generated answers.

Here's a comparison of two approaches to structuring a document:

Approach 1	Approach 2
Scope of Plan Coverage This plan covers Employees of ACME. An "Employee" shall mean a common law Employee of ACME, Inc. (ACME), ACME International Corporation and ACME Software Technology GmbH. Independent contractors and/or "leased workers" engaged by a staff leasing company aren't Employees for purposes of the Plan and therefore aren't eligible to participate in the Plan. Full and part time employees are covered as follows – You might participate in one of the medical, dental, vision, life/accidental death and dismemberment (AD&D), long term disability (LTD), and health care and dependent care flexible spending accounts (FSA) benefits offered under the Plan when you meet one of the eligibility requirements listed below. • Regular Full-Time Employee on the ACME U.S. payroll scheduled to work 30 or more hours per week. • Regular Part-Time Employee on the ACME U.S. payroll scheduled to work 20-29 hours per week.	Scope of Plan Coverage This plan covers Employees of ACME and their dependents. Eligible Employees An "Employee" shall mean a common law Employee of ACME, Inc. (ACME), ACME International Corporation and ACME Software Technology GmbH. Independent contractors and/or "leased workers" engaged by a staff leasing company aren't Employees for purposes of the Plan and therefore aren't eligible to participate in the Plan. FULL & PART TIME EMPLOYEES You might participate in one of the medical, dental, vision, life/accidental death and dismemberment (AD&D), long term disability (LTD), and health care and dependent care flexible spending accounts (FSA) benefits offered under the Plan when you meet one of the eligibility requirements listed below. • Regular Full-Time Employee on the ACME U.S. payroll scheduled to work 30 or more hours per week. • Regular Part-Time Employee on the ACME U.S. payroll scheduled to work 20-29 hours per week. • Interns and temporary Employees aren't eligible to participate in the Plan as they're not regular Employees of ACME. INPATRIATES An Inpatriate (and eligible Dependents) residing in the United States for 90+ days and scheduled to work 30 -- 40 hours/week for an ACME Entity will be enrolled, upon notifying , in the following benefits: • ACME Premium PPO Medical • Dental Plan II

Approach 1	Approach 2
- Interns and temporary Employees aren't eligible to participate in the Plan as they're not regular Employees of ACME. INPATRIATES are covered as follows – An Inpatriate (and eligible Dependents) residing in the United States for 90+ days and scheduled to work 30 -- 40 hours/ week for an ACME Entity will be enrolled, upon notifying , in the following benefits: - ACME Premium PPO Medical - Dental Plan II - Vision Service Plan (VSP) Option 1 - An Inpatriate (and eligible Dependents) are not eligible for Life/Accidental Death and Dismemberment (AD&D) and Long-Term Disability (LTD) coverage. Your eligible dependents may be covered by the medical, dental, vision, and life insurance options under the Plan when Your Dependent meets one of the eligibility requirements listed in this section. NOTE: Your Dependents may not enroll in a particular benefit option offered under the Plan unless You are also enrolled	- Vision Service Plan (VSP) Option 1 - An Inpatriate (and eligible Dependents) are not eligible for Life/Accidental Death and Dismemberment (AD&D) and Long-Term Disability (LTD) coverage. ELIGIBLE DEPENDENTS Your eligible Dependents may be covered by the medical, dental, vision, and life insurance options under the Plan when Your Dependent meets one of the eligibility requirements listed in this section. NOTE: Your Dependents may not enroll in a particular benefit option offered under the Plan unless You are also enrolled.

Approach 2 is highly recommended. Breaking the document into semantically related content, organized by the various coverage categories, enables the agent to provide more relevant answers to questions about benefits coverage. Also, note that the various heading levels are visually distinguishable.

Add Connector Tool

You can use the Connector tool to connect your AI agents to various data sources such as SharePoint, web pages, and other custom content sources. For more information, see *How do I use connectors in AI Agent Studio?*

4 Use Cases

Single Agent with Multiple Workers

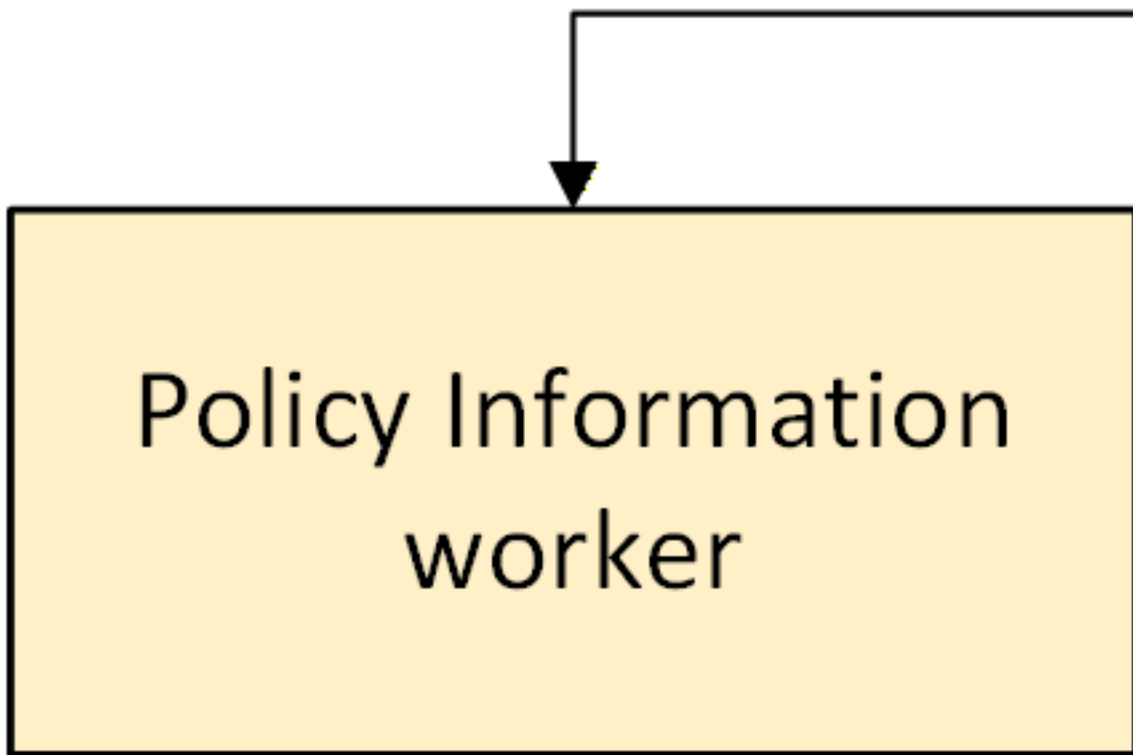
A worker is a specialized agent designed to perform a narrowly defined task or skill, and act as a modular building block for supervisor and workflow agents. Each worker is configured with specific instructions and has access to the tools required to perform that role effectively. Workers don't make high-level decisions or independently orchestrate tasks; instead, they perform well-scoped operations with precision.

Workers are designed to operate as part of a single or multi-agent system, and not as standalone entities. Workers are typically embedded within supervisor or workflow agents that coordinate their usage. This includes deciding when and how to initiate workers, pass appropriate context, and integrate the output from the worker into the overall task operation.

For example, a supervisor agent, based on the problem context, might dynamically select and initiate workers in a non-deterministic manner. A workflow agent might call workers in a predefined, deterministic sequence. In both cases, workers remain task-focused and don't control the broader flow of operations.

In this pattern, a Benefits Advisor acts as a central agent, helping employees understand and make the most of their benefits packages. The agent coordinates several specialized workers to provide a personalized support for medical, dental, and vision coverage.

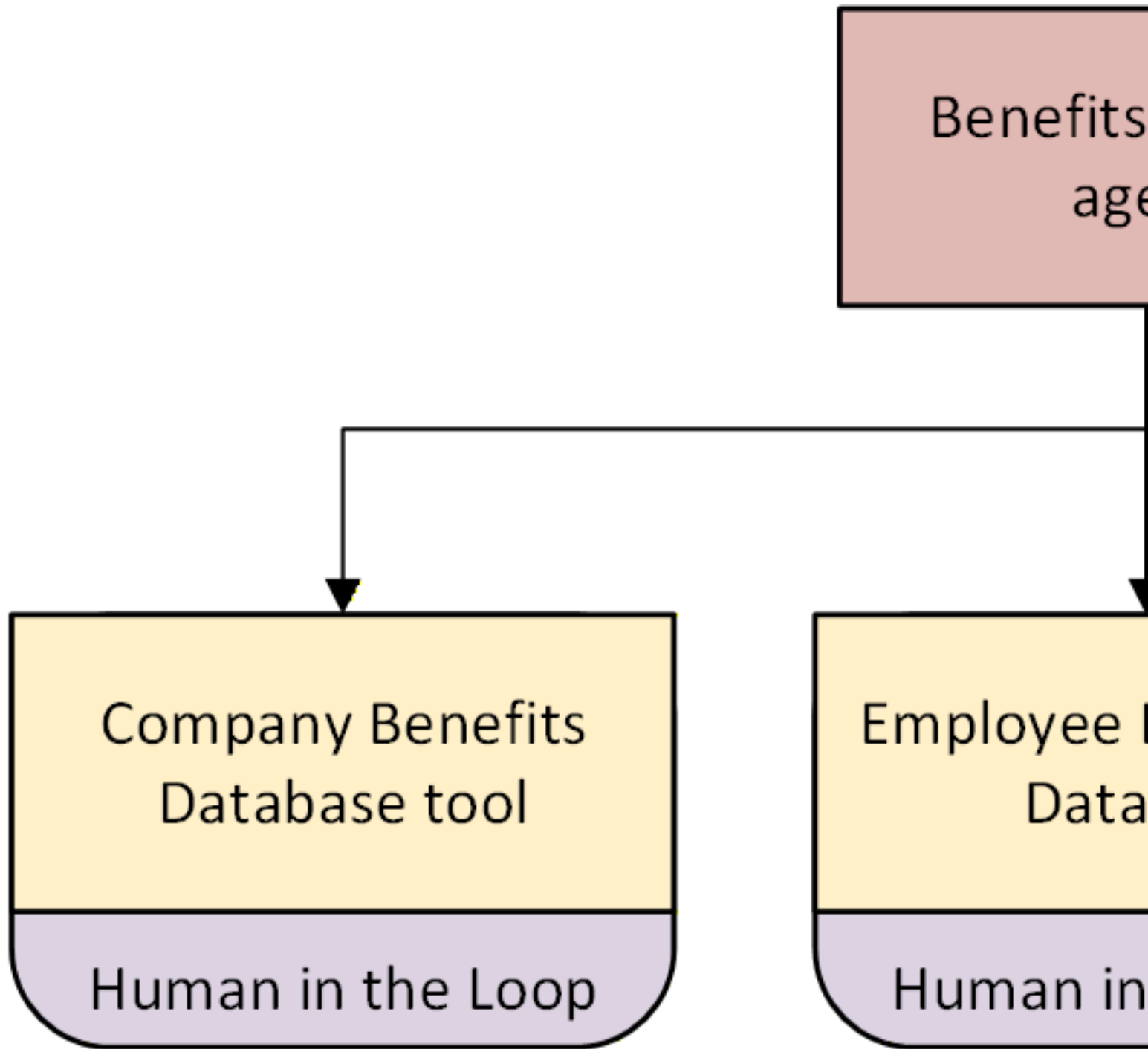
- The **Benefits Advisor** agent serves as the initial contact, helping the employee understand their benefits, and answers any preliminary questions.
- The **Policy Information** worker provides the employee with detailed information on available medical, dental, and vision benefits.
- The **Personal Enrollment** worker uses the employee's enrollment details to offer tailored advice on how they can optimize their benefits package.
- The **Eligibility and Assistance** worker answers queries regarding eligibility, coverage, helping to resolve issues related to benefits.



Single Agent with Multiple Tools and a Human in the Loop

This pattern shows the Benefits Advisor as a central agent that helps employees understand and optimize their benefits. This agent coordinates specialized workers to provide personalized insights on medical, dental, and vision coverage.

- The **Benefits Advisor** agent starts the interaction with a friendly introduction and asks for the employee's needs regarding benefits.
- The **Company Benefits Database** tool retrieves information on the company's benefits policies, eligibility, and plan details.
- The **Employee Enrollment Data** tool uses the employee's personal enrollment information to provide tailored advice, helping them understand and optimize their coverage.
- The **Benefits Optimizer** tool analyzes the employee's selections and offers personalized suggestions for improvement.
- The *Human-in-the-Loop* flag determines whether a user must provide explicit approval before the agent retrieves sensitive data, such as salary or personal information.



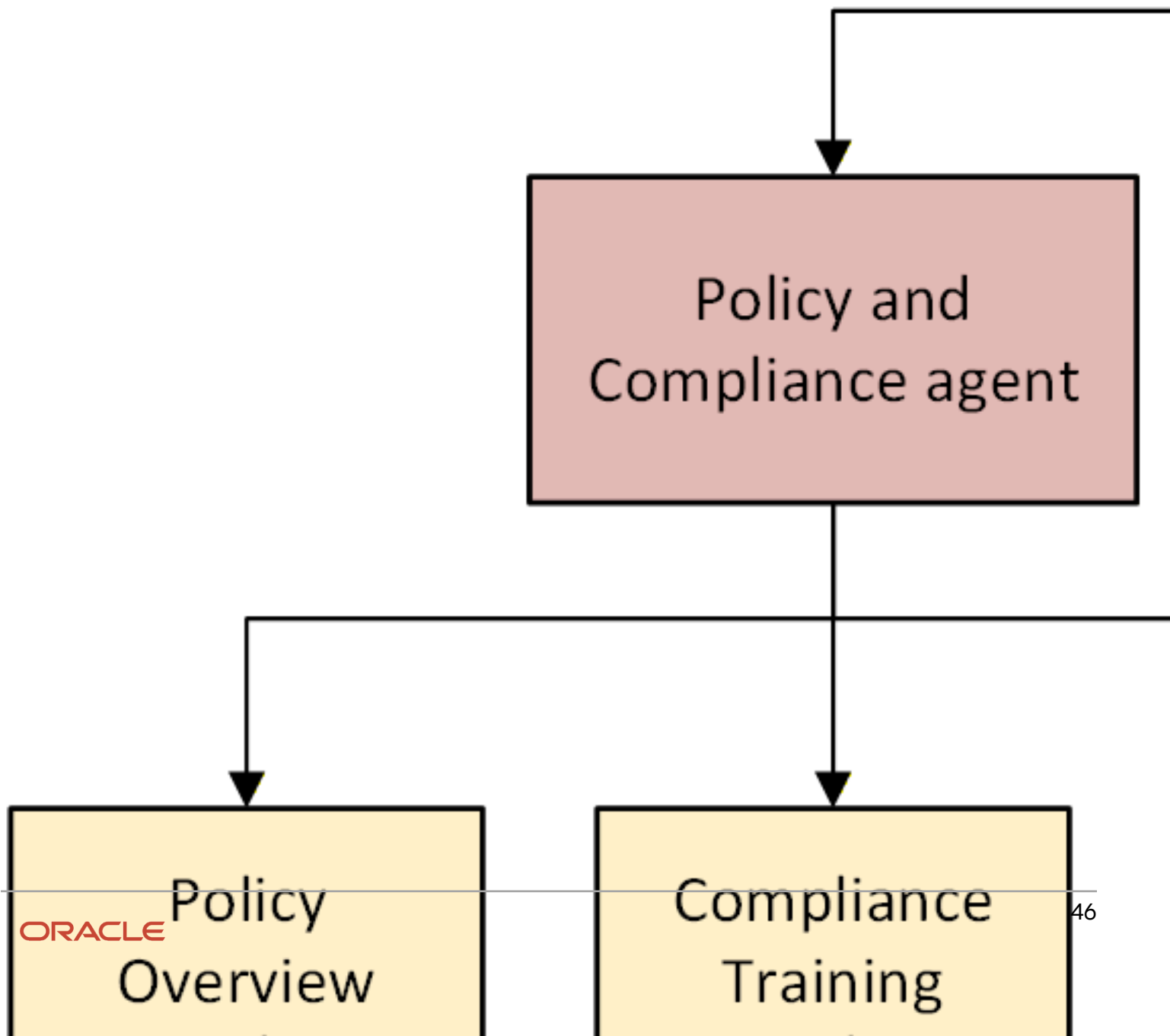
Multiple Agents With Multiple Tools and Delegations

In this pattern, the Central Onboarding Assistant agent acts as a supervisor, guiding new employees through their onboarding process. The supervisor agent delegates tasks to specialized agents, each focused on a specific aspect of onboarding, supported by dedicated workers.

Agent	Description
Policy and Compliance Agent	This agent helps employees understand and comply with company policies, and is supported by these workers. • Policy Overview worker: Shares key policies like leave, dress code, and workplace behavior. • Compliance Training worker: Provides access to mandatory training. For example, safety and social media policies. • Document Management worker: Ensures the employee completes and submits all required forms, such as tax forms and nondisclosure agreements.
Cultural Integration Agent	This agent helps employees adapt to the company's culture and connect with teams, and is supported by these workers. • Company Values worker: Explains the vision, mission, and goals of the organization. • Community Introduction worker: Connects employees with internal groups, events, and networking opportunities. • Workplace Norms worker: Provides guidance about unwritten rules, such as communication etiquette and meeting culture.
Tools and Resources Agent	This agent ensures that employees have access to tools, systems, and knowledge required for their job, and is supported by these workers. • Account Setup worker: Helps employees set up email, communication tools, and HR platforms. • Resource Guide worker: Provides access to handbooks, FAQs, and internal documentation. • Technical Support worker: Troubleshoots issues with tools or platforms.

Process

- The Central Onboarding Assistant agent acts as the entry point that gathers basic information about the employee's role and specific needs, and delegates tasks to specialized agents based on the query.
- The Policy and Compliance agent handles queries, or tasks, related to company rules and legal documentation, such as, "What's the process for requesting leave?"
- The Cultural Integration agent handles queries that focus on helping employees feel part of the team and aligned with the company values, such as, "How do I join company events?"
- The Tools and Resources agent handles queries about gaining access to systems and essential knowledge, such as, "How do I log into the time-tracking tool?"



Product-Specific Use Case Examples

Oracle Fusion Cloud CX

Agent Type	Key Characteristics	When to Use	Example
Customer service agent	• Interactive • Rule-based • Escalation-friendly • Customer-focused	• Repetitive, high-volume queries • Customer support • Query resolution	Chatbot for customer support agent answers FAQs and resolves issues.
Learning agent	• Adaptive • Feedback-driven • Continuously improves • Explainable	• Adaptive systems • Chatbot improvement • Demand forecasting refinement	Chatbot improvement agent learns from customer feedback to improve responses.

Oracle Fusion Cloud HCM

Agent Type	Key Characteristics	When to Use	Example
Copilots	• Collaborative • Proactive • Decision-making • Domain-specific	• Complex tasks requiring expertise • Draft documents • Input validation	Hiring copilot agent helps to build job descriptions and suggests interview questions.
Document referencing agent	• Knowledge-intensive • Accurate • Up-to-date • Query-driven	• Knowledge-intensive tasks • Answers to policy or FAQ questions • Compliance checks	Benefits agent answers employee questions about benefits.
Evaluating agent	• Assessment-focused • Criteria-based • Actionable insights • Feedback-driven	• Performance evaluation • Candidate ranking • Feedback analysis	Candidate evaluation agent ranks candidates based on interview performance.
Scheduling agent	• Task-oriented • Reactive • Limited decision-making • User-centric	• Routine task scheduling • Calendar management • Appointment booking	Interview scheduling agent coordinates availability between candidates and hiring managers.
Task automation agent	• Repetitive task handler • Efficient	• Repetitive, high-volume tasks • Payroll processing	Payroll processing agent automates time sheet

Agent Type	Key Characteristics	When to Use	Example
	- Error-resistant - Trigger-based	Form generation	collection and payslip generation.

Oracle Fusion Cloud SCM

Agent Type	Key Characteristics	When to Use	Example
Collaborative workflow agent	- Multi-stakeholder - Real-time coordination - Role-based task allocation - Communication-driven	- Complex processes requiring collaboration - Supplier collaboration - Crisis management	Supplier collaboration agent coordinates with procurement, supplier, and warehouse bots to work together.
Decision-making agent	- Data-driven - Recommendation-focused - Transparent - Adaptive	- Data-driven tasks - Supplier evaluation - Candidate shortlisting	Supplier evaluation agent evaluates suppliers based on cost and quality.
Goal-seeking agent	- Optimization-focused - Real-time adjustments - Objective-driven - Explainability	- Resource allocation - Route planning - Workforce scheduling	Route optimization agent plans optimal delivery routes.
Monitoring agent	- Proactive - Alert-driven - Data-intensive - Response-oriented	- Proactive management - Inventory monitoring - Social media tracking	Inventory monitoring agent tracks in-stock levels and alerts for replenishment.

5 Expressions in AI Agent Studio

Expressions

You can use expressions to capture, store, and reuse values instead of deriving them each time, across various steps in your agents and workflows, from AI Agent Studio. For example, you can use expressions to reuse extracted fields such as `personId`, intermediate node outputs, and so on. Expressions are different from static values. Static values can be anything you specify, whereas expressions are represented in the `{{any expression}}` format. You can directly enter expressions in the required format, or you can add it from the **Insert Expression**  menu that's displayed with the fields. Here are the types of values allowed in expressions:

- String
- Date
- Integer
- Number
- Boolean
- Object
- Array

Variables Specific to Workflows

Expressions contain different types of variables that provide the necessary context. They've a defined set of scopes in workflows.

Scope	Description
Global	Values that can be referenced anywhere in the workflow. For example, Current Date , Trigger Type , and variables defined at the workflow level.
Node	Values such as the node input variable's value and the output of each node in the workflow.
Trigger	Every workflow can have multiple triggers and each trigger will have its own set of inputs.

Here are the types of variables available in expressions that are specific to workflows:

- System Variables
- Workflow Variables
- User Variables
- Trigger Variables
- User-defined Variables

- Node Variables

System Variables

System Variables are predefined runtime read-only values provided by the application. These variables can be referenced in inputs, agent mappings, and prompts using the `{{context.$system...}}` syntax. For example, `{{context.$system.$currentDate}}`, `{{context.$system.$inputMessage}}`.

Scope	Expression	Description
Current Date	<code>{{context.\$system.\$currentDate}}</code>	Current system date (ISO format)
Current Date and Time	<code>{{context.\$system.\$currentDateTime}}</code>	Current system time stamp
Trigger Type	<code>{{context.\$system.\$triggerType}}</code>	Trigger that started the workflow
Input Message	<code>{{context.\$system.\$inputMessage}}</code>	Raw user chat input
User Name	<code>{{context.\$system.\$chatHistory}}</code>	Full conversation history for the current session (prior user and assistant turns)

Workflow Variables

Workflow variables are workflow-scoped runtime values that are created and managed within a workflow process.

Scope	Expression	Description
Workflow Name	<code>{{context.\$workflow.\$name}}</code>	The ID of the workflow being run. The variable is always available at runtime, and remains stable across runs unless the workflow is renamed.
Trace ID	<code>{{context.\$workflow.\$traceId}}</code>	Unique identifier for a specific workflow run instance that's generated per run. This variable is critical for debugging and support.
Conversation ID	<code>{{context.\$workflow.\$conversationId}}</code>	Identifier for the conversation or session associated with the workflow run.

User Variables

User variables are predefined runtime values that describe the user currently interacting with the application or chat. The user's details are available using the `{{context.$user...}}` syntax. For example, user name can be retrieved using the expression `{{context.$user.$name}}`.

Scope	Expression	Description
User Name	<code>{{context.\$user.\$name}}</code>	ID of the authenticated user who initiated the workflow execution

Trigger Variables

When a workflow is called by an email trigger or REST trigger, trigger variables are available for use.

- REST Trigger

Scope	Expression	Description
Triggers > REST > Input > field	{{context.\$triggers.REST.\$input.<field>}}	Specific request field

- Email Trigger

Scope	Expression	Description
Triggers > Input > subject	{{context.\$triggers.EMAIL.\$input.subject}}	Subject line of the incoming email that triggered the workflow
Triggers > Input > fromAddress	{{context.\$triggers.EMAIL.\$input.fromAddress}}	Email address of the sender
Triggers > Input > content	{{context.\$triggers.EMAIL.\$input.content}}	Body of the email
Triggers > Input > headers	{{context.\$triggers.EMAIL.\$input.headers}}	Full set of email headers
Triggers > Input > attachments	{{context.\$triggers.EMAIL.\$input.attachments}}	Attachments included in the email

User-defined Variables

In addition to the system provided variables, you can also add variables that can be shared across nodes in a workflow. You can define these variables when creating the workflow, and they're available throughout your workflow's flow. You can use the defined user variables in user questions or conversations.

To add user defined variables, do these steps:

1. When creating or editing a workflow, select the Variables tab.
2. Select **Add** and enter the details:
 - Name: Enter the name for the variable
 - Type: Choose the type, for example, string.
 - Scope: Choose the scope as **User Question** or **Conversation**.
3. Apply the changes and save.

You can now reference the variable using {{context.\$variables.<variableName>}}.

To assign a value for the variable you created, do these steps:

1. Edit the workflow in which you want to add the user-defined variable.

2. Add a node of type **Logic > Set Variables** and enter the details to create the variable node.
 - o From the Variables section, select **Add**.
 - o From the **Name** field, choose the user defined variable you had created.
 - o Enter the value for the variable. You can enter a static value, or a dynamic value that's derived from the runtime context.
3. Save your changes.

Scope	Expression	Description
Variables > <name>	{{ \$context.\$variables.<name> }}	Workflow-level variable

Node Variables

Node Inputs

Node inputs are variables used to pass input to the node in the workflow.

Scope	Expression	Description
Entire input	{{ \$context.\$nodes.<nodeCode>.\$input }}	Returns the complete input object passed to the specified node.
Specific input field	{{ \$context.\$nodes.<nodeCode>.\$input.<field> }}	Returns a single field from a node's input object.

Node Outputs

Node outputs are variables that contain the output of the node in the workflow.

Scope	Expression	Description
Entire output	{{ \$context.\$nodes.<nodeCode>.\$output }}	Returns the complete output object produced by the specified node (schema-based if defined).
Specific output field	{{ \$context.\$nodes.<nodeCode>.\$output.<field> }}	Returns a single field from a node's output object.

- o
 - Use entire input or output objects when flexibility is needed or when passing data to an LLM for reasoning.
 - Use specific fields when doing these actions:
 - o Writing conditions
 - o Assigning workflow variables
 - o Populating email subjects or concise prompts
 - o Referencing a field that doesn't exist at runtime resolves to empty or null

- **Node Error Details**

Node error details are variables that are available when a node in the workflow fails and the Error Handler is called. They're scoped to the node that encountered the error.

Scope	Expression	Description
Error information	<code>{{context.\$nodes.<nodeCode>.\$error.\$info}}</code>	Detailed error information returned by the failed node
Error code	<code>{{context.\$nodes.<nodeCode>.\$error.\$code}}</code>	Machine-readable error code associated with the node failure

- **Node Execution Status**

Node execution status variable contains the status of the node in the workflow.

Scope	Expression	Description
Node status	<code>{{context.\$nodes.<nodeCode>.\$status}}</code>	Processing status of the node

6 AI Agents

Create and Configure AI Agents

Agents are one of the building blocks in AI Agent Studio that can create action plans and interact with users. You can add agents to your workflows for improved efficiency.

Here are the steps to create an agent.

1. Go to **AI Agent Studio** and open the **Resources**  tab.
2. From the **Agents** subtab, select **Add**.
3. On the Agent Settings panel, configure the agent using these tabs.
 - o Details
 - i. Enter a name for the agent. The code is automatically populated based on the name.
 - ii. Select the family to which this agent belongs.
 - iii. Select the product within the family to which this agent belongs.
 - iv. Enter a description for the agent. You can also generate a description by selecting **Generate** and then **Go**.
 - v. In the **Maximum Interactions** field, indicate the number of times the agent can interact with the topics and tools assigned to it.
 - o Prompts
 - i. In the **Agent Role** specify how the agent must behave when responding.
 - ii. In **Prompt** enter the main prompt to guide the agent.
 - iii. Select the **Summarization Mode**.
 - o Default: The summary specified in **Summarization Prompt** is used for summarizing the agent's response.
 - o Custom: Enables you to specify custom summarization instructions in **Summarization Prompt**.
 - o Disabled: Indicates that summarization isn't used for the agent.
 - o LLM: Choose to use the default model or select a model. When using a custom model, specify the model properties.
 - o Input Variables
 - i. Select **Add** to pass inputs to the agent.
 - ii. Specify the name of the variable and its type.
 - iii. Enter a description for the variable.
 - iv. Specify the value of the variable.
 - v. Select **Apply** to save the variable.
 - o Output: Define the overall structure of the agent's output.
4. From the Available Tools panel, add the required tools.
5. From the Available Topics panel, add the required topics.
6. Select **Create** to save the agent.

Your agent is now available for selection when creating workflows for the family and product the agent belongs to.

7 Use Workflow Builder

Create Workflows

A workflow consists of a sequence of nodes used for deterministic, rule-based orchestration of tasks in which every step is preconfigured. Workflows are ideal for scenarios where compliance, repeatability, and governance are essential. The workflow tasks are represented as a connected sequence of nodes. Each node within the workflow performs a defined function, such as extracting data, calling a business object function, running an LLM, or sending an email. The output from each node is seamlessly passed to the following step, ensuring a controlled and efficient process flow.

When designing your workflow, keep these process breakdown steps in mind:

- Model the manual workflow. Start from a high-level process and break it down into smaller, automatable steps.
- Identify task dependencies and group nodes by type. For example, distinguish between back-end logic tasks and those driven by LLMs. Doing so helps to streamline operation and minimize unnecessary LLM usage.
- For each task, define the node type, required input values, expected outputs, and any transformation logic needed.
- For recommendations about improving your agent's efficiency, see *How do I make agents respond faster?*

Let's look at an example to illustrate how a workflow operates in practice. The following workflow is designed to help buyers manage order inquiries within predefined constraints. It allows buyers to retrieve details about their existing orders, including order status, shipment tracking, payment status, and order history. The workflow provides real-time updates on order progress and answers specific questions regarding terms, delivery dates, and other order-related inquiries. For complex issues or inquiries that fall outside its predefined capabilities, the workflow seamlessly escalates the request to a sales representative, ensuring that the buyer receives the necessary support while maintaining an efficient self-service experience.



Workflow Example

Step	Node Name	Description
UserIntent	LLM	Classifies the request as create, enquire, modify, cancel, or not supported.
UnsupportedReturn	Return	Returns the unsupported request response.
UnsupportedIntentOutput	LLM	Generates a buyer-friendly response for unsupported requests.
Start	Start	Begins the workflow.
setBuyerContext	Set Fields	Loads the buyer profile, default order attributes, and supported intents into workflow variables.
RetrieveOrderStatusTypes	Document Retrieval	Retrieves valid sales-order status values from the reference document.
Response	LLM	Wraps the buyer-facing content into the standard response JSON.
requiresLineItemDetails	Condition	Determines whether detailed line-item information is required.
QueryGenerator	LLM	Converts the buyer inquiry into a structured output and identifies support or additional information needs.
ProcessInput	Code	Extracts the current user message from the incoming interaction.
OrderIntentRouter	Switch	Routes the create-order subflow.
isValidRequest	Condition	Checks whether the classified intent is supported.
IsSupportedQuery	Condition	Checks whether the enquiry is supported.
IntentRouter	Switch	Routes supported requests to the Create, Enquiry, Modify, or Cancel flows.
getOrderableItems	Business Object	Searches the orderable product catalog.
Get Sales Order Details	Business Object	Retrieves sales-order headers, totals, and line summaries.
Fetch Sales Order Lines Details	Business Object	Retrieves detailed line information for the first matching sales order.
ExtractOrderInfo	LLM	Extracts product, quantity, order-line information and buyer defaults; determines

Step	Node Name	Description
		whether to create an order, perform product discovery, or request missing fields.
ExtractModifyOrderInfo	LLM	Extracts order header ID, product ID, and requested quantity.
ExtractModifyIntent	LLM	Determines whether the modification is to add or remove a line item.
ExtractCancelOrdreNumber	LLM	Extracts the sales order number to cancel.
End	End	Completes the workflow.
DSL Query Generation	LLM	Generates Elasticsearch query clauses for product discovery.
CreateOutput	LLM	Converts processing results into buyer-friendly content.
createOrderFunction	Business Object	Creates a sales order using buyer defaults and extracted order details.
createErrorOutput	LLM	Explains missing mandatory order fields.
ClarificationOutput	LLM	Generates a clarification message describing supported enquiry options.
Cancel Order	Business Object	Cancels the specified sales order.
AskClarification	Return	Returns the clarification response.
addNewLineToOrder	Business Object	Adds a new product line to an existing sales order.
Add (False Branch)	Pass-through	Passes control to the next node on the false branch.
Add (False Branch)	Pass-through	Passes control to get sales order details.
Add (False Branch)	Pass-through	Passes control to create output.

Here are the steps to create your own workflow.

1. Go to **AI Agent Studio**.
2. From the Workflows tab, select **Add**.

3. Configure the workflow.

Details Tab

Field	Description
Agent Team Name and Code	Enter Agent Team Name and Code. Use letters, numbers, and underscores for the code.
Family	Select the family to which this agent team belongs.
Product	Select the product within the family to which this agent team belongs.
Description	Add a description of the workflow.

LLM Tab

Field	Description
Provider	Select a model that best meets your workload's capability, performance, and cost requirements.

Security Tab

Field	Description
Add	Select the roles which will have access to this workflow.

Chat Experience Tab

You can enable users to upload files directly in chat or from third-party providers and receive responses in their preferred language.

These are the supported file types users can upload to the chat: PDF-Tagged / Scanned, HTML, JSON, MARKDOWN, XML, TXT, DOCX, DOCX-Multimodal, XLSX, CSV, PPTX, JPG, PNG, and ZIP.

Field	Description
Answer in user's language	Select this option to allow users to receive answers in their preferred language.
Enable file upload	Select this option to allow users to upload files from local storage.

Field	Description
Enable third-party file upload	Select this option to allow users to upload files from connected cloud storage accounts. To enable this functionality, add credentials for your provider by navigating to the Credentials tab in AI Agent Studio and completing the Chat Experience setup.

Questions Tab

Field	Description
Starter Questions	Enter initial questions for the workflow.
Follow-up Questions	Enable this option to indicate that the workflow can ask follow-up questions based on the user's conversation history.

Triggers Tab

Create triggers to begin the workflow. Triggers define the conditions or events that start the workflow and identify when and why it begins.

Field	Description
Type	Add the data type of the trigger.

For more information about setting up triggers, see [Configure Workflow Triggers](#).

Variables Tab

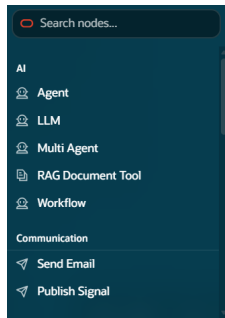
Define workflow-level variables, making them accessible to all nodes within the workflow. Variables are ideal for storing IDs, constants, or any values you need to share and use across different parts of the workflow.

Field	Description
Type	Add the data type of the variable.

Error Handling Tab

Specify where to send an email when a workflow instance runs into a permanent error. You can also use context expressions in these fields for more clarity.

4. Add nodes to the workflow.



Tips

- Connect nodes in sequence to control how input values are received, processed, and forwarded throughout the workflow.
- You can insert new nodes before, after, or between existing nodes as needed.
- For more information about Vector nodes, see [Configure Vector DB Writer and Reader Nodes in a Workflow](#).

5. Publish the workflow.

Users can view the published workflows from the Agents and Applications page. To access this page, go to **Me > Quick Actions > Show More > AI Agent Studio > AI Agent**.

Note: Make sure that your users have access to interact with the AI agents. For information, see [How can I give users access to AI Agents?](#)

Nodes in Workflows

You can use various nodes in your workflows, such as LLMs, agents, condition nodes, and so on.

Here are some of the nodes that are available for workflows:

Type of Node	Node Name
AI	Agent
	LLM
	Multi Agent For more information, see Add Multi Agent Nodes in AI Agent Studio .
	RAG Document Tool
	Workflow
Communication	Send Email
	Publish Signal

Type of Node	Node Name
Data	Business Object Function
	AI Guidance
	Document Processor For more information, see Add a Document Schema to a Document Processing Node of Workflows .
	External REST
	Tool For more information, see Configure Tool Nodes in a Workflow .
	Vector DB Reader
	Vector DB Writer
Logic	Code
	Set Variables
	Policy Nodes
Workflow Control	Human Approval For more information, see Add an Approval Process Channel to a Human Approval Node of Workflows .
	Reference For more information, see Add a Reference Block to a Reference Node of Workflows .
	Reference Block For more information, see Add a Reference Block to a Reference Node of Workflows .
	If Condition
	For Loop
	Parallel
	Return For more information, see Configure a Return Node in a Workflow .
	Switch
	Wait
	While Loop

Settings in Workflows

There are various settings and configurations that you can define while creating your workflows, such as enabling your workflow to be triggered using emails, setting up approval processes, defining document schemas to use in your documents, and so on.

Here are some of the available functionalities:

- Define triggers to activate your workflow. See [Configure Workflow Triggers](#).
- Define approval processes. See [Add an Approval Process Channel to a Human Approval Node of Workflows](#).
- Define document schemas. See [Add a Document Schema to a Document Processing Node of Workflows](#).

Configure Vector DB Writer and Reader Nodes in a Workflow

Vector DB Writer and Vector DB Reader nodes enable workflows to store and retrieve business knowledge using vector embeddings and semantic search. You can use these nodes to build Retrieval-Augmented Generation (RAG) workflows that capture, manage, and reuse enterprise knowledge across business processes. The Vector DB Writer node stores content in a vector index, and the Vector DB Reader node retrieves the most relevant content based on semantic similarity and optional metadata filters.

Operational Limits and Defaults

To ensure efficient vector operations, keep node entries within specified sizing and retrieval limits.

Parameter	Recommendation
Document size per entry	50 KB
Number of retrievals	15 by default, consisting of 10 semantic retrievals and five text retrievals
Maximum results per read	Up to five

Here are some sizing tips:

- Keep the number of retrievals to a minimal amount of relevant items.
- Don't store more than 500 objects per workflow and use strong metadata filters to improve precision.
- Apply a semantic match score threshold before using results. If the scores are low or have no value, implement a fallback path to a deterministic look-up or a human-in-the-loop intervention.
- Avoid noisy, raw, or sensitive data.
- Keep entries fresh with Upsert or Overwrite for updates and Delete for stale records.

Configure a Vector DB Writer Node

Vector DB writer node stores high-value knowledge as embeddings for future semantic retrieval. These nodes are also referred to as vector store nodes. You can use expressions to specify values for supported fields. For more information, see [Expressions](#).

To configure a Vector DB Writer node:

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow and add a Vector DB Writer node to it.
3. Enter a name. The code is automatically populated based on the name.
4. Enter a description for the Vector DB Writer node.
5. Select an error handler node to define failure handling.
6. In the **Operation Type** list, select the operation to perform on the vector index:
 - **Delete** to remove an existing vector entry.
 - **Insert** to create a new vector entry.
 - **Overwrite** to replace an existing vector entry.
 - **Upsert** to update an existing entry or create one if it doesn't already exist.
7. In the **Index Name** field, specify the vector index where the content will be stored.
8. In the **Bulk Mode** field, select **Yes** to enable uploading many documents in a single operation and improve ingestion performance, or select **No** to disable it.
9. In the **Content** field, specify the text that must be converted into vector embeddings.
10. In the **Content Type** field, specify the type of content being stored, such as text or JSON.
11. In the **Document ID** field, specify a unique identifier for the document.
12. In the **Parent Object ID** field, specify the identifier of the parent object that owns the document.
13. In the **Grandparent Object ID** field, specify the identifier of the grandparent object in the record hierarchy.
14. In the **Properties** field, enter a unique metadata property name, select its data type, enter the property value or use an expression for a dynamic value, and then select **Apply**. You can use these properties later to filter vector search results.
15. Publish the workflow.

When the workflow runs, the Vector DB Writer node converts the specified content into vector embeddings and stores them in the selected vector index. Depending on the selected operation type, the node inserts, updates, overwrites, or deletes vector entries in the index. The stored content can later be retrieved by Vector DB Reader node.

Best Practices for Configuring Vector DB Writer Nodes

A vector store works best when you write clear, meaningful, metadata-rich knowledge while avoiding noise, duplication, and topic drift.

Best Practice	Description
Store only public, reusable knowledge	Exclude sensitive, permission-controlled, or private information, for example, personally identifiable information (PII), credentials, confidential documents, or any information requiring permission checks.
Focus on durable, high-value content	Store information that will have long-term value for future workflows, such as case resolutions, structured summaries, or validated insights. Avoid writing noisy, one-off, or ephemeral content, like raw chat logs or temporary instructions.

Best Practice	Description
Clean and normalize before writing	Always standardize content, summarize documents, deduplicate overlaps, remove irrelevant details, and attach key metadata.
Keep data up to date	Vector DB writer must reflect the latest truth. Stale entries decrease accuracy and lead to incorrect answers. - Set auto-refresh triggers to update vectors whenever business objects change (for example, case closures, policy updates, PO resolutions). - Use Overwrite or Upsert to keep summaries up to date. - Delete outdated or unused entries to prevent stale data from accumulating. - Maintain freshness so workflows rely on correct, up-to-date knowledge.
Follow smart update practices	Update vectors in a controlled and intentional manner. Don't create more entries for each update, or store conflicting or outdated data. - Avoid creating new entries for every update. Refine existing data instead of duplicating. - Use Upsert to update and improve existing summaries. - Use Overwrite for significant changes or new, higher-quality content. - Prune redundant or conflicting entries to keep the vector store clean. - Establish clear versioning strategies for policies, product versions, and configuration changes to prevent fragmentation.
Always include rich metadata	Tag with business object IDs, product numbers, version information, and other searchable attributes. Use a consistent metadata schema across agents and indexes.
Prevent data pollution	Check for existing knowledge to avoid duplicates and inconsistent tags. Regularly remove low-value content.
Exclude business object data	Duplicating this data in vector memory creates noise, redundancy, and version drift.

Configure a Vector DB Reader Node

Vector DB Reader nodes retrieve the most relevant knowledge using semantic similarity and metadata filters. You can use expressions to specify values for supported fields. For more information, see [Expressions](#).

To configure a Vector DB Reader node:

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow and add a Vector DB Reader node to it.
3. Enter a name. The code is automatically populated based on the name.
4. Enter a description for the Vector DB Reader node.
5. Select an error handler node to define failure handling.
6. In the **Index Name** field, specify the vector index to search.
7. In the **Query** field, specify the natural language query that describes the information to retrieve.
8. In the **Document ID** field, specify the identifier of a document if you want to retrieve a specific vector entry.
9. In the **Parent Object ID** field, specify the identifier of the parent object that owns the document.

- 10. In the **Grandparent Object ID** field, optionally specify the identifier of the grandparent object in the record hierarchy.
- 11. In the **Data Fields** field, specify the metadata fields that should be returned with the search results.
- 12. In the **Filter Criteria** field, define metadata filters to narrow the search results before semantic ranking is performed.
- 13. In the **Maximum Results** field, specify the maximum number of matching results to return.
- 14. Review the predefined output schema for the selected artifact.
- 15. Publish the workflow.

When the workflow runs, the Vector Read node performs a semantic search on the specified vector index. If filter criteria are specified, only vector entries that satisfy the filters are considered during retrieval. The node returns the highest-ranked results, which can be passed to downstream workflow nodes for further processing.

Best Practices for Configuring Vector Read Nodes

A vector read node works best when you write clear, specific queries that use metadata filters to ensure semantically correct responses, while preventing topic drift and hallucinations.

Best Practice	Description
Write clear, intent-driven queries	Ensure queries are specific and aligned with the exact workflow or agent task. For example, use precise prompts like "What resolved similar issues?" instead of vague ones like "Help me with this."
Use metadata filters for precision	Apply filters (for example, entity type, business object ID, and product number) to target only relevant content.
Validate retrieved results before using them	Check metadata and context before trusting or using results in workflows. This validation prevents incorrect answers, silent workflow failures, and LLM hallucinations grounded in bad evidence.
Set an appropriate <code>maxResults</code> value	Set the <code>maxResults</code> field to return three to five, or more results. Don't depend on a single query result.
Add graceful fallback logic	Implement backup steps, such as business objects or API lookups, to handle cases where queries return no results.
Avoid broad or irrelevant retrieval	Scope queries to just what's needed. Skip irrelevant or noisy indexes to boost performance and accuracy of retrieved results.

Configure Tool Nodes in a Workflow

Tool nodes provide direct access to common tools that workflow agents can use to perform specific tasks. You can use Tool nodes to generate deep links, process chat attachments, identify intent changes, and retrieve information about the signed-in user. The fields available for configuration depend on the selected tool type.

Configure a Tool Node

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow and add a Tool node to it.
3. Enter a name. The code is automatically populated based on the name.
4. Enter a description for the Tool node.
5. Select an error handler node to define failure handling.
6. In the **Type** field, select one of the following tool types:
 - o Deep Link
 - o Connector
 - o Chat Attachments Reader
 - o Intent Change Indicator
 - o User Session
7. Configure the fields for the selected tool type. The configuration fields vary depending on the tool type that you select.

Tool Type	Description
Deep Link	Enables workflows to reference and execute an existing Deep Link tool without recreating it. This tool is used to generate context-aware navigation links to Oracle Fusion Cloud Applications; for example, links to business objects or transactions. This tool mirrors the functionality of Deep Link usage in supervisor agents. It references preexisting Deep Link tools, returns a URL that points to a specific business function, and enables seamless navigation from the workflow to the application UI. Select Deep Link as the type and enter these details: - o Family: Select the family of products to associate this tool node with. - o Product: Select the product you're working with. - o Tool: Select the tool to use in this step. - o Recommend: Select Recommend, add a query and then select Go. This field searches for existing Deep Link tools that match the specified query and returns relevant recommendations.
Connector	Enable adding an existing connector as a tool to connect to internal or external data sources. - o Tool: Select the connector tool to use. - o Function: Select the function to use from the connector.
Chat Attachments Reader	Enables workflows to process files uploaded during a chat session. The output is the extracted content, provided as plain text in Markdown or Base64 format. This tool is also known as Multifile Processor.

Tool Type	Description
	Select Chat Attachments Reader as the type and enter these details: ◦ Conversation Identifier: Specify the variables to uniquely identify conversations and apply them to the attachments. Type {{ to add a variable, or use expressions to specify the conversation identifier. For more information, see Expressions. ◦ Parse Files: Select to enable extraction of readable content or metadata. This option is on by default and is best for most text-based use cases. ◦ Make Raw File Content Available (in base64 format): Select to get the original bytes of the attachment for downstream processing or storage. For example: - Pass files to downstream APIs that accept Base64, like vLLM. - Preserve the original file for storage when the integration expects Base64. - Process nontext or hard-to-parse files like images, scans, and proprietary formats. - Re-create the file later, for example, by decoding Base64 back into the original binary in another step.
Intent Change Indicator	Enables workflows to reset or segment the chat history based on the intent. Use expressions to specify the identifier used to detect intent changes. For more information, see Expressions .
User Session	Enables adding the User Session tool as a node and returns the data from the tool relevant for the signed in user, such as PersonNumber or Username . Select User Session as the type.

8. Publish the workflow.

When the workflow runs, the Tool node runs the configured tool and returns the output generated by the selected tool. The output can be used by downstream workflow nodes to continue processing.

Add a Policy Model to a Policy Node of Workflows

Policy models provide reliable, repeatable, and deterministic policy-based decision making at runtime in AI Agent Studio.

You can create a standalone policy model or use a template. Use a standalone model for unique policies that don't follow an established pattern or use a template to create a family of policy models with common inputs, outputs, and decision logic. Add policy rules by uploading a policy document or entering the rules manually. Then generate code that converts unstructured business policies into decision logic that AI agents use to evaluate rules, perform calculations, and make decisions consistently. After publishing, associate the policy model with a Policy node in a workflow. At runtime, the Policy node evaluates the defined rules using the associated policy model and returns the result.

Create a Standalone Policy Model

1. Go to **AI Agent Studio** and open the **Policy**  tab.
2. Select the **Policy** subtab.
3. Select **Add**.

4. Enter a name and description for the policy. The code is automatically populated based on the name.
5. (Optional) From the **Role** list, select a security role and add it.
6. From the **Template** list, select **Generic**.
7. (Optional) From the **Context** list, select a context to scope the policy to a specific workflow.
8. In the **Prompt** field, enter a prompt to provide additional guidance for generic function generation.
9. Select **Choose File**, and upload one or more policy source documents that contain your business rules.
10. Select **Generate** to generate the functions. It might take some time to generate the function.
11. From **Functions**, select a generated function and review it. View the function name, code, and description in the Info subtab, parameter details in the Input subtab, and return values in the Output subtab.
12. From the Test Cases subtab, generate or upload test cases to validate the function. Test cases verify expected results and pass or fail status before the function is used in workflows.
13. Select **Run Test cases** to verify that the function behaves as expected. Use the **Builder Assistant**  to debug failed test cases.
14. Save and close the policy model or publish it.

Published policy models are available from the Policy subtab for use in Policy nodes within workflows.

Create a Policy Model Using a Template

1. Go to **AI Agent Studio** and select the **Policy** tab.
2. Select the **Policy Templates** subtab.
3. Select **Add**.
4. Enter a name and the code is automatically populated based on the name.
5. In the **Prompt** field, select **Optimize**.
6. Enter a description, and then press **Enter** to generate a prompt that defines the task the template should perform on the input document and specifies the expected output. It might take some time to generate the prompt.
7. From **Functions**, select **Add Function**.
8. Enter a name and description for the function. The code is automatically populated based on the name.
9. For **Prompt**, first select **Optimize**. Then describe the function's role, processing logic, evaluation criteria, and output requirements for analyzing the input data and then press **Enter** to generate a prompt. It might take some time to generate the prompt.
10. In the Input subtab, select **Add Parameter** to add the required parameters.
11. In the Output subtab, define the return output details, and then select **Apply**.
12. Publish the policy template. Published policy templates are available from the Policy Template subtab for reuse across policy models.
13. Select the **Policy** subtab.
14. Select **Add**.
15. Enter a name and description for the policy. The code is automatically populated based on the name.
16. (Optional) Add a security role by selecting a role from the **Role** list and then selecting **Add**.
17. From the **Template** list, select the policy template that you created.
18. (Optional) From the **Context** list, select a context to scope the policy to a specific workflow.
19. Select **Choose File**, and upload one or more policy source documents that contain your business rules.
20. Select **Generate** to generate the functions. It might take some time to generate the function.
21. From **Functions**, select a generated function and review it. View the function name, code, and description in the Info subtab, parameter details in the Input subtab, and return values in the Output subtab.
22. From the Test Cases subtab, generate or upload test cases to validate the function. Test cases verify expected results and pass or fail status before the function is used in workflows.
23. Select **Run Test cases** to verify that the function behaves as expected. Use **Builder Assistant**  to debug failed test cases.

24. Save and close the policy model or publish it.

Published policy models are available from the Policy subtab for use in Policy nodes within workflows.

Associate a Policy Model with a Policy Node

1. From the Workflows tab, edit a workflow and add a Policy node.
2. Enter a name and description for the Policy node. The code is automatically populated based on the name.
3. Select an error handler node to define failure handling.
4. From the **Policy** list, select a published policy model.
5. From the **Function** list, select the policy function to run.
6. If the selected function displays the Parameters field, configure the required input parameters. Use expressions to specify the routing decisions. For more information, see [Expressions](#).
7. Publish the workflow.

When the workflow runs, the Policy node runs the selected policy function and returns the evaluation results to downstream workflow nodes.

Add a Document Schema to a Document Processing Node of Workflows

Document schemas define the structure and fields of data to be extracted from documents in agentic flows.

You can generate a document schema from a sample document, publish it for reuse, and associate it with a document processor node. When the workflow runs, the node uses the selected LLM and schema to extract structured information from documents and return the extracted data in the format defined by the schema.

Create a Document Schema

1. Go to **AI Agent Studio** and open the **Resources**  tab.
2. Select the **Document Schema** subtab.
3. Select **Add**.
4. Enter a name and description for the document schema. The code is automatically populated based on the name.
5. In **Generate Schema From File**, select **Choose File** and upload a supported sample document. See [What file types, limits, and processing capabilities are supported in AI Agent Studio?](#)
6. Select **Generate Schema**.
7. Review the generated schema definition.
8. Publish the document schema, or save and close it to publish later.

Published schemas appear in the Document Schema subtab and can be reused across workflows. If you save and close a document schema, a draft copy appears in the Document Schema subtab. You can open the draft, make changes, and publish it. You can also open a published document schema, go to the draft, update it, and republish the schema.

Associate a Document Schema to a Document Processor Node

1. From the Workflows tab, edit a workflow and add a document processor node to it.

2. Enter a name and description for the document processor. The code is automatically populated based on the name.
3. Select an error handler node to define failure handling.
4. Select a product family and a product.
5. Select the business object that provides the functions and parameters for the node.
6. Select the function to run on the selected business object.
7. In the **Data Extraction** section, set **Use LLM for data extraction** to **Yes**.
8. Use the default LLM configured for the workflow or select another model.
9. From the **Schema Template** list, select a published document schema.
10. Publish the workflow.

When the workflow runs, the document processor uses the selected schema and LLM to extract information from the document. The extracted data is returned in the format defined by the schema, so workflow nodes, applications, or agents can process it more easily.

Add a Reference Block to a Reference Node of Workflows

Reference blocks define reusable logic in agentic workflows.

You can include artifacts such as agents, LLMs, tools, and other workflow nodes in a reference block, associate the block with a reference node, and reuse that logic multiple times in a workflow. When the workflow runs, the reference node uses the logic defined in the reference block.

Create a Reference Block

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow and add a Reference Block node to it.
3. Enter a name and description for the reference block. The code is automatically populated based on the name.
4. Select an error handler node to define failure handling.
5. Add any required parameter names and their data types.
6. Add the nodes and logic that you want to reuse within the workflow.

The reference block is now available for use by reference nodes in the workflow.

Associate a Reference Block With a Reference Node

1. From the Workflows tab of AI Agent Studio, edit a workflow and add a Reference node to it.
2. Enter a name and description for the reference node. The code is automatically populated based on the name.
3. Select an error handler node to define failure handling.
4. From the **Reference Block** list, select the reference block that you want to use.
5. Add the input values required by the reference block.
6. Leave the output schema blank so that the node can use the output of the reference block.
7. Publish the workflow.

When the workflow runs, the reference node runs the selected reference block and returns the output to the calling node.

Add Multi Agent Nodes in AI Agent Studio

You can use Multi Agent nodes in which a supervisor agent routes requests to specialized worker agents.

Configure each worker agent with a specific responsibility and use the supervisor prompt and output specification to control routing and response consistency. In the node's properties, you can define how the supervisor agent evaluates incoming requests and selects the appropriate worker agent.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Edit a workflow and add a **Multi Agent** node to it.
3. Enter a name and description for the node. The code is automatically populated based on the name.
4. Select an error handler node to define failure handling.
5. Specify the maximum number of interactions the node can run.
6. Specify whether the agent must respond in the user's language. Otherwise, the workflow default language is applied.
7. Specify whether the agent must use the hallucination guardrail prompt every time it runs.
8. Specify whether to include chat history as context for the supervisor and worker agents.
9. In **Agent Persona and Role**, enter the supervisor agent's role, tone, and behavior instructions.
10. In **Prompt**, use expressions to specify the routing decisions. For more information, see [Expressions](#).
11. Select and add topics to define the areas covered by the Multi Agent node.
12. Specify whether to enable summarization to generate a conversation summary after routing completes. If enabled, use expressions to specify the summarization prompt.
13. In **LLM**, select a language model.
14. Define the output specification for downstream steps and expression tools.
15. Add **Agent** nodes within the Multi Agent node as worker agents.
16. Enter a name. The code is automatically populated based on the name.
17. From the **Agent** list, select an existing agent as worker agent for the Multi Agent node. The properties of the required fields are populated with the values defined in the selected agent.
18. Publish the workflow.

When the workflow runs, the supervisor agent receives the user request, evaluates the intent, and routes the task to the most appropriate worker agent. The selected worker agent performs the task and returns the response. If a request spans multiple topics, the supervisor agent can coordinate across multiple worker agents and return a consolidated response.

Configure a Return Node in a Workflow

Configure a Return node to stop processing a workflow branch and return a value to the calling node.

Return nodes are typically used in reusable workflow components, such as reference blocks. Define the return value directly or use expressions to reference workflow variables and outputs from previous nodes.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Edit a workflow and add a Return node to it.
3. Enter a name and description for the node. The code is populated automatically based on the name.

4. Select an error handler node to define failure handling.
5. Use expressions to specify the value the Return node must send to the calling node. For more information, see *Expressions*.
6. Publish the workflow.

When the workflow reaches the Return node, the current workflow branch ends, and the configured value is returned to the calling node. The returned value can then be used by downstream workflow nodes, applications, agents, or other workflow components.

Configure a Send Email Node in a Workflow

Use the Send Email node to send email notifications from a workflow. You can configure the email recipients, subject, message content, and attachments. The node also supports context-sensitive expressions to dynamically populate the email fields. After you configure the node, connect it to the appropriate workflow path to send emails when the workflow runs.

1. Go to **AI Agent Studio**, and open the **Workflows**  tab.
2. Edit a workflow and add a Send Email node to it.
3. Enter a name and description for the node. The code is populated automatically based on the name.
4. From the **Error Handler** list, select an error handler node to define failure handling.
5. In the **Recipients** field, enter the email addresses or use expressions to specify the value that must be send to the calling node. For more information, see *Expressions*.
6. In the **CC** field, specify one or more email addresses or use expressions.
7. In the **Email Subject** field, enter the subject of the email. You can enter static text or use expressions.
8. In the **Email Template** field, enter the content of the email. You can add a static message, and if needed, you can also use expressions to include values dynamically.
9. In the **Attachments** field, specify one or more attachments. You can enter a value directly or use expressions.
10. Publish the workflow.

When the workflow reaches the Send Email node, the email is sent to the specified recipients.

Configure Workflow Triggers

Workflow triggers are automated, predefined events or conditions that initiate a workflow when specific criteria are met.

Note: Triggers activate after the workflow is published.

Webhook

A webhook is an automated, event-driven mechanism for sending real-time data from an application over HTTP. You can configure a webhook trigger to call a workflow.

1. Go to **AI Agent Studio**.

2. From the Workflows tab, edit a workflow.
3. Select **Settings**  and go to the **Triggers** tab.
4. Select **Add** for webhook trigger, enter a name, select type, and then select **Apply**.
5. Publish the workflow.

Email

You can configure an email trigger to call a workflow when an email is received in a monitored account. Each workflow can monitor only one unique inbound email account. You can't use the same inbound email account for another workflow.

To allow the workflow to be triggered by incoming emails:

1. Register the inbound email account in **AI Agent Studio**. See [Add Email Accounts](#).
2. Go to **AI Agent Studio**.
3. From the Workflows tab, edit a workflow.
4. Select **Settings**  and go to the **Triggers** tab.
5. Select your registered inbound email account.
6. Publish the workflow.

To verify that emails received in the selected inbound email account will now trigger this workflow:

1. Go to the **Monitoring and Evaluation** tab and open the **Activities** subtab.
2. Go to **Inbound Emails** to view the contents of the trigger email messages and verify that the messages are initiating the workflow correctly.

Schedule

You can configure a scheduled trigger based on a defined time interval, to call a published workflow without user interaction or API calls. Scheduled triggers are useful when workflows need to run regularly or at fixed intervals.

Note: Make sure you have the FAI Batch Job Manager (`ORA_DR_FAI_BATCH_JOB_MANAGER_DUTY`) permission assigned in the Security Console.

To allow the workflow to be triggered at a scheduled date and time:

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow.
3. Select **Settings**  and go to the **Triggers** tab.
4. Select **Add** for schedule triggers and choose the type of schedule to create. Based on the selected type, specify other details.

Type	Description
Interval	Run the workflow between fixed time periods, for example, every 100 minutes, every 10 hours, or every 5 days.
Recurrence	Run once at a specific time or repeatedly on calendar patterns, such as daily, weekly, or monthly. Enter a start time and an optional date and time for the recurrence to end.

5. Publish the workflow.

To verify that your scheduled trigger is starting the workflow:

1. Go to the **Monitoring and Evaluation** tab and open the **Activities** subtab.
2. Go to **Schedules** to view the entry and verify the scheduled trigger is initiating the workflow correctly.

Signals

You can configure a signal trigger to call a workflow when a subscribed signal is received. A signal is a governed event contract that you define in AI Agent Studio and subscribe to from a workflow. Signal triggers are useful when workflows need to respond automatically to business events without user interaction or API calls.

To create a signal in AI Agent Studio:

1. Go to **AI Agent Studio**.
2. Select the **Resources**  tab and open the **Signals** subtab.
3. Add a new signal definition.
4. Enter the signal code, version, name, status, description, and payload schema.
5. Select **Create**.

Created signals appear in the Signals subtab and can be reused across workflows.

To subscribe to a signal:

1. Go to **AI Agent Studio**.
2. From the Workflows tab, edit a workflow.
3. Select **Settings**  and go to the **Triggers** tab.
4. In **Signals** section, select **Add**.
5. Select an active signal definition.
6. In **Max attempts** field, select the maximum number of times delivery can be attempted before it's exhausted.
7. In **Retry delay** field, select the base delay before a failed delivery is retried.
8. In **Backoff policy** field, select a backoff policy to determine how the retry delay changes after each failed delivery attempt.
9. Select **Apply** and then publish the workflow.

Your workflow is now subscribed to the signal you selected. It can start automatically when that signal is published.

Add an Approval Process Channel to a Human Approval Node of Workflows

Approval processes determine how approval requests are routed, who must approve them, how approvers are notified, and which rules or conditions apply.

Associate a published approval process with a human approval node by selecting the Approval Process channel. Approval process channels provide a structured, reusable framework for managing approvals in agentic flows. At runtime, the workflow is paused while the approval request is routed through the selected approval process. The workflow resumes after the required approval outcome is received.

Create an Approval Process

1. Go to **AI Agent Studio**.
2. Open the **Approvals**  tab, and select **Add**.
3. Enter a name. The code is automatically populated based on the name.
4. To add a description, first select **Generate**, add a short description, and then select **Go**.
5. Modify the description if required, and select **Create**.
6. From **Notification Channels**, select the channels to use for approval notifications. For an email, specify the email account.
7. Add the approval payload fields that the approval process requires.
8. Add the approval levels and configure the approver details.
 - a. Enter a level name and description.
 - b. Select the approver type and specify the approver details.
 - c. To add a rule logic, first select **Generate rule**, add a logic in plain language, and then select **Go**.
9. Add a condition by selecting a field, operator, and value.
10. Publish the approval process.

Published approval processes appear in the Approvals tab and can be selected in human approval nodes within workflows.

Associate an Approval Process to a Human Approval Node

1. From the Workflows tab of AI Agent Studio, edit a workflow and add a human approval node to it.
2. Enter a name and description for the human approval node. The code is automatically populated based on the name.
3. Select an error handler node to define failure handling.
4. From the **Channel** list, select **Approval Process**.
5. (Optional) Enable feedback and configure the feedback options.
6. If you select **Yes**, choose the node to return to when feedback or retries are enabled.
7. Specify the maximum number of times the node can run.
8. From the **Approval Process** list, select a published approval process.
9. You can provide additional details such as subject and message. Otherwise, the node uses the properties of the selected approval process.
10. Publish the workflow.

When the workflow runs, the human approval node sends the approval request through the selected approval process.

Add and Reuse Functions in AI Agent Studio

JavaScript function templates enable you to define common logic once and use it across multiple agentic workflows.

Create function templates from the Functions subtab under the Resources tab in AI Agent Studio. Each template can include one or more JavaScript functions and generated test cases to validate function behavior before publishing. After a function template is published, it becomes available from the Functions subtab for use in Code nodes across agentic workflows.

Create a Function Template

1. Go to **AI Agent Studio** and open the **Resources**  tab.
2. Select the **Functions** subtab.
3. Select **Add**.
4. Enter a name and description for the new function template. The code is automatically populated based on the name.
5. Select **Add** to create a function.
6. Enter a name and description for the function. The code is automatically populated based on the name.
7. In the Code subtab, write the JavaScript source code manually or generate the function code.
 - a. To generate the function code, select **Optimize**, describe the logic that you want the function to perform, and then press the **Enter** key. The function code is generated and added to the Code subtab. It might take some time to generate the code. After the code is added, you can view the function parameters in the Input subtab and the return values in the Output subtab.
 - b. From the Code subtab, select **Run**. If an error is displayed, review and modify the generated code.
8. From the Test Cases subtab, generate or upload test cases to validate the function. Test cases verify expected results using sample inputs and include expected output, actual output, and pass or fail status before the function is used in workflows.
9. Select **Run Test Cases** to verify that the function behaves as expected.
10. Save and close the function template or publish it.

Published function templates are available from the Functions subtab for use in Code nodes within workflows.

Use a Reusable Function in a Workflow

1. From the Workflows tab, edit a workflow and add a Code node.
2. Enter a name and description for the code node. The code is automatically populated based on the name.
3. Select an error handler node to define failure handling.
4. From **Source**, select **Reusable Functions**.
5. From the next two lists, select the published function template and then the function you want to use.
6. (Optional) Review and update the output specification.
7. Publish the workflow.

When the workflow runs, the Code node calls the selected reusable function from the published function template and returns the function output to the next node in the workflow.

Configure an AI Guidance Node in a Workflow

Use the AI Guidance node to write guidance to or read guidance from the AI Guidance Store.

The AI Guidance Store provides shared storage for enterprise guidance, such as summaries, recommendations, alerts, insights, and actions for consistent use across agentic experiences.

You can associate guidance with a business object and associated business record, using the business object's record-resolution and security model. You can also associate guidance with an agentic application or an Ask Oracle plugin.

1. Go to **AI Agent Studio**.

2. From the Workflows tab, edit a workflow and add an AI Guidance node to it.
3. Enter a name and description for the node. The code is automatically populated based on the name.
4. Select an error handler node to define failure handling.
5. From **Operation**, select the operation that you want the node to perform. Read mode is selected by default.
 - **Read:** This operation retrieves existing guidance for the applicable context. For a business object, the node validates the current user's access to the business object before retrieving its guidance.
 - **Write:** Select this operation when the workflow has generated guidance that you want to store and make available to other users. Specify the guidance to store and the applicable attributes for the guidance.
6. In **Reference Type**, specify the reference with which the guidance is associated. Select one of the following tabs as per your requirement:
 - **Business Object:** Associate the guidance with a specific business object and business record.
 - From the **Family** list, select the applicable product family.
 - From the **Product** list, select the applicable product.
 - From the **Business Object** list, select a business object enabled for AI Guidance Store.
 - In the **Record Key** field, enter the value that identifies the business record or use an expression to specify the value. For more information, see *Expressions*.
 - **Agentic Application:** Associate the guidance with an agentic application.
 - From the **Application Code** list, select the agentic application to associate with the guidance.
 - **Ask Oracle Plugin:** Associate the guidance with an Ask Oracle plugin.
 - From the **Plugin Code** list, select the Ask Oracle plugin to associate with the guidance.
7. In the **Filter** field, enter the criteria to filter the applicable guidance or use an expression to specify the filter criteria. For more information, see *Expressions*.
8. In the **Returned Reference Filter** field, enter the criteria to filter the references returned by the node or use an expression to specify the filter criteria. For more information, see *Expressions*.
9. In the **Target Persona** field, enter the persona for which the guidance applies or use an expression to specify the value. Use ALL when the guidance applies to all supported personas. For more information, see *Expressions*.
10. Review **Output Specification**. The output schema is predefined for the selected artifact and is read-only.
11. Publish the workflow.

When the workflow runs, a Write operation stores the generated guidance in the AI Guidance Store and associates it with the configured reference. For a business object, the business object is the primary reference for the stored guidance, making the guidance available to subsequent consumers.

A Read operation retrieves existing guidance for the configured context. For a business object reference, the node resolves the applicable business record and validates the current user's access to that object before retrieving its guidance. The configured filters and target persona further determine the applicable guidance returned by the node.

8 Use Applications Builder

Overview of Agentic Applications

Agentic apps help in combining your agents to form active, intelligent apps, enabling their use as your decision-making partners. You can consider the agents as workers and the agentic app is the productized solution that bundles those workers with the integrations, knowledge, and controls that are needed. The apps can include multiple agents, each of which is purpose-built for a specific category or domain.

Instead of asking one agent about everything from sales performance to inventory management, you can build specialized agents, for example, a Sales agent, an Inventory agent, a Finance agent; each with deep expertise in their domain. These agents can be used to create your agentic app. Doing so provides better insights and actions because each agent can maintain focused context, domain-specific reasoning, and tailored prompting for its area of responsibility. You can also scale it up by adding additional agents as needed.

The best agentic apps don't simply observe and report; they analyze, prioritize, and recommend actions that drive business forward.

- **Proactive alerting over passive monitoring:** Agents identify anomalies, opportunities, and risks in real-time, surfacing them with context and urgency. Instead of making users hunt for problems, the agents bring critical issues directly to their attention when they matter.
- **Decision support, not data dumps:** Apps filter signals from noise by analyzing situations and presenting only actionable decisions. Instead of providing users with every data point, provides synthesized insights with clear recommendations and trade offs.
- **Action-oriented intelligence:** Every agent interaction leads toward concrete actions, such as, approve a proposal, adjust a strategy, resolve a blocker. If it's purely informational, we can use an API or dashboard instead of agentic apps.
- **Context-aware prioritization:** Apps understand business goals and user workflows to determine what's urgent versus what can wait. Not all alerts are equal; agents should respect user's time and attention.

Here are the main pillars of agentic apps:

- **Information Displays**

Shows you data. These are agent-generated visualizations, such as lists, charts, tables, that show relevant data in real-time using predefined patterns. Agents built for agentic apps answer questions by creating information displays, except when intent clarification, confirmation, or a brief response is more appropriate. Each information display is generated with a topic key that captures the context of the user request. As conversation topics shift, the page template may hide or remove displays that are no longer relevant.

Agents must be configured to support specific information display types, granting them the capability to generate the displays. The implementation details are provided using prompting. At application startup, each agent may generate an initial information display that's shown to the user immediately. The agent determines whether to show an initial display and which display to show, and the host application can override either decision through prompting.

- **Advisors**

Answers your questions. Advisors are agents that answer user questions either independently or in coordination with other agents.

- Single agent mode: Users focus on a specific agent as their advisor
- Multi agent mode: Questions are routed to two or more capable agents, and outputs are consolidated into a unified response

Agents typically respond with both answers and supporting information displays. By default, information displays prioritize showing detailed data within the visualization itself, with accompanying text summarizing key insights. When a single agent acts as advisor, only that single agent produces an information display. In multi-agent scenarios, only selected agents generate displays, with the final answer summarizing their collective output.

- **Actions**

Helps you make decisions. Actions are agent-generated decision points that require human approval. Here are the artifacts in actions:

- Title describing the topic
- Description of what the action performs
- Instructions for the agent to execute if the action is called (including commands and parameters)

This configuration is agent-generated and not modifiable by users. Actions are generally associated with a single agent, and when the action is called, that same agent processes the action. The agent either performs the action immediately or engages the user if further clarification is needed.

Agents must be explicitly configured to generate actions. The generation logic and invocation handling are defined in the agent's prompting. At application startup, each agent can produce an initial list of priority actions that represent immediate decision points an app user may make. When answering questions, agents can include one or more actions in their response, which are then presented to the user.

Information displays may also expose actions directly within their pattern. They work identically and their presentation is defined by the associated pattern.

- **Communications**

Helps you reach out. Communications are agent-suggested outbound messages, for example, emails or Slack messages that follow consistent predefined templates. Here are the fields in Communications:

- Title and description explaining what the communication does and who it targets
- Template parameters used to compose the message

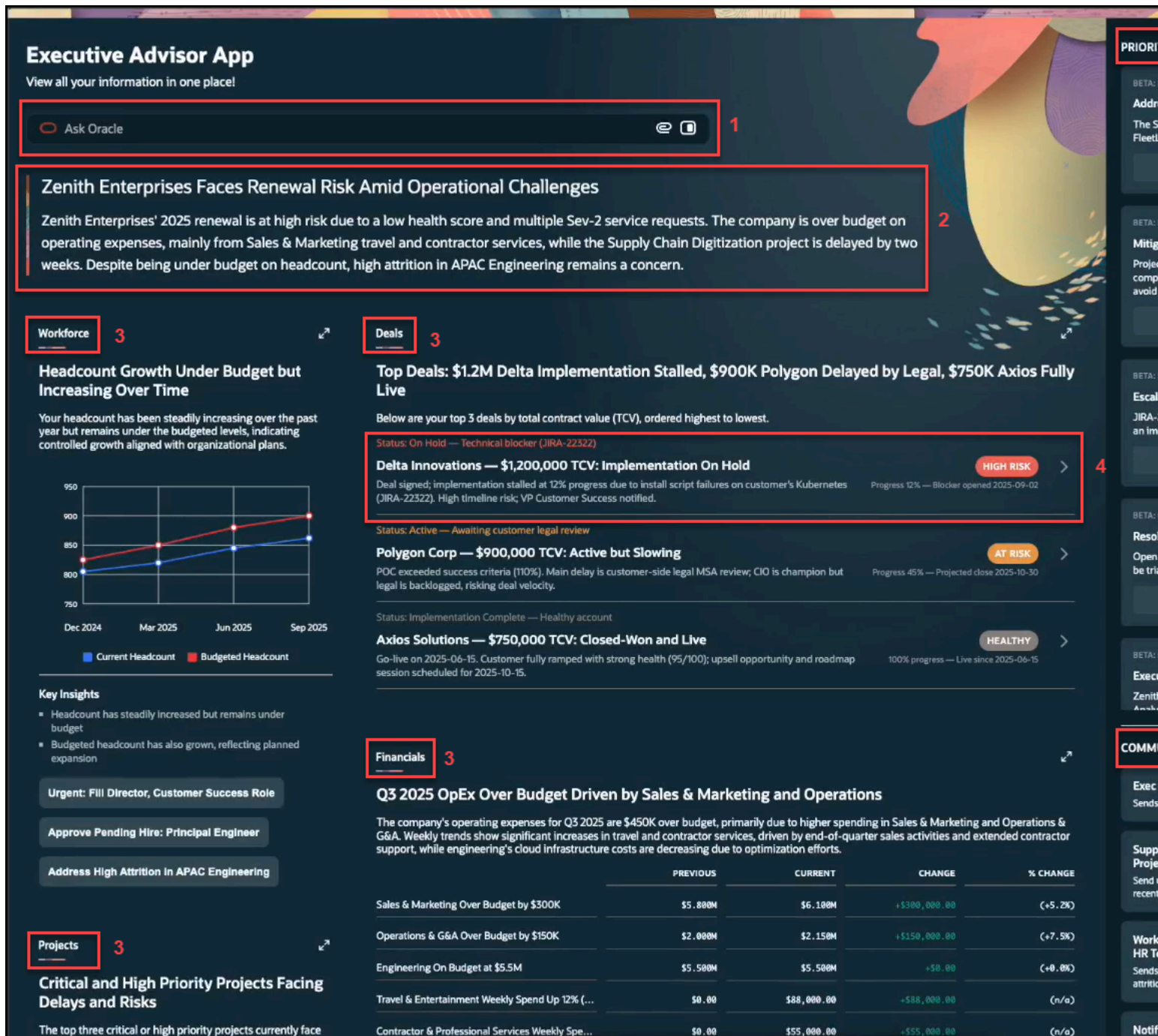
Communications always initiate outbound messages; they never intercept or receive incoming messages. The communication suggestions are routed to a specific agent for delivery. This agent doesn't need to be an advisory agent. It can be a generic, reusable agent shared across applications. The designated agent then processes the communication request, performs any additional work, and delivers the message using the appropriate tool, such as email, Slack, and Microsoft Teams.

All communications adhere to the templated patterns for consistency and uniformity. Messages aren't free-form; they conform to specific templates with defined parameter sets. Agents can produce communication suggestions when responding to inquiries and at app initialization. They can prefill template parameters based on their knowledge and prompting. Communication suggestions can also be app-configured and associated with specific agents or all available agents.

When a communication item is selected, the associated agents attempt to populate any unfilled template parameters. The app user can then modify these values before the message is sent to the template engine for final processing.

Anatomy of an Agentic Application

Here's an example of an Executive Advisor agentic app that consists of four agents.



Callout Number	Name	Description
1	Ask Oracle Advisor	Lets you ask natural language questions against the set of agents included in the app.
2	Summary Section	Displays the most critical information across all the domains in the app's agents. The summary is typically the first content users read when opening the app.
3	Agents	The example agentic app includes four agents, each represented by a tile on the dashboard. An agent can provide its own headline to convey the most important information for the app user. Users can zoom in on any agent to keep focus on that agent. Any subsequent questions, actions, and communications are then scoped to the selected agent.
4	Widget	A widget generated by an agent from the widget library. Each widget has space for the agent to provide its own insights, and capabilities that the user can choose from. Widgets can also have agent-specific actions that the user can act upon.
5	Priority Actions	App-specific critical information that require the user's immediate attention and provide options for the user to take action.
6	Communications	Outbound message suggestions that may be predefined by the app or generated by the LLM. It can start a flow allowing the user to construct the content and send the communication.

Create and Configure Workflows for Use in Agentic Applications

To build agentic apps with workflows, first configure the workflows for optimal results. To use a workflow in agentic apps, make sure to publish the workflow at least once. After publishing, you can edit the workflow, and use the draft version of your workflow for your agentic app.

Best Practices

- Make sure that the description of your workflow clearly defines the workflow's capabilities and specialization. When multiple workflows are engaged, this description is used to help identify which workflow is best suited to handle a given query.
- Design smaller, topic-focused workflows. You can combine multiple workflows as needed within the agentic app.
- Keep workflows lean to reduce latency. Limit the number of LLM and agent nodes where possible to reduce perceived delay when resources are requested from the workflow.
- Pass user input and history explicitly. In workflows, conversation history and user input aren't automatically passed to internal LLM, agent, or workflow nodes. Use a code node or variable assignment to pass what you need (for example, using `$context.$system.$inputMessage`), and then reference those variables in subsequent LLM or agent calls.
- When possible, end your workflow with an agent or LLM node to help prevent output streaming.

Create and Enable Workflows for Using in Agentic Applications

An agentic app is built using workflows. These workflows provide the response when the agentic app requests for displays, actions, communications, summary text, subtitle text, or user-query answers. The recommended method is for the workflow to receive the request from the agentic app using the `OraMessageHint` webhook variable. Create your workflow structure to receive the webhook message, branch the workflow, and then provide the needed output to the agentic app.

To use workflows in agentic apps, you must first enable the workflows from AI Agent Studio.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to edit the workflow you want to use in agentic apps.
3. Select **Settings** .
4. From the **Details** tab, enable **Expose to agentic apps**.
5. Publish your workflow.

OraMessageHint Variable

`OraMessageHint` is the primary app-aware workflow variable, and its value can be retrieved using the expression `$context.$app.$OraMessageHint`. It tells your workflow why the app is calling it. In prompt templates, use expressions such as `{{ $context.$app.$OraMessageHint }}`. In Code nodes, use the value as `$context.$app.$OraMessageHint`.

Here are some common values for `OraMessageHint`:

Value	Description
Summary	Indicates that the app is asking for summary text. Return short summary text for that agent's contribution.
InitSubtitle	Indicates that the app is asking for the dynamic subtitle. Return the subtitle text itself.

Value	Description
InitDisplay	Indicates that the app is asking for startup displays. Return one or more oraInfoDisplay blocks, and it may also include actions and communication suggestions when that improves the startup experience.
InitActions	Indicates that the app is asking for startup actions. Return zero or more action suggestions.
InitCommunications	Indicates that the app is asking for startup communication suggestions. Return zero or more oraComms blocks.
Query	Indicates that the user asked something through the app. Use the input message and answer the user. This response can also include displays, actions, and communications when appropriate.
InvokeAction	Indicates that a UI action sent work back to the workflow. Use this for follow-up handling after a widget command triggers an action path that sends a command to an agent.
AdditionalContent	Indicates that the app is loading an extra panel under a focused agent. Use OraPanelName to tell which panel is being requested.
FillParameters	Indicates that the app is asking one or more agents to fill communication parameters before generating or sending a response.
SendCommunication	Indicates that the app is asking a target workflow to send or complete a communication flow.

Expressions Useful for Agentic Applications

Here are some expressions that you can use in your workflows:

Expression	Description
<code>{{ \$context.\$app.\$OraMessageHint }}</code>	App stage that's calling the workflow
<code>{{ \$context.\$system.\$inputMessage }}</code>	Current user message or app-supplied input for the current stage
<code>{{ \$context.\$app.\$OraAppContext }}</code>	Current app context value
<code>{{ \$context.\$app.\$OraUserContext.fullName }}</code>	Current user's complete name
<code>{{ \$context.\$app.\$OraUserContext.userId }}</code>	Current user's ID
<code>{{ \$context.\$app.\$OraAttachments }}</code>	Attachments available to the current app query flow
<code>{{ \$context.\$app.\$OraAction }}</code>	Action payload or command supplied to the workflow action path

Expression	Description
<code>{{ \$context.\$app.\$OraCommParamsToFill }}</code>	Communication parameters that must be filled
<code>{{ \$context.\$app.\$OraCommNonTemplateParamList }}</code>	Non template communication parameters for sender-style flows
<code>{{ \$context.\$app.\$OraPanelName }}</code>	Additional panel that's requesting content
<code>{{ \$context.\$app.\$OraUiContext }}</code>	What the user is currently viewing, including widget summary updates

Here's an example user context object:

```
{
  "fullName": "John Doe",
  "userId": "jdoe1212",
  "userName": "John.Doe@example.com",
  "guid": "sfsdf332323fdsf"
}
```

Variables in Agentic Applications

Multiple variables can be used in workflows created for agentic apps.

Variable Name	How the Value is Populated
<code>OraMessageHint</code>	Set by the app runtime based on what the app is doing right now: startup display load, startup actions, communications, query, subtitle, and so on.
<code>inputMessage</code>	The current message for that request. For a user query, it's the user prompt. For some action and communication flows, it's the command or content being sent into the workflow.
<code>OraAppContext</code>	Passed from the current app context that may come from the app URL, from app-to-app navigation, or from an action that switches context.
<code>OraUserContext</code>	Passed from the signed-in user and is available to app-aware workflows.
<code>OraAttachments</code>	Populated for query flows when the conversation includes attachments.
<code>OraPanelName</code>	Populated only when the app is loading additional panel content for a focused agent.
<code>OraCommParamsToFill</code>	Populated when the app is asking agents to fill in communication parameters.
<code>OraCommNonTemplateParamList</code>	Available for communication sender flows that need non template parameter information.
<code>OraUiContext</code>	Populated when the app runtime sends UI specific context along with the workflow request, including <code>init</code> display panels.

Enable Workflows to Generate Widgets in Agentic Applications

In your workflow, the LLM and agent nodes can generate widgets. Initial graphics and insight displays are produced by agent output blocks that carry a `patternId` and a widget-specific `config`. In App Builder, the display prompt tells the runtime agent what widget output to generate.

Enable Workflows to Generate Widgets

To enable workflows to generate widgets, do these steps:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to edit the workflow you want to generate widgets.
3. Go to the LLM or agent node that should generate widgets.
4. Edit the node and open the **App Experience** tab.
5. Select the options to enable widgets for actions and communications as needed.
6. Select the types of widget that are to be generated by the workflow.
7. Add instructions for the widget.

Types of Widgets

Here are the types of widgets an LLM or agent node can generate:

Type of Widget	Sample Prompt	Details	Example
Card	Use a <code>cardWidget</code> to present one important update as a clean, scannable card. Include a short subject, an optional subtitle for context, a brief summary, and if helpful a status line or time stamp, so the update feels informative without being crowded.	• Use it for: a single alert, update, or status card • Required: <code>subject</code> • Common fields: <code>summary</code>, <code>timestamp</code>, <code>badgeText</code>, <code>variant</code> • Actions: <code>link</code> and optional <code>additionalLink</code>, each with <code>text</code> and <code>action</code> • Variant values: <code>error</code>, <code>warning</code>, <code>info</code>, or omitted for neutral Use commands such as <code>ora.Invoke(...)</code>.	<pre><oraInfoDisplay key="maintenance-card"> { "patternId": "cardWidget", "config": { "subject": "Maintenance Complete", "summary": "Scheduled database maintenance finished successfully.", "timestamp": "1 hour ago", "badgeText": "Info", "variant": "info", "link": { "text": "View Details", "action": "ora.Invoke(\"viewMaintenanceSummary\")" } } } </oraInfoDisplay></pre>
Multi Card	Use a <code>multiCardWidget</code> to display multiple cards in one widget. For example, for generic alerts or status summaries.	• Use it for: multiple peer cards in one widget when a table or message list would be the wrong shape • Required: <code>cards []</code>	<pre><oraInfoDisplay key="ops- status-cards"> { "patternId": "multiCardWidget", "config": {</pre>

Type of Widget	Sample Prompt	Details	Example
		Optional top-level fields: <code>layoutMode</code> layoutMode values: <code>default</code>, <code>collapsed</code>, <code>single-col</code> cards[] entries: each card uses the same fields as <code>cardWidget</code> including <code>subject</code>, <code>summary</code>, <code>optional subtitle</code>, <code>status</code>, <code>statusPriority</code>, <code>timestamp</code>, <code>badgeText</code>, <code>link</code>, <code>additionalLink</code>, and <code>variant</code> Card color values: use the same <code>variant</code> and <code>statusPriority</code> values as <code>cardWidget</code>, including <code>success</code> and <code>neutral</code> Use <code>layoutMode</code> only if you need a compact two-column layout (<code>collapsed</code>) or a one-card-per-row layout (<code>single-col</code>). Use the top-level widget title and text for shared context, and keep card-specific details inside each <code>cards[]</code> entry.	<pre>"cards": [{ "subject": "Database Cluster", "subtitle": "Primary region", "summary": "Replication lag returned to normal after failover testing.", "status": "Healthy", "statusPriority": "success", "timestamp": "8 min ago", "badgeText": "Info", "link": { "text": "View Details", "action": "ora.Invoke(\"openDatabaseStatus \")" } }, { "subject": "API Capacity", "subtitle": "Public gateway", "summary": "Traffic remains elevated and is close to the scaling threshold.", "status": "Watching autoscale behavior", "statusPriority": "warning", "timestamp": "12 min ago", "badgeText": "Warning", "variant": "warning" }] } </oraInfoDisplay></pre>
Artifact Preview	Use a <code>artifactPreviewWidget</code> when the artifact content itself should be previewed inline and opened in the artifact viewer. For generic alerts or status summaries, use <code>cardWidget</code> or <code>multiCardWidget</code> .	Use it for: reviewable text, rich text, structured rich text, PDF, or URL artifacts that should open directly in the artifact viewer Required: <code>items[]</code> Each item requires: <code>artifactId</code>, <code>title</code>, <code>mode</code>, and <code>content</code> mode values: <code>preview</code>, <code>edit</code>, <code>custom</code> Optional item fields: <code>subtitle</code>, <code>actionText</code> action: only for custom items content shape: mirrors the artifact viewer / <code>editArtifact</code> payload	<pre><oraInfoDisplay key="artifact-previews"> { "patternId": "artifactPreviewWidget", "properties": { "items": [{ "artifactId": "customer- note-rich-text", "title": "Customer Note", "subtitle": "Standard rich text draft", "mode": "edit", "content": { "id": "customer-note-1", "type": "richText", "title": "Customer Note", "value": "<p>Team,</</pre>

Type of Widget	Sample Prompt	Details	Example
		with id, type, optional title, value, optional sections, url, mediaType, highlightText, commitText, and optional metadata For preview and edit, don't use an action property. The runtime opens the artifact viewer directly from the item's content payload. Only custom items run an explicit command. preview opens the viewer in read-only mode, and edit opens the viewer in editable mode. On save, the runtime returns the edited value using the normal oraFormSubmit format and adds the item's artifactId under metadata.artifactId. Some artifact types are not writable. For example, PDFs and URLs are always treated as read-only, so mode: "edit" behaves the same as mode: "preview" for PDF and URL content. For type: "url", provide url. The preview card and artifact viewer show the URL in an iframe, and no edited value is submitted back to the workflow. For type: "structuredRichText", provide sections instead of value. Each section should include name, locked, and text. Locked sections remain visible in the editor but cannot be modified. The runtime renders the sections into the rich text editor using those locked boundaries.	<pre> strong></p><p>Please review the latest customer note before the renewal call.</ p>Pricing review is still openLegal requested one contract editCustomer wants final confirmation by Friday", "commitText": "Save", "metadata": { "source": "customer-note" } }, { "artifactId": "qbr-email- preview", "title": "QBR Customer Email", "subtitle": "Structured rich text draft", "mode": "edit", "content": { "id": "qbr-follow-up", "type": "structuredRichText", "title": "QBR Follow-Up Email", "sections": [{ "name": "Greeting", "locked": false, "text": "<p>Hi Jordan,</p>" }, { "name": "Summary", "locked": false, "text": "<p>Thanks again for the time today...</p>" }, { "name": "Closing", "locked": true, "text": "<p>Best,
Account Team</p>" },], "commitText": "Save", "metadata": { "source": "qbr" } }, { "artifactId": "renewal- brief-pdf", "title": "Renewal Brief", "subtitle": "Executive package", "mode": "preview", "content": { </pre>

Type of Widget	Sample Prompt	Details	Example
			<pre>"id": "renewal-brief", "type": "pdf", "title": "Renewal Brief", "url": "/artifacts/ renewal-brief.pdf", "mediaType": "binary" }, { "artifactId": "order- dashboard-url", "title": "Order Dashboard", "subtitle": "Live dashboard", "mode": "preview", "content": { "id": "order-dashboard", "type": "url", "title": "Order Dashboard", "url": "https:// example.com/orders/ dashboard" } } } } } } </oraInfoDisplay></pre> When a writable artifact is saved, the app sends the edited result back to the workflow as an <code>oraFormSubmit</code> block. The example below shows the exact payload shape the workflow receives, including the edited value and the widget's <code>metadata.artifactId</code>. <pre><oraFormSubmit id="qbr- follow-up"> {"newValue": "Updated text from the editor", "metadata": {"source": "qbr", "artifactId": "qbr- email-preview"}} </oraFormSubmit></pre> For <code>structuredRichText</code>, the same submit envelope is used, but <code>newValue</code> contains the updated <code>sections[]</code> array instead of a string. <pre><oraFormSubmit id="qbr- follow-up"> {"newValue": [{"name": "Greeting", "locked": false, Jordan, </p>"}], {"name": "Summary", "locked": false, "t summary from the</pre>

Type of Widget	Sample Prompt	Details	Example
			<pre>editor.</p>"}, {"name": "Closing", "locked": true, "text": "Account Team"}], "metadata": { "source": "qbr", "artifactId": "qbr-email-preview"}} </oraFormSubmit></pre> submitId: the artifact payload <code>content.id</code> when present; otherwise <code>artifactId</code> newValue: for <code>text</code> and <code>richText</code>, this is a string; for <code>structuredRichText</code>, this is the updated <code>sections[]</code> array with the latest <code>text</code> for each section metadata: the original metadata payload, passed back unchanged except that the widget adds <code>artifactId</code> under <code>metadata.artifactId</code>
Messages List	Use a <code>messageListWidget</code> to show a short list of recent messages, alerts, or notable events. Each item should have a clear title, optional supporting context like a subtitle or summary, and a time stamp when useful, so the list feels friendly, current, and easy to scan.	Use it for: alerts, activity feeds, and small collections of highlight items Required: <code>items[]</code> with at least <code>title</code> per item Item fields: <code>subtitle</code>, <code>summary</code>, <code>status</code>, <code>badgeText</code>, <code>priority</code>, <code>timestamp</code>, <code>image</code> Interactivity: item <code>action</code> and item <code>additionalAction</code> with <code>text + command</code> Runtime behavior is considered. <code>additionalAction</code> is only shown when the widget is in a focused panel. Also, the runtime supports success as a priority in addition to alert, warning, and medium.	<pre><oraInfoDisplay key="system-alerts"> { "patternId": "messageListWidget", "config": { "subtitle": "Real-time metrics", "items": [{ "title": "Memory Usage Spike Detected", "subtitle": "82% confidence", "summary": "Memory consumption increased by 40% in the last 5 minutes", "status": "Investigating root cause", "badgeText": "Medium", "priority": "medium", "timestamp": "3 min ago", "action": "ora.Invoke(\"openMemoryIncident \", { \"metric\": \"memory \" })", "additionalAction": { "text": "View Metrics", "command": "ora.Invoke(\"viewMetrics \", { \"metric\": \"memory \" })" } }], { {</pre>

Type of Widget	Sample Prompt	Details	Example
			<pre>"title": "Network Latency Increase", "summary": "Average response time increased from 120ms to 180ms.", "status": "Under investigation", "badgeText": "Warning", "priority": "warning", "timestamp": "16 min ago" }] } } </oraInfoDisplay></pre>
Change List	Use a changeListWidget to compare a few key metrics between a previous value and a current value. Add a short subtitle to explain what's being measured, keep the rows easy to read, and only include a message if there's a meaningful insight or anomaly worth calling out.	Use it for: current-versus-previous metric comparisons Required: <code>displayType</code>, <code>items</code> displayType: <code>percentage</code>, <code>raw</code>, or <code>currency</code> items[]: <code>title</code>, <code>currentValue</code>, <code>previousValue</code> Optional: <code>subtitle</code>, <code>messages</code> Values should stay numeric. Formatting comes from displayType, not from embedding percent signs or currency symbols inside the numbers.	<pre><oraInfoDisplay key="campaign-change-list"> { "patternId": "changeListWidget", "config": { "subtitle": "Campaign performance comparison", "displayType": "percentage", "items": [{ "title": "Email Open Rate", "currentValue": 28.2, "previousValue": 24.5 }, { "title": "Click-through Rate", "currentValue": 4.6, "previousValue": 3.8 }, { "title": "Conversion Rate", "currentValue": 1.9, "previousValue": 2.1 }], "messages": ["Conversion rate drop may indicate landing page issues"] } } </oraInfoDisplay></pre>
Chart	Use a chartWidget to visualize a simple trend, comparison, or proportion in a way that's easy to understand at a glance. Choose the chart type that fits the story, use clear labels, and include one or two short insights underneath to help explain what the chart is showing.	Use it for: trends, comparisons, or proportions Required: <code>type</code>, <code>data</code> type: <code>line</code>, <code>bar</code>, or <code>pie</code> data.labels[]: x-axis labels or categories data.datasets[]: each dataset needs <code>label</code> and numeric <code>data[]</code>	<pre><oraInfoDisplay key="sales-trend-chart"> { "patternId": "chartWidget", "config": { "type": "line", "data": { "labels": ["January", "February", "March", "April", "May"],</pre>

Type of Widget	Sample Prompt	Details	Example
		Optional: <code>insights[]</code>	<pre>"datasets": [{ "label": "Sales Revenue", "data": [10000, 25000, 15000, 40000, 30000] }], "insights": ["Revenue peaked in April", "Steady growth trend overall"] }] </oraInfoDisplay></pre>
Record	Use a <code>recordWidget</code> to show a structured record as a simple form that can either be reviewed or edited. Include clearly labeled fields, choose field types that match the data, and keep the layout straightforward so it feels comfortable for someone filling in or checking details.	Use it for: forms and read-only detail views Required: <code>id</code>, <code>fields[]</code> Optional: <code>readOnly</code> Field types: <code>text</code>, <code>textarea</code>, <code>number</code>, <code>date</code>, <code>select</code>, <code>system</code> Field fields: <code>id</code>, <code>type</code>, <code>value</code>, plus <code>label</code> and <code>options</code> when applicable In editable mode, the runtime automatically submits the form through an <code>ora.Agent(<oraFormSubmit . . . command. You don't configure a separate submit command in the widget itself.</code>	<pre><oraInfoDisplay key="employee-form"> { "patternId": "recordWidget", "config": { "id": "employee_form", "readOnly": false, "fields": [{ "id": "full_name", "type": "text", "label": "Full Name", "value": "John Doe" }, { "id": "employee_id", "type": "number", "label": "Employee ID", "value": 12345 }, { "id": "department", "type": "select", "label": "Department", "value": "engineering", "options": [{ "value": "engineering", "label": "Engineering" }, { "value": "sales", "label": "Sales" }] }, { "id": "session_context", "type": "system", "value": { "conversation_id": "abc123" } }] } } </oraInfoDisplay></pre>

Type of Widget	Sample Prompt	Details	Example
Multi Record	Use a multiRecordWidget to present structured information in a compact table with clear columns and rows. Keep the values short and scannable, use badge-style cells for statuses when helpful, and only add row actions if the experience should explicitly support taking action from the table.	Use it for: tabular record lists Required: <code>cols[]</code> and <code>rows[]</code> Column forms: plain string, or object with <code>label</code> and optional <code>showOnExpand</code> Row cells: plain string, or badge object with <code>type</code>, <code>text</code>, and <code>priority</code> Badge priority values: <code>alert</code>, <code>warning</code>, <code>medium</code>, <code>success</code>, <code>neutral</code> Optional row actions: action button and <code>drillDownAction</code> on the first column Optional: <code>subtitle</code> for table context and accessibility Optional multiselect fields: <code>id</code>, <code>selectMode</code>, <code>multiSelectActionText</code>, <code>multiSelectMetadata</code> The <code>showOnExpand</code> columns only appear when the widget is focused. This is useful for secondary columns that should stay hidden in compact mode. Badge priorities follow the same meaning as message list priorities: <code>alert</code> is red, <code>warning</code> is orange, <code>medium</code> is informational blue, <code>success</code> is green, and <code>neutral</code> is gray. To enable checkbox selection, set <code>selectMode</code> to <code>multiSelect</code>. This renders a checkbox for each row and a submit button below the table. The button label defaults to <code>Submit</code> unless you provide <code>multiSelectActionText</code>. <code>multiSelectMetadata</code> is an optional JSON object or value that's copied through unchanged into the submitted <code>oraFormSubmit</code> payload under the top-level <code>metadata</code> property. Use it to tell the receiving workflow what the selection is for, such as an action name, source context, routing hint, or any other extra data the workflow needs when processing the selected rows.	<pre> <oraInfoDisplay key="pending-approvals-table"> { "patternId": "multiRecordWidget", "config": { "subtitle": "Pending approvals", "cols": ["Request ID", "Type", "Status"], "rows": [{ "cells": ["REQ-101", "Budget", { "type": "badge", "text": "PENDING", "priority": "warning" }], "action": { "text": "Review", "command": "ora.Invoke(\"reviewRequest \", { \"id\": \"REQ-101\" })" }, "drillDownAction": "ora.Invoke(\"openRequest \", { \"id\": \"REQ-101\" })" }, { "cells": ["REQ-102", "Travel", { "type": "badge", "text": "PENDING", "priority": "warning" }], "action": { "text": "Review", "command": "ora.Invoke(\"reviewRequest \", { \"id\": \"REQ-102\" })" } }] }] } } </oraInfoDisplay> </pre> Example for multiselect table with checkbox submit. <pre> <oraInfoDisplay key="selected-orders"> { "patternId": "multiRecordWidget", "config": { </pre>

Type of Widget	Sample Prompt	Details	Example
			<pre>"id": "selected_orders", "subtitle": "Orders awaiting batch approval", "selectMode": "multiSelect", "multiSelectActionText": "Approve Selected", "multiSelectMetadata": { "action": "approve_orders", "source": "batch_review_queue", "approvalLevel": "manager" }, "cols": ["Order ID", "Customer", "Amount", "Status"], "rows": [{ "cells": ["ORD-5001", "Acme Corp", "\$12,500", { "type": "badge", "text": "PENDING", "priority": "warning" }] }, { "cells": ["ORD-5002", "Global Inc", "\$8,750", { "type": "badge", "text": "PENDING", "priority": "warning" }] }] } </oraInfoDisplay></pre> When the user submits selected rows in multiselect mode, the app sends an <code>oraFormSubmit</code> block back to the workflow using the widget's <code>id</code>. In the example below, <code>multiSelectMetadata</code> is copied directly into <code>metadata</code>, while the checked table rows are returned under <code>value[0].value</code>. <pre><oraFormSubmit id="selected_orders"> { "value": [{"id": "selectedRows", "value": [{"cells": ["ORD-5001", "Acme Corp", "\$12,500", { "type": "badge", "text": "PENDING", "p</pre>

Type of Widget	Sample Prompt	Details	Example
			<pre>] }, {"cells": ["ORD-5002", "Global Inc", "\$8,750", {"type": "badge", "text": "PENDING", "p }}]}], "metadata": {"action": "approve_orders", "source" </oraFormSubmit></pre>
Sankey	Use a sankeyWidget to show how volume or activity flows from one stage to another across a process. Define clear node names, keep the flow easy to follow from left to right, and use realistic values so the visualization tells a simple story about where things are going.	• Use it for: flows between stages or states • Required: nodes [], edges [] • nodes[]: id, name • edges[]: source, target, value Node IDs are numeric and edges point to those IDs. Keep values positive and keep the flow easy to follow.	<pre><oraInfoDisplay key="support-flow-sankey"> { "patternId": "sankeyWidget", "config": { "nodes": [{ "id": 0, "name": "Web Visitor" }, { "id": 1, "name": "Web Chat" }, { "id": 2, "name": "AI Chat Bot" }, { "id": 3, "name": "Resolved by Bot" }, { "id": 4, "name": "Transfer to Human" }, { "id": 5, "name": "Resolved by Human" }], "edges": [{ "source": 0, "target": 1, "value": 1000 }, { "source": 1, "target": 2, "value": 1000 }, { "source": 2, "target": 3, "value": 700 }, { "source": 2, "target": 4, "value": 300 }, { "source": 4, "target": 5, "value": 300 }] } } </oraInfoDisplay></pre>

Interactive widgets

For interactive widgets, use `ora.Invoke("actionCode", payload)`, which represents the app-defined action path.

Best Practices for Creating Workflow Nodes to Be Used in Agentic Applications

Here are some best practices related to creating nodes for workflows, to be used in agentic apps:

- Start by deciding which app stages your workflow should handle.
- Branch early on `oraMessageHint` so each stage stays simple.
- For `InitDisplay`, return display output and optionally include actions or `oraComms` when that improves the startup experience. For `InitCommunications`, return `oraComms`. For `InitActions`, return actions.
- Use `oraAppContext` and `oraUserContext` whenever the app experience depends on who the user is or what object the app is currently showing.
- If a node needs the current question or a derived prompt, pass it explicitly instead of assuming a downstream node will reconstruct it for you.
- Test the exact stage you're writing before including that workflow into the app.

Build Agentic Applications Using App Builder

You can create agentic apps from the Applications tab in AI Agent Studio. To build agentic apps, do these steps:

1. Go to **AI Agent Studio** and select the **Applications**  tab.
2. Select **Add** and enter the name, code, and description for your agentic app.
3. Select **Create**.
4. Enter the values in App Settings.

Add Workflows

Workflows are one of the primary components of your agentic app.

1. In the Application Builder, select **Add Section** and search for the workflow to include in your agentic app.

Note: Only the workflows that are enabled for using in agentic apps are available for selection. To use workflows in draft status, ensure that you've published the workflow atleast once. You can then add the draft workflow to your agentic app, and enable the **Use draft workflow while developing** option that's displayed on the right panel.

2. You can also assign custom agents for **Ask Oracle** and **Summary** sections.
If you're selecting a custom agent for Summary, enter the **Summary Prompt**.

Add Communications

Communications enable your agentic app to send emails and messages according to predefined templates.

Prerequisite

Define the templates for communication.

1.  In the App Builder, select  and add a document template.
2. Select the format for the template.
You can add templates of type PowerPoint, PDF, email, text, or podcast.

- PowerPoint: Add a PowerPoint template here. You can also define each slide separately. While creating the slide, specify the title for each slide or select **Allow agent to choose my title**.
- Email: Select **Allow agent to generate recipients**, **Allow agent to generate the subject** and add sections which can be filled by the agent when you select **Allow agent to fill in this section**. You can also include additional information in the email template.
- PDF: Select **Allow agent to generate the title** and **Allow agent to fill in this section** to add the title and content for the PDF template.
- Text: Select **Allow agent to fill in this section** to add content for the template of type Text.
- Podcasts: Add a podcast template and define one or more podcast sections. For each section, select a **Host**, provide **Data instructions** that describe the information to be discussed, and optionally enable **Add voice instructions** to configure the narration style, tone, or pacing. You can also include multiple hosts. The template provides a preview of the generated podcast content with audio playback. When creating a communication, select the podcast template and an applicable agent. At runtime, the selected agent generates the podcast script for each section using the configured template.

Add the communication

1. In the Application Builder, select **Add Communication** and add the title and description for the communication.
2. Select the agent the communication is applicable for.
 - a. Select the template that the agent must use when sending messages.
 - b. Enter the action to be performed in **Action Text**. For example, Send Email, Send Alert, Send to Agent.Optionally, you can also select an agent to send the communication to in **Target Agent**.

Add Actions

Create actions that are to be processed in the course of app flow, including requiring human approval for any step. Actions and commands work in a simple sequence.

1. A widget or another UI element triggers a command.
2. The command identifies which action to run.
3. The action starts.
4. The action runs its steps in order.
5. Those steps perform the real behavior.

Here, the command is the trigger, action is the specified workflow, and the steps are the ordered instructions within the workflow.

1. Select  and add a new action.
2. Enter the action code, display name, and description.
3. Select **Add Step** and choose the action.

Step Name	What It's For
Keep Action Available in UI	Keeps the original action available on UI even after it runs, instead of treating it as a one-time action.
Navigate to App	Opens another app. ○ Enter the code of the app to navigate to.

Step Name	What It's For
	○ Select Pass Payload to Context to send the payload. ○ Enter the context to send the payload to.
Send Agent Command	Sends follow-up work to an agent. Use this when the action should ask an agent to do something next. ○ Select Pass Payload to Context to send the payload. ○ Enter the command.
Refresh Agents	Refreshes one or more agent displays. Use this when the user should immediately see the updated content. Select the agent that's to be refreshed.
Switch App Context	Changes the current app context, optionally refreshing the app afterwards. ○ Select Use Payload as Context if needed. ○ Enter the context. ○ Select Reload App to refresh.
Edit Artifact	Opens the artifact editor sidebar. Use this when the action should let the user review or modify text, rich text, structured rich text, PDF, or URL artifacts.

Publish the Agentic Application

After adding the relevant artifacts and testing your app in the App Builder, publish your app. Users can view the published agentic applications from the Apps tab in the Agents and Applications page. To access this page, go to **Me > Quick Actions > Show More > AI Agents Studio > AI Agents**.

Note: Make sure that your users have access to interact with the AI agents. For information, see [How can I give users access to AI Agents?](#)

Run Agentic Applications from AI Agent Studio

You can run published agentic applications from AI Agent Studio.

1. Go to **AI Agent Studio**.
2. Select the **Applications**  tab.
3. Select **Run**  for a published agentic application.

Your agentic application displays the actions you can perform, priority communications, and any additional inputs defined in the agentic application configuration.

Here are some options available when you run an agentic application:

- Enable away notifications: After entering your query in **Ask Oracle**, select **Alert me when this completes** . If you switch to another tab while the query runs, a bell icon is displayed on your agentic application's browser tab when the response is complete.
- Switch display mode: The default mode is **Dark mode**. To change it, select .
- Edit settings: Select **Settings**  to configure the agentic application settings.

9 Edit Agents, Workflows and Agentic Applications

Edit a Workflow

You can edit workflows that are in draft status. You can also edit some preconfigured workflows.

Edit a Workflow in Draft Status

To edit a workflow in draft status, do these steps:

1. Go to **AI Agent Studio** and select the **Workflows**  tab. Then, search for and select the workflow you want to edit.
2. Select **Edit**  to modify a workflow.
3. Select **Settings**  to edit the workflow details such as the LLM, chat experience, questions, and security. For more information about the workflow settings, see [Create Workflows](#).
4. If needed, use **Debug**  to trace and test the workflow. For more information, see [Get Started with the Debugger](#).
5. Publish your workflow.

Edit a Preconfigured Workflow

Preconfigured workflows are identified by the **Seeded** tag. You can edit some of the preconfigured workflows.

To edit a preconfigured workflow, do these steps:

1. Go to **AI Agent Studio** and select the **Workflows**  tab. Then, search for and select the preconfigured workflow you want to edit.
2. Select **View** .
3. Select **Customize** to edit the preconfigured workflow. If the Customize option isn't displayed, it's not an editable workflow.
Note: In some preconfigured workflows, you can only edit the prompts in the LLM and Agent nodes.
4. If needed, use **Debug**  to trace and test the workflow. For more information, see [Get Started with the Debugger](#).
5. Publish your workflow.

Create a Copy of a Preconfigured Workflow

You can create copies of preconfigured workflows. The copy will have a different code and you can edit it further.

To create a copy of a preconfigured workflow, do these steps:

1. Go to **AI Agent Studio** and select the **Workflows**  tab. Then, search for and select the preconfigured workflow you want to copy.
2. Select **Duplicate** . Enter a new name for the workflow and edit it.
3. If needed, use **Debug**  to trace and test the workflow. For more information, see [Get Started with the Debugger](#).
4. Publish your workflow.

Edit a Supervisor Agent

You can edit previously created supervisor agents from the Resources tab in AI Agent Studio.

1. Go to **AI Agent Studio** and select the **Resources**  tab.
2. Select the **Supervisor Agents** subtab.
3. Select **Edit**  to modify the supervisor agent.

You can make changes in the new page that's displayed for editing the supervisor agent.

Edit and Test Supervisor Agents Iteratively in Playground

You can use **Edit in Playground** to edit and test your supervisor agents and large language model (LLM) nodes. This option enables iterative, real-time testing and refinement of supervisor agent instructions and parameters, without deploying changes to the production environment. It bridges the gap between high-level intent and the step-by-step process, so you can safely test, adjust, and fine-tune your agent's behavior before deploying it to production.

To edit and test supervisor agents:

1. Go to **AI Agent Studio** and select the **Marketplace**  tab.
2. Select the **Supervisor Agents** subtab.

Note: You can also access the Supervisor Agents subtab from the Resources  tab.

3. Search for the supervisor agent and then select **Edit**  to modify it.
4. From the editing area, select **Edit in Playground** to edit and test your supervisor agents.
5. Edit the agent details to adjust the prompt logic and model parameters. To edit and view real-time results in a dual-pane layout, you can select **View Results** .

Prompts Tab

You can design templates and instructions that guide the supervisor agent's behavior.

Note: The system prompt and summarization prompts are specific to supervisor type of agents.

Field	Description
System Prompt	Edit the system prompt to define your agent's identity, job description, and rules.
Summarization Prompt	Select the summarization mode, and edit the summarization prompts to include only essential instructions relevant to your specific use case.

Note: You can add expressions to fields using **Insert Expression**  . For more information, see [Expressions](#).

LLM Tab

Choose the type of model and change the custom model properties if needed. You can either use the default model or choose a model.

Field	Description
Provider	Select the default model or a custom model. For custom models, define the model properties.

Input Tab

Enter the specific test data, variables, or context required for a supervisor agent or LLM node to run. You can simulate real-world triggers and verify that your agent processes incoming information correctly before saving your changes.

Field	Description
User Input	Enter your queries or actionable commands that initiates the supervisor agent's logic.
Evaluation	Use evaluation sets to assess your supervisor agent's performance. An evaluation set contains one or more test questions, the expected agent responses, and the metrics to be measured. For more information, see Evaluate Agents .
Variables	Configure variables, making them accessible to all nodes within the workflow.

Field	Description
Additional Variables	Add additional variables to the prompt. For example, to add the current system date to your prompt, select the Current Date and Time option using Insert Expression  .

Output Tab

Configure the overall structure of the supervisor agent's output using JSON schema, to specify the exact output.

Field	Description
Specification Mode	Select this mode to directly modify the JSON schema for the output.
Simple Mode	Select this mode to define the output values and types. The corresponding JSON schema will be generated automatically and displayed in the specification mode for any further changes.

6. Select **Save and Test** to verify the changes you made and test the supervisor agent or the LLM node. While the test is running, you can monitor the results and performance in these sections:
 - Input: Review the messages you entered in the Input Message field on the Input tab. To view the JSON schema, select **JSON**. To view the final response, select **Human**.
 - Output: Select a session to open the detailed trace view. The trace shows a step-by-step timeline of the run, including which tools were called, how long each step took, and metrics for each step. Select the color-coded trace lines to view details, such as latency and input or output token usage. To view the final response, select **Human**.
7. Select **Run History**  to view, compare, and reuse previous test runs. You can view a detailed record of every run and track your supervisor agent's performance.
 - Select **Edit**  to add a comment in the change log.
 - Select **Apply**  to load a proven configuration into the panel and resume iterative testing from a high-quality baseline. This action only affects the testing environment and won't update your production supervisor agent until deployed.
 - Use **Expand**  to view change log details of the test.
 - You can compare any two runs by selecting them. Select **Compare** to view a pane displaying the details of each run and highlighting their differences.
 - Select **Save and Test** to save your changes to the supervisor agent, and run the agent with the test configuration.
8. Select **Done**.
9. After testing your changes, publish your supervisor agent.

Edit an Agentic Application

You can edit agentic applications that are in draft status.

1. Go to **AI Agent Studio** and select the **Applications**  tab.
2. Search for the agentic application in draft status and select **Edit** .
3. Edit and publish the agentic application.

Create a Copy of an Agentic Application

You can create copies of agentic applications that are in published or draft status.

1. Go to **AI Agent Studio** and select the **Applications**  tab.
2. Search for the agentic application and select **Duplicate** .
3. Enter a new name and code for the agentic application and select **Create**.
4. Edit and publish the agentic application.

10 Debug and Test AI Agents

Overview of the Debugger

The debugger in AI Agent Studio helps you to trace, test, and troubleshoot agentic flows in real time. It provides visibility into node execution, context, inputs, outputs, and LLM reasoning, helping you understand the runtime behavior of the agentic flow. Before you publish your workflow or agentic app, you can use the debugger to validate its behavior. It shows how a request flows through the agent, what inputs are passed to each node, what outputs are returned, and how the final response is generated. This visibility helps you isolate defects and reduce trial and error.

Here are some of the tasks you can do using the debugger:

- Shorten build cycles: Rerun the agentic flow from a specific node instead of replaying the entire flow.
- Improve quality: Catch incorrect routing, malformed payloads, poor tool responses, and weak summaries during design time itself.
- Increase reliability: Repeat common scenarios using saved run profiles.
- Lower operational risk: Use step-through debugging and output inspection to understand why an agent made a particular decision.
- Accelerate root-cause analysis: Trace an incorrect response back to the node where the issue originated.
- Improved collaboration: Help product managers, developers, and support teams collaborate by reviewing and discussing node inputs and outputs.

This table provides information about some of the debugging actions supported in AI Agent Studio.

Debugging Action	When to Use	What You Can Inspect or Verify
Run workflow	Run a full test path for the workflow.	Validate the complete workflow.
Run saved profile	Reuse saved test inputs.	Run regression tests for common scenarios.
Inspect nodes	View runtime data for a node.	Inspect inputs, outputs, variables, and context.
Breakpoint	Pause before or at a specific node.	Inspect state of each node before continuing the debug session.
Step through nodes	Run one node at a time.	Verify that each node passes the correct output to the next node.
Resume debugging	Continue the debug session.	Resume workflow run after inspection.
Rerun From Node	Retest the workflow from a specific node.	Retest the selected node and downstream logic.
Pin output	Reuse stable output data.	Verify how cached output affects downstream processing.
Edit output	Test downstream processing with modified output data.	Verify how downstream systems consume the modified payload.

Debugging Action	When to Use	What You Can Inspect or Verify
Override node configuration	Test the changes to prompts or configurations.	Verify the effect of temporary changes before applying them permanently.

Here's a screenshot showing the three main working areas in the debugger.

2

Test and Chat pane
Chat with the application, test inputs, and preview responses in real time.

1

Workflow canvas
Visualize and edit the workflow run. Add, connect, and configure nodes to build the application logic.

Weather Activity Suggestion App Workflow
App-compatible weather activity workflow that begins with weather preferences, interprets OraMessageHint intent, gathers hint-specific weather context, and uses LLM nodes to render concise app responses.

2

user preferences

Hello FUSION FUSION! Today is 07/16/2026, and the current time is 7:46 AM. The weather information is illustrative and not available for your location. Here are three activities you might enjoy regardless of the weather:

1. Reading a new book or exploring an interesting podcast.
2. Trying out a new recipe or cooking a favorite meal.
3. Doing a home workout or some light stretching to start your day.

Let me know if you'd like suggestions tailored to a specific weather or location!

Ask me anything

1

Get User Preferences
Business Object Function

Success

Determine App Intent
Code

Success

Route Ora Hint
Switch

Success

Query

Fetch Query Weather Data
LLM

default

Fetch Init Weather Data
LLM

Callout Number	Name	Description
1	Workflow canvas	This canvas displays the workflow nodes and the connections between them. Depending on the workflow, it can include start and end

ORACLE

110

Callout Number	Name	Description
		nodes, LLM or agent nodes, Decision nodes, Function or Tool nodes, Loop nodes, Human Approval nodes, and Return nodes. During a workflow run, the canvas shows the runtime status of each node. Completed nodes display a success status, while the active or stopped node is highlighted. Branch labels indicate the path followed after a Switch or If Condition node.
2	Test and Chat pane	In debugger, you can use Ask Oracle to submit a test prompt and review the response. Use realistic prompts that represent the path you want to test, such as a request that must trigger a list, detail, policy, update, approval, or lookup flow. Use the conversation to view debugger status messages that indicate where the run is paused and which node currently has control.
3	Node Details pane	Selecting a node displays the node details that include the Results and Configuration tabs. - The Results tab is the primary debugging view, that displays the runtime data for the selected node. It provides the following information: - Duration: How long the node took to run. - Output: The value produced by the node. - Input: The payload passed to the node. - Variables: Variables available at that step. - Context: Runtime context, prior state, user input, and run metadata. - Token metrics: When available for the node type. - The Configuration tab enables you to verify the node's setup, such as the LLM prompt, model configuration, bound function, tool configuration, parameter mappings, or routing logic.

Get Started with the Debugger

Use the following sequence for efficient debugging:

1. Start with a saved run profile when possible.
2. Run the workflow end-to-end once.

3. Identify the first visible symptom.
4. Trace backward from the symptom through node outputs and inputs.
5. Set a breakpoint before the suspected failing node.
6. Pin stable upstream outputs to avoid unnecessary reruns.
7. Step through the suspected node and its downstream handoff.
8. Use output simulation to test edge cases.
9. Use configuration override to test a fix.
10. Accept the configuration change only after validation.
11. Remove temporary breakpoints, pins, mocks, and overrides.
12. Rerun the full workflow.
13. Retest the affected run profiles.

Test and Debug a Workflow

Follow these steps to validate the workflow path from the user prompt to the final response.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug**  to view the conversation pane with **Ask Oracle**.
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Watch the highlighted run path on the canvas.
6. Select completed nodes to inspect their results.
7. Review the final response in the conversation panel.

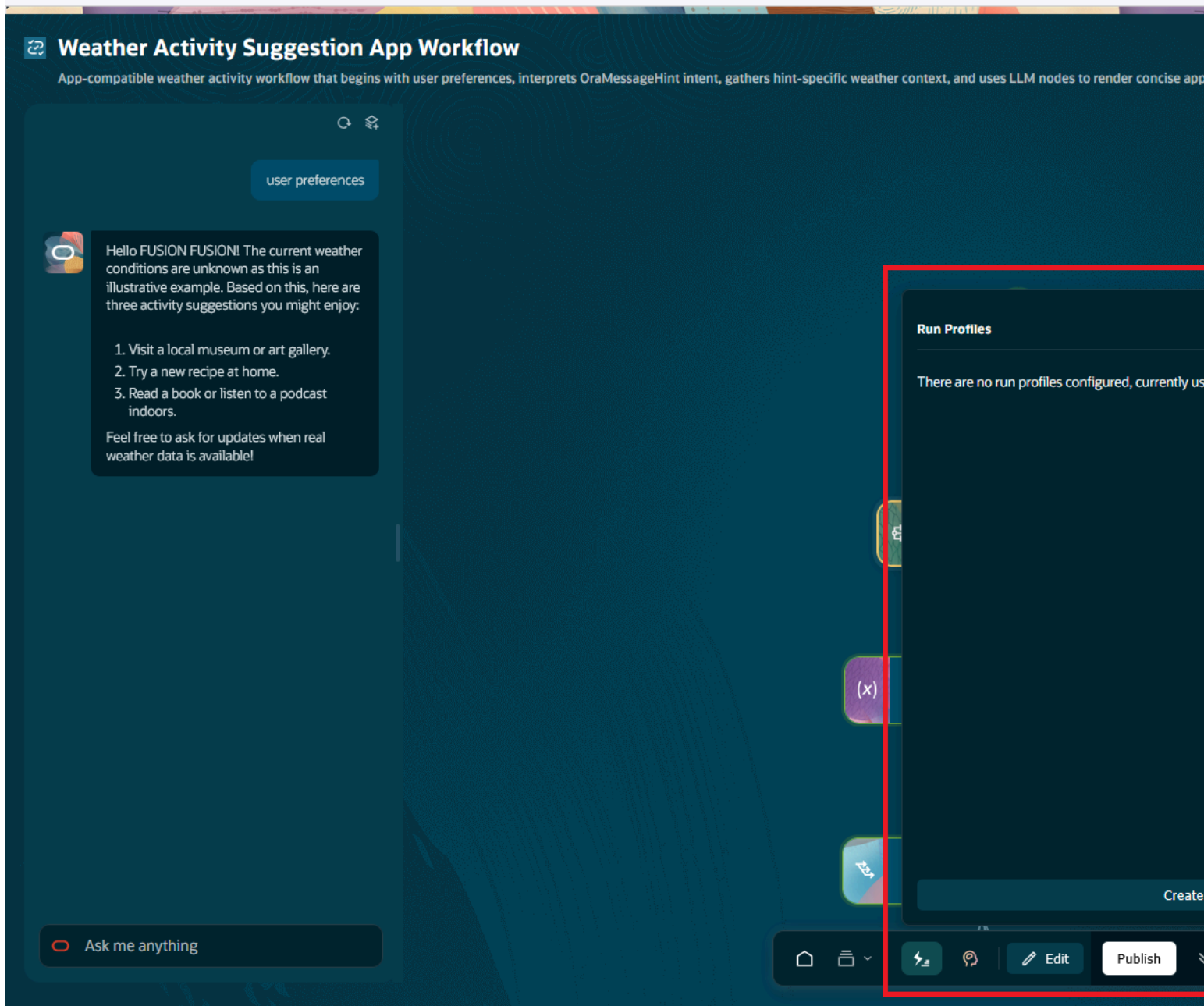
For example, if a prompt is expected to follow a specific business path, the workflow must extract or receive the request, route it through the correct decision node, call the required data or tool nodes, transform or summarize the result, and display the expected response in the conversation pane.

If the final response is incorrect, don't start by changing the final prompt. Instead, trace the data backward through the nodes to identify where the error was introduced.

Create and Use Run Profiles

Use run profiles to save and replay common test inputs without retyping prompts, ensuring they cover typical business scenarios, workflow branch paths, and known edge cases.

Here's a screenshot that displays the Run Profiles option.



Create a Run Profile

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** , and then select **Run Profiles** .
4. Select **Create new profile**.
5. Select **Edit**  to enter a name for the profile.

Note: Enter a profile name that describes the business scenario, not just the prompt text. For example, use **Policy question with attachment** or **Specific record search** instead of **Test 1**.

6. Select the type and provide the other profile details.

The run profile is saved and you can run the test case when needed.

Run an Existing Profile

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** , and then select **Run Profiles** .
4. Select the profile that matches the scenario you want to test.
5. From the toolbar, select **Run** .
6. Confirm the expected route on the canvas.
7. Inspect the key nodes on that route. For more information, see [Inspect Node Results](#).
8. Confirm the final chat response.

Recommended Profile Coverage

Create run profiles to validate each workflow path and ensure comprehensive test coverage for the following scenarios:

- Each switch branch
- Each major user intent
- Positive and negative test cases
- Known bug scenarios
- Edge cases, such as missing attribute values, invalid attribute values, empty results, or policy lookup failures

Inspect Node Results

Ensure to inspect each node by selecting a completed or paused node and reviewing its runtime data using the following order.

1. Output: Confirm what the node produced.
2. Input: Confirm what the node received.
3. Variables: Confirm the local values used during execution.
4. Context: Confirm what state and prior outputs were available.
5. Configuration: Confirm that the node is wired and configured correctly.

If the node uses expressions such as `{{context.$nodes.NODE_CODE.$output}}`, validate the expression before interpreting the result. Confirm that there aren't any spaces inside the braces, the referenced node code exists, the referenced node ran before the selected node, and the property path matches the actual output shape.

LLM, Agent, and Multi Agent Nodes

For LLM, agent, and multi-agent nodes, inspect the following to verify that the model is receiving the correct inputs and producing valid outputs:

- The user prompt, the generated prompt, or the task instruction
- System instructions, if available
- The input context passed to the model
- The model output
- Token usage or duration metrics, when available
- Whether the output is valid for downstream use
- Whether chat history is enabled when the node depends on prior conversation

Here are some of the common issues encountered while debugging LLM, agent, and multi-agent nodes:

- Malformed JSON
- Missing fields
- Inconsistent intent labels
- Over-summarized output
- Output that doesn't match the downstream node's expected schema
- Model response that ignores the required business format

Decision Nodes: Switch and If Condition

For Switch and If Condition nodes, inspect the following to verify that the correct branch is selected based on the input data:

- The expression or condition being evaluated
- The actual input value used at runtime
- The selected branch or outcome
- The branch label shown on the canvas
- Whether the downstream path matches the user's request

Data, Tool, and Service Nodes

When creating run profiles for Business Object Function, External REST tool, Tool, Vector DB Reader, Vector DB Writer, RAG, Email, and Workflow nodes, make sure you cover the following scenarios:

- The selected object, tool, endpoint, index, workflow, or service configuration
- The input payload and parameter mappings
- Query attributes, filters, operators, or request body values
- The returned object structure, status, and error fields
- Empty, partial, or unexpectedly large result sets
- Whether downstream nodes expect the same data structure that the service returned

For retrieval nodes, also verify that the retrieved content is relevant to the user's request and that downstream summarization uses the retrieved content correctly. For writer or update nodes, confirm the operation, target identifier, and payload before allowing the workflow to continue.

Code, Set Variables, and Return Nodes

Verify that each node is configured correctly and produces the expected output.

- For Code nodes, inspect the input context, source code configuration, return type, and output structure. Confirm that the returned value matches the node's configured return type and any downstream schema expectations.
- For Set Variables nodes, inspect each variable's name, scope, type, and value. Confirm that downstream nodes read the intended variable from the correct scope.
- For Return nodes, inspect the return value and confirm that it's the intended workflow response payload. Use a Return node only when the workflow intentionally emits a specific structured result because not every workflow requires one.

Loop, Wait, Human Approval, and Parallel Nodes

- For Loop and While Loop nodes, inspect the collection or condition, each iteration's inputs and outputs, and whether the nested nodes receive the expected item or loop state.
- For Wait nodes, inspect the wait duration, loop-back target, message, and maximum iteration settings.
- For Human Approval nodes, inspect the channel type, approval or feedback settings, message template, actions, timeout behavior, and loop-back behavior.
- For Parallel nodes, inspect each branch outcome separately. Confirm that all required branches ran and that downstream nodes can handle both combined and branch-specific outputs.

Start, End, and Comment Nodes

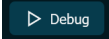
- Start and End nodes are structural nodes. Use them to confirm where the workflow begins and where each execution path ends.
- Comment nodes are diagram-only annotations. They help explain the workflow canvas but don't produce runtime inputs or outputs.

Debug Switch Routing

Switch nodes are decision checkpoints. A switch evaluates a runtime value and routes execution to the configured branch. Use branch names that clearly map to the workflow's supported intents or scenarios.

Confirm that routing conditions are evaluated correctly and requests are directed to the appropriate branch using these steps:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.

3. From the toolbar, select **Debug** .
4. Select the upstream intent-extraction node.
5. Inspect the extracted intent or subscription value.
6. Select the Switch node.
7. Select **Results**.
8. Inspect the switch input and the selected output branch.
9. Follow the highlighted branch on the canvas.
10. Inspect the first downstream node on that branch.

Verify routing decisions and identify potential issues that could cause requests to follow an incorrect branch using this checklist:

- The user prompt interpretation
- The upstream LLM prompt
- The switch condition
- The branch label or expected value
- The downstream node's assumptions about routing values

Set and Use Breakpoints

Breakpoints pause the workflow at specific nodes so you can inspect the node's state before continuing. They are useful because they let you stop the workflow before it moves too far downstream. This makes it easier to isolate the exact state at a decision point.

Set a Breakpoint

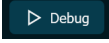
Here are the steps to set a breakpoint in the workflow.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Locate the node where you want the run to pause.
6. Hover near the breakpoint control on the node  and select it to activate the breakpoint. Confirm that the breakpoint indicator turns red , indicating active breakpoint state.

We recommend setting breakpoints immediately before nodes that are expensive, error-prone, or important for data handoff. For example:

- Before a function, service, or tool call
- Before a RAG Document Tool node
- Before a final summary node
- After an intent-extraction node
- Before a node that recently changed

Run to a Breakpoint

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow. The workflow runs and when the run reaches the breakpoint, the workflow pauses.
5. Read the chat or debug status message to confirm the stopped node.
6. Select the paused node and inspect the results from the **Results** tab.

Breakpoints are useful because they let you stop the workflow before it moves too far downstream. This makes it easier to isolate the exact state at a decision point.

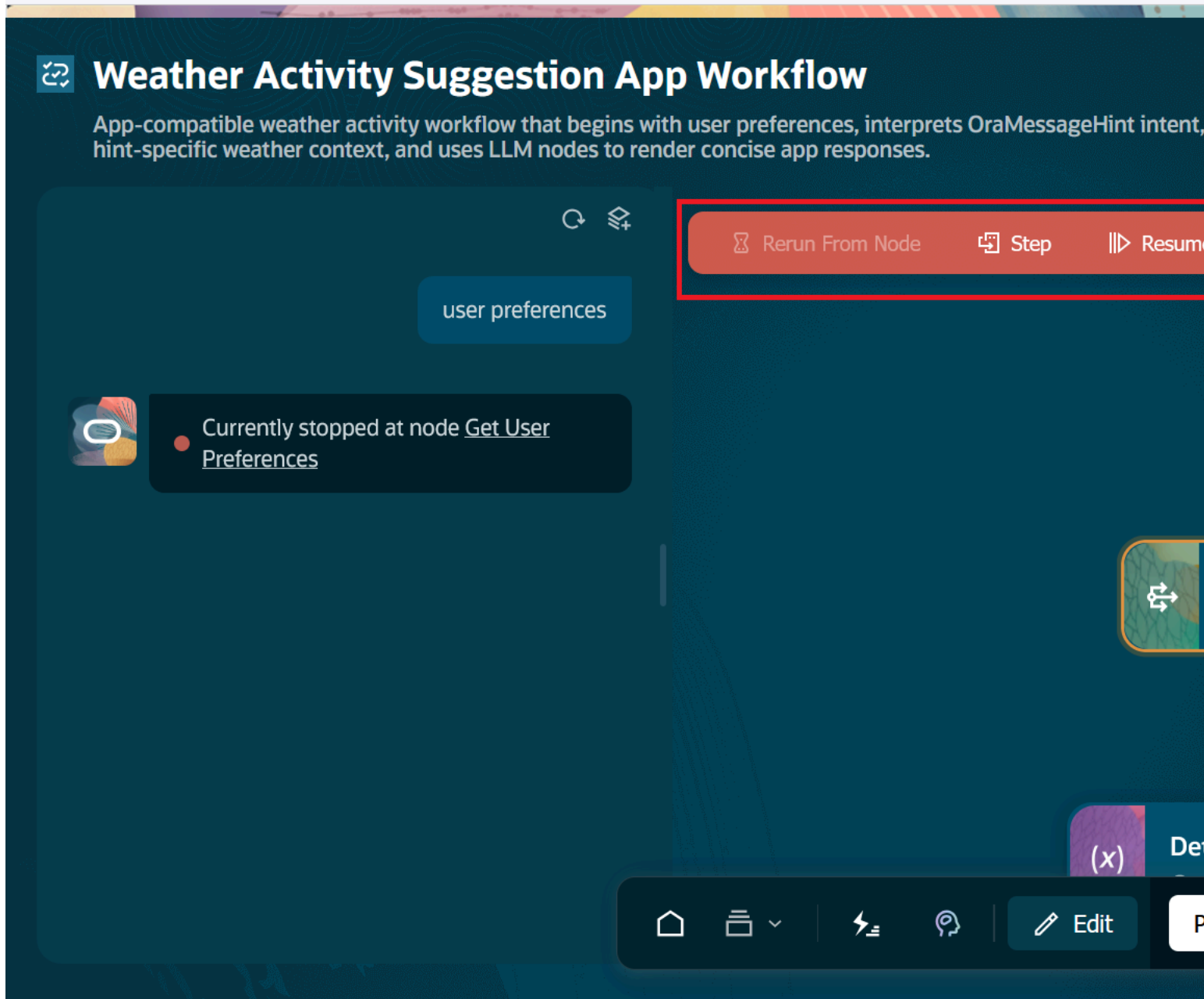
Remove a Breakpoint

Select the active breakpoint control  to disable it. Remove any temporary breakpoints before final validation so the workflow can run end-to-end.

Rerun, Step Through, or Resume a Workflow

When the run pauses, a red control bar appears near the top of the canvas. Here's a screenshot that displays the Rerun From Node, Step, and Resume options.

- Rerun From Node: Restart the run from the currently selected node.
- Step: Run the next node and pause again.
- Resume: Continue the run until the next breakpoint or the end of the workflow.



Rerun From Node

Select **Rerun From Node**  from the control bar, when the upstream work is already correct and you want to retest the current node and downstream nodes. Here's a list of typical uses:

- Retest a node after changing its configuration.
- Replay a data, tool, or service call without repeating earlier intent extraction.
- Retest the final summarization after confirming the retrieved data.
- Continue debugging from a paused node using the current context.

Before using Rerun From Node, inspect the selected node's input. If the input is incorrect, rerunning from that node will reproduce the same problem. In that case, go upstream and fix the node that produced the incorrect input.

Step Through a Workflow

Use **Step**  when you need to inspect every handoff. This pattern is useful between a payload-generation node and a data, tool, or service node, where the upstream output must match the downstream node's expected input shape.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Let the run pause at a node.
6. Select the last node that was run.
7. Inspect Input, Output, and Context.
8. Select **Step** .
9. Inspect the next node.
10. Repeat until you understand the handoff.
11. Select **Resume**  after completing the step-by-step inspection.

Resume Run

Select **Resume**  from the control bar, when the current state looks correct and you want the workflow to continue. The workflow runs until it reaches the next breakpoint or it completes.

Pin or Unpin a Node Output

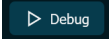
Pinning freezes a node's output so the debugger can reuse the same result in subsequent runs. Pinning is useful when debugging a final response node. If the retrieval or service step has already returned the correct data, pin its output and focus on improving downstream behavior. This saves time and avoids repeatedly running the same upstream node.

Use pinning in the following scenarios to reduce unnecessary execution and simplify workflow testing:

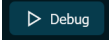
- A stable upstream node doesn't need to run again.
- A node consumes tokens or has a high execution cost.
- A node calls an external service, data function, or tool.
- You want to isolate downstream behavior.
- You want to reproduce a scenario using the same input every time.

Use pinned outputs to speed up debugging by reusing previously generated results.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.

3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Run the workflow until the node produces a valid result.
6. Select the node.
7. Select the **Results** tab.
8.  Select **Pin output** to pin an action, and the pin indicator is displayed on the node.

After pinning, subsequent debug runs can use the cached output instead of running the node again. Remove a pinned output when you want the node to run again and generate fresh results.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Run the workflow until the node produces a valid result.
6. Select the node.
7. Select the **Results** tab.
8.  Select **Overriding Output** to unpin an action.
9. Run the workflow again to confirm that the node completes its run.

Edit Node Output

Editing a node's output lets you test downstream behavior with controlled data.

Pinning outputs enables you to do these things:

- Force a specific intent
- Inject a specific business record
- Test an empty result set
- Test malformed or missing fields
- Verify downstream error handling
- Confirm that a summary node handles a specific data shape

To test downstream behavior with custom data, edit a node's output before continuing execution.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Select the node whose output you want to control.
6. Open the node's results panel.

7. Select **Overriding Output** .
8. Modify the text or JSON structure.
9. Continue with Step, Rerun From Node, Resume or start a new run.
10. Inspect downstream nodes to confirm that they consumed the modified output correctly.

Here are the structure items you can verify:

- Required keys
- Expected data types
- Arrays and objects
- String quoting
- Field names expected by downstream nodes

Override Node Configuration

A configuration override enables you test a temporary node change during debugging without immediately committing it to the workflow.

Here's a list of common causes of payload or execution issues:

- A missing or incorrect prompt instruction
- A malformed parameter mapping
- An incorrect query field
- The wrong operator
- An incomplete variable reference
- A prompt that doesn't produce the expected JSON shape

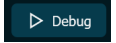
Use a configuration override to test temporary changes without modifying the workflow definition.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Select the node you want to adjust.
6. Select the **Configuration** tab.
7. Select **Override**.
8. Edit the relevant configuration, such as prompt text, variables, tool settings, or parameter logic.
9. Run, step through, or rerun the workflow to test the change.
10. Verify that a temporary override indicator, such as an orange alert icon, appears on the node.
11. If the override fixes the issue, select **Accept** to apply it to the workflow. If it doesn't fix the issue, select discard or reset the override.

Treat overrides as temporary until they have been validated. A good practice is to test the override using at least two run profiles before accepting it.

Validate Final Chat Output

After the workflow completes, inspect both the final response and the node-level data that produced it. Validate the final response against the source data and the expected response format for the workflow's use case.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Read the final chat response, and then select the final summary node.
6. Inspect its input includes the expected source data.
7. Inspect its output matches the final chat response.
8. If the final response is incorrect, trace backward to the previous node, and continue tracing until you find the first incorrect input or output.

This process helps prevent unnecessary changes to the final response prompt when the real issue is upstream routing, retrieval, or payload generation.

Checklist for Publishing Workflow After Debugging

Before publishing the workflow changes, ensure to complete the following checklist:

- Test each major run profile.
- Exercise each switch branch.
- Remove or disable breakpoints used during debugging.
- Unpin pinned outputs unless intentionally retained for a debugging session.
- Clear temporary mock outputs.
- Accept or discard temporary configuration overrides.
- Check that LLM nodes produce outputs in the expected format.
- Check that Data, function, tool, and service nodes receive valid payloads.
- Check that RAG and tool nodes receive relevant inputs and return useful content.
- Check that final summary nodes receive the expected source data.
- Check that final chat responses are accurate, readable, and aligned with the user's request.
- Test error and empty result scenarios where relevant.

FAQs for Debugging and Testing AI Agents

Why is the workflow selecting the wrong branch?

A mismatch between the intent extracted by the upstream LLM node and the switch condition used for routing can cause this issue. If the workflow is taking the wrong path, follow these steps to identify whether the issue is caused by incorrect intent extraction or a switch condition mismatch.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the upstream LLM node that extracts the intent.
6. Confirm the extracted value matches the user's request.
7. Inspect the switch input and confirm the selected branch.
8. Adjust the extraction prompt or switch condition.
9. Retest with multiple run profiles.

How do I troubleshoot a data or function node that returns unexpected data?

A Data or Function node can produce unexpected data because of a malformed input payload, incorrect filter conditions, missing or incorrect attributes, service-side issues, or problems with the underlying source data.

Use the following steps to determine whether the issue originates from the input payload, node configuration, connected service, or source data:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the Function node input to verify that it contains the expected payload.
6. Confirm that all attribute names, values, and operators are correct and match the function or service requirements.
7. Inspect the Function node output to determine whether the unexpected data originates from the function itself.
8. If the input is incorrect, trace the workflow upstream and inspect the payload-generation node to identify where the data became invalid.
9. If the input is correct but the output is incorrect, investigate the selected function, tool, API endpoint, service configuration, or the underlying source data for errors or inconsistencies.

By validating the input, output, and upstream data flow, you can isolate whether the issue originates in the workflow, the function, the connected service, or the source data.

Why is the LLM generating malformed JSON?

A common cause is that the prompt doesn't clearly constrain the expected output format, allowing the model to produce responses that don't match the required JSON schema. To diagnose and resolve the issue, follow these steps:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the output generated by the LLM node.
6. Compare the generated output with the schema expected by the downstream node.
7. If applicable, update the LLM node's output specification.
8. Add or refine output-format instructions using a configuration override (for example, explicitly specify the required JSON structure).

Note: Clearly define the expected JSON structure in your prompt, including all required fields, their data types, and any formatting requirements. Providing a precise and well-defined output specification helps the LLM generate valid and consistent JSON more reliably.

9. Step into the downstream node.
10. Verify that the downstream node successfully accepts the corrected JSON output.

How can I fix an incorrect final summary?

First, inspect the input to the final summary node and verify that the source records are correct. Next, review the final summary output. If the source data is accurate, improve the summarization prompt. If the source data is incorrect, trace the issue back to the retrieval or routing step to identify the root cause.

How do I troubleshoot a workflow that stops at an unexpected node?

A workflow that stops unexpectedly is often caused by an active breakpoint, a rerun state, or the current debug control state. Read the chat or debug status message, check for active breakpoint indicators, select the highlighted node, and review the pause controls. Then use resume, step, or remove the breakpoint as appropriate to continue the workflow.

How can I identify the cause of inconsistent downstream behavior during workflow debugging?

Downstream behavior is often difficult to reproduce because upstream outputs change between runs. To make debugging consistent, create or select a run profile, run the workflow until the upstream data is correct, pin the stable

upstream node output, and mock or edit the output if you need to test a specific edge case. Then step through the downstream nodes to observe and troubleshoot the behavior consistently.

How do I fix an expression or variable that resolves to the wrong value?

An expression or variable may resolve to the wrong value because of invalid expression syntax, an incorrect node code, a missing upstream dependency, or an incorrect output property path.

To troubleshoot the issue, follow these steps:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Select the node you want to adjust.
6. Go to the Configuration tab.
7. Inspect the node configuration for expressions, such as `{{ $context.$nodes.NODE_CODE.$output }}`.
8. Confirm there aren't any spaces inside the expression braces.
9. Verify that the referenced node code exists and is upstream of the current node.
10. Inspect the output of the referenced node to ensure it contains the expected data.
11. Compare the expression path with the actual output structure to verify the property path is correct.
12. Correct the expression or rewire the workflow so the referenced node runs before the current node.

What should I do if the output from one node doesn't match the input expected by the next node?

A mismatch between the data returned by the upstream node and the format expected by the downstream node usually causes this behavior. For example, the data may be returned as a string, array, object, or different field structure.

Use the following steps to troubleshoot the problem:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the upstream node output to understand the data being produced.
6. Inspect the downstream node input to see what data it expects.
7. Compare the required fields, data types, and nesting between the two nodes.
8. Use output editing to test the expected data shape without changing the upstream node.
9. Adjust the upstream prompt, code, return value, or field mapping so the output matches the downstream node's expected format.
10. Retest the workflow using the affected run profile to verify the issue has been resolved.

What should I do if pinned or edited output is hiding the real problem during workflow debugging?

Pinned or edited output can hide the real issue because the debugger reuses controlled output instead of executing the live node.

To troubleshoot the problem, follow these steps:

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the workflow canvas for pin or output override indicators.
6. Inspect the selected node's results to confirm whether the output is pinned or edited.
7. Clear any pins or output overrides when you want to test the live run path.
8. Rerun the full workflow so the node runs with live data.
9. Compare the live output with the previously pinned or mocked output to identify the actual source of the issue.

How do I fix a workflow that works with a configuration override but still fails in a full run?

The override may only fix one path, while another run profile, branch, or upstream payload continues to produce invalid input.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Confirm which node has the temporary override.
6. Retest the same path with the override enabled.
7. Run at least one adjacent profile or branch that depends on the same node.
8. Inspect the input differences between the passing and failing runs.
9. Accept the override only after the affected profiles pass.
10. Discard the override if it's hiding a broader upstream issue.

How do I troubleshoot a loop or parallel branch that produces incomplete results?

Incomplete results from a loop or parallel branch are typically caused by an empty collection, an incorrect loop condition, a branch that was never reached, an iteration output mismatch, or a downstream aggregation issue.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the loop collection, while condition, or parallel branch outcomes.
6. Confirm that each expected branch or iteration ran successfully.
7. Inspect the outputs of the nested nodes within each branch or iteration.
8. Verify that downstream nodes receive the expected combined or per-branch data.
9. Test the workflow with empty, single-item, and multi-item inputs to ensure it behaves correctly in all scenarios.

How do I fix an external service, tool, or retrieval node that fails?

Missing configuration, invalid parameters, service side errors, empty retrieval results, or mismatched authentication or context can all cause this failure. Use these steps to verify the configuration and isolate the cause of the failure.

1. Go to **AI Agent Studio** and open the **Workflows**  tab.
2. Select **Edit**  to modify a workflow.
3. From the toolbar, select **Debug** .
4. Enter a prompt in **Ask Oracle** and submit it to run the workflow.
5. Inspect the node configuration and confirm the selected tool, endpoint, index, or workflow is correct.
6. Inspect the runtime input payload to verify the node is receiving the expected data.
7. Review the returned status, error, and metadata fields for clues about the failure.
8. If the input is wrong, trace the issue back to the payload-generation or mapping node.
9. If the input is correct but the result is still wrong, investigate the configured service, data source, or selected tool.
10. Pin a known good response only when you're isolating downstream behavior.

11 Monitor and Evaluate AI Agents

Monitor and Evaluate AI Agents

Monitor and gain insights into how your AI agents are performing, and also evaluate the agents for accuracy. You can also track the interactions with your agents, understand real-world usage patterns, identify common errors, and measure overall performance.

- **Monitoring:** Monitoring tracks performance and provides insight on how your agents behave in production. Monitor agents to ensure that your quality bars for response time and token counts are maintained over time. You can also see any errors logged here.
- **Evaluation:** Evaluate agents before you deploy them, to ensure that they're ready for production. Test your agents for response correctness, response time, and token usage to meet your quality standards. You can also check the quality of answers generated through the document tool to assess how effectively agents utilize the retrieved context from the retrieval-augmented generation (RAG) metrics. After making any changes to your agent, or after a model update, rerun evaluations to confirm that your agent continues to perform as expected. This proactive approach helps you maintain high-quality experiences for your users.



Note: The AI Agent Studio interface has been updated since this video was recorded. In the current UI, open the Monitoring and Evaluation page by selecting the **Monitoring and Evaluation**  tab from the left navigation pane of AI Agent Studio. The page opens in a new browser tab, where you can continue following the steps shown in this video.

This table summarizes some key metrics, their descriptions, and their availability for monitoring or evaluation.

Metric	Description	Available to Evaluate	Available to Monitor
Error Rate	Percentage of user sessions that ended in an error.	Yes	Yes
Error Count	Total number of errors recorded.	Yes	Yes
Session Count	Total number of conversations initiated with agents.	Yes	Yes
P99 Latency	The maximum wait time in milliseconds for 99% of users, revealing any areas where you should review and optimize the prompts or structure of the agent.	Yes	Yes

Metric	Description	Available to Evaluate	Available to Monitor
P50 Latency	The maximum wait time in milliseconds for 50% of users, helping identify performance issues. You can view this metric in the details of the monitoring or evaluation results.	Yes	Yes
Total Tokens	Cumulative number of tokens used by all agents.	Yes	Yes
AI Units	Measure of value of the AI services provided by Oracle. It's a formula applied on the tokens consumed.	Yes	Yes
Median Correctness	The 50th percentile of correctness scores across evaluation runs. Each score (0–1) is computed by comparing the agent's answer to the reference answer provided in the evaluation set.	Yes	No
Session Count	The number of unique conversational sessions between a user and an AI agent. One session can include multiple messages or evaluation runs.	Yes	Yes
Groundedness	Alignment of the generated answer with the retrieved source content, indicating how faithfully the response reflects the supporting information.	Yes	No
Answer Relevance	Degree to which the answer directly addresses the user's question, measuring how fully and precisely the content covers the required subject matter.	Yes	No
Context Relevance	Quality and appropriateness of the retrieved information, assessing whether it meets the necessary standards to be considered reliable evidence.	Yes	No

Prerequisites

Aggregate the metrics that are displayed on the Monitoring and Evaluation tab in AI Agent Studio.

1. Go to **Navigator > Tools > Scheduled Processes**.
2. In Scheduled Processes, click **Schedule New Process**.
3. Leave the type as **Job**.
4. Search for and select **Aggregate AI Agent Usage and Metrics**.
5. Run the **Aggregate AI Agent Usage and Metrics** scheduled process.

You can schedule this process to run on a recurring basis, for example, once a day.

The process aggregates the metrics that are displayed in the Monitoring and Evaluation tab of AI Agent Studio.

Monitor Agents

1. Go to **Navigator > Tools > AI Agent Studio**.
2. Select **Monitoring and Evaluation**  tab, and then select the **Monitoring** subtab.
 - The Monitoring subtab displays aggregated metrics of all the agent runs over the selected time frame.
 - All agent runs are monitored, including agents in draft status.
3. Select the agent to view additional details.

Each row represents a single session and displays the number of turns (back-and-forth messages), the session's completion status (successful or error), and the number of tokens used.
4. Select any session in the list to open the detailed trace view. This view displays a step-by-step timeline of the entire conversation, showing exactly which tools were called, the duration of each step, and the metrics for each step.

Evaluate Agents

Use evaluation sets to assess your agents' performance. An evaluation set contains one or more test questions, the expected agent responses, and the metrics to be measured. Evaluation sets are specific to each agent, and an agent can have multiple evaluation sets. For more information, see [Best Practices for Creating Evaluation Sets](#).

1. Create an evaluation set for an agent.
 - a. Go to **AI Agent Studio**.
 - b. Select the **Monitoring and Evaluation**  tab.

All evaluations run on the agents are displayed in the Evaluation subtab.
 - c.

To create an evaluation set, click **Manage Evaluations** and select .
 - d. Enter a name, code, and description for the evaluation set, and select the agent team to be evaluated.
 - e. Choose the run mode.

Sequential: Runs the questions in the exact order you define them. Use this if one question depends on the context of the previous one.

Random: Runs the questions in a random order.

- f. Select **Enable document tool evaluation** if you want to view RAG metrics in the evaluation report.
- g. From the Questions subtab, add common questions that users are likely to ask the agent and the answers you would like the agent to deliver. Ensure both the questions and answers are concise, user-friendly, and reflect best practices.

You can either add your questions and the expected answers individually, or upload a CSV file with the questions in the first column and the expected answers in the second column.

- h. From the **Metrics** subtab, edit each metric to set the pass and fail criteria. For example, to indicate that the test fails if the correctness score is less than 0.7, choose **<** as the threshold condition and enter 0.7 as the threshold value.
 - i. Select **Create** to save the evaluation set.
2. Run the evaluation set.
 - a. On the Manage Evaluations page, select the **Initiate Evaluation Run** action for your evaluation set.
 - b. Choose the version of the agent to evaluate and run the evaluation.
 3. Analyze the results.
 - a. Click the evaluation set to view the Evaluation Runs page.
 - b. Select the evaluation run and select the **View Run Results** action.

Subtab	Information Displayed
Response performance	- Comparison of the expected response versus the actual response from the agent for each question, along with the metrics for each question in the evaluation. - Trace provides information about the detailed timeline for each question in the evaluation.
Correctness	Detailed breakdown of the correctness score. The LLM provides an initial score and feedback, and you can add your own feedback for record keeping in the Correctness Score by Human column.

Compare Evaluation Runs

You can see a side-by-side comparison of two different runs of the same evaluation, and easily spot regressions or improvements in latency, correctness, and token usage. Doing so, you can understand how an agent's performance changes over time, especially after you've made modifications.

1. From the Evaluation subtab, select the evaluation.
2. Select any two runs and then select **Compare**.
 - o The Summary subtab displays a high-level overview of the performance differences between the runs.
 - o The Details subtab provides a granular, question-by-question breakdown of the runs. For each question in the evaluation set, you can directly compare the actual response from Run 1 against the actual

response from Run 2. You can also compare the specific latency, tokens used, and trace links for each question, making it easy to pinpoint exactly where and why performance or accuracy has changed.

View the Measured Value of Agents

Use the **Value** dashboard to understand the business impact of your workflows. This visual dashboard provides a consolidated view of your AI fleet, performance by Oracle Fusion Cloud Applications products, and metrics for individual workflows. These metrics help you estimate and visualize the tangible ROI of your AI initiatives. You can track deployment, usage, and estimated time and cost savings for your published workflows in one view.

Configure Performance Metrics for a Workflow

To configure performance metrics for a specific workflow:

1. Go to **AI Agent Studio**.
2. Select **Workflows**  tab.
3. Search for the workflow to edit, and select **Edit** .
4. Select **Settings**  to edit the workflow details.
5. On the Details tab, scroll to the **Capture estimated Cost and Time savings** section.
6. Configure the estimated values for ROI tracking metrics by setting **Time saved each agent run** and **Cost saved each agent run**. If you don't enter any values, the default values apply.
7. Publish your workflow.

You can now view the measured value of the workflow using the Value subtab of the Monitoring and Evaluation tab.

Monitor AI Agent ROI and Key Efficiency Indicators

1. Go to **AI Agent Studio**.
2. From the Monitoring and Evaluation tab, select the Value subtab. You'll see a dashboard that displays real-time ROI per agent and team, allowing you to filter by Family or Product to monitor key metrics such as **Total Time Savings**, **Total Cost Savings**, **Total Usage**, and **Total Agents**.
3. Select any item on the dashboard to see the total time saved (hours) and total cost saved (USD) across the workflow.
4. You can also configure the estimated ROI metric values for a workflow. Select **Workflows**, then search for the workflow and select **Edit**  to update the settings. Select **Apply**, and then save your changes.

Best Practices for Creating Evaluation Sets

Creating a good evaluation set helps in improving the efficiency of your evaluation. Here are some best practices for creating evaluation data sets and testing protocols for agent teams.

The foundation of a successful evaluation is a high-quality data set. This data set consists of paired questions or inputs, and expected responses. The expected response must be factually correct and grounded in the source context.

Example Data Set

Question	Expected Response
Is aromatherapy covered?	No, aromatherapy isn't covered. According to the provided context, aromatherapy is listed under "Alternative Treatments" which aren't covered by United Healthcare Medical Plans.
Do you pay for thermometers?	Based on the provided context, thermometers aren't covered. The document "MEDICAL SUPPLIES AND APPLIANCES" lists thermometers as excluded supplies.
Is laser surgery for eyes covered?	Based on the provided context, laser surgery for eyes isn't covered. The "VISION" section states that surgery to correct nearsightedness, including laser surgery, is listed under plan exclusions.

General Evaluation Guidelines

These principles apply to all agent teams, to ensure comprehensive testing.

- **Safety and grounding:** Validate that the agent adheres to safety policies. It must avoid hallucinations, that is, if no relevant information is found, it should state that, rather than guessing.
- **Ambiguity and multipart queries:** Evaluate the agent's ability to handle vague queries, implicit reasoning, or single turns containing multiple distinct questions.
- **Negative testing:** Introduce out-of-scope or unanswerable questions, for example, topics not covered by the available tools, to ensure the agent responds with an appropriate "I don't know" or redirect.
- **Multiturn dialogue:** If the agent is conversational, design tests that require maintaining context and history across several turns, for example, answering a follow-up question based on the previous answer.
- **Latency and performance:** Measure response time and efficiency, particularly in scenarios involving calls to multiple tools or complex reasoning.

Evaluation Guidelines for Workflows

Evaluations for workflows must test the overall logic flow and the robustness of individual nodes.

Workflow Structure and Logic

- **Path coverage:** If the workflow has multiple paths or branches, ensure the evaluation tests all paths.
- **Scenario depth:** Test multiple distinct scenarios for the same path to ensure consistency.
- **Unsupported scenarios:** Include tests that identify scenarios the workflow doesn't support, to verify graceful failure or error handling.

Workflow Nodes

A workflow includes multiple node types, and the evaluation must cover the key scenarios applicable to each type.

Node	Scenario
LLM	Test the LLM prompt to identify if it can handle all types of questions and formatting instructions.
Code	• Robustness: Include questions that test the stability of the code. • Edge Cases: Test for scenarios that cover different edge cases to ensure the code doesn't break the flow.

Node	Scenario
Document Tool (RAG)	Questions designed for Retrieval-Augmented Generation (RAG) must test the agent's ability to handle complexity, not just simple keyword retrieval. - Long Range Context: Test if the agent can resolve dependencies scattered across distant sections or several pages of a document. - Distributed Context: Ensure the agent can aggregate information from multiple noncontiguous parts of the document to answer comprehensively. - Concealed Context: Test the ability to find and extract specific, obscure details deep within the text. - Reasoning: Check if the agent can apply reasoning or logic to the retrieved information to provide a correct answer. - Table-Sourced: Test the ability to interpret and pull accurate data from tables within the document.
Business Object	- Function Coverage: Ensure evaluations test all business functions available in the business objects. - Parameter Variation: Test the same business function from different angles and with different parameters. For example, if a BO creates an object, test it with various input types.
REST	- Endpoint Coverage: Ensure evaluations test all functions available in the REST tool. - Scenario Variation: Test the same REST tool for different scenarios and parameters, for example, handling different payload structures or update types.
Deep Link	Validation: Confirm that deep links are generated correctly and for the appropriate scenarios.
Model Context Protocol (MCP)	- Tool selection: Validate that the correct MCP tool is called. - Function accuracy: Validate that the correct functions are called within the MCP tool.
Document Processor	Format Handling: Test the node for various attachment types (PDF, txt, and so on) to ensure consistent text extraction.
Vector DB Reader	Retrieval Accuracy: Test if the node retrieves the most semantically relevant chunks based on the input query.

For the other supported nodes, make sure evaluation tests cover other nodes including Tools, Vector DB Reader, and Vector DB Writer.

12 Promote Published Agents

Migrate Agentic Applications or Workflows from One Instance to Another

You can migrate an individual application or a workflow from one environment to another using the import and export options available in AI Agent Studio. For example, you can export from the test or staging environment and import into the production environment.

Migrate an Agentic Application

To migrate an agentic application:

1. From the source environment, go to **AI Agent Studio** and open the **Applications**  tab.
2. Locate the application you want to export and select **Export** . A JSON file is downloaded to your local system. You can edit it to enter the export details.
3. From the target environment, go to **AI Agent Studio** and open the **Applications** tab.
4. Select **Import** and upload the previously downloaded JSON file. Your application is imported successfully.
5. Publish the application.

Migrate a Workflow

To migrate a workflow:

1. From the source environment, go to **AI Agent Studio** and open the **Workflows**  tab.
2. Locate the workflow you want to export and select **Export** . A ZIP file is downloaded to your local system. You can extract it locally and edit it to enter the export details.
3. From the target environment, go to **AI Agent Studio** and open the **Workflows**  tab.
4. Select **Import** and upload the previously downloaded ZIP file. Your workflow is imported successfully.
5. Publish the workflow.

The imported application or workflow will be available for use in the new environment only after you publish it in that environment. Also, if any of your agents use the following tools, make sure you complete the corresponding post-import tasks for each tool:

- Document tool: Upload and republish the document to ensure its content is correctly processed in the new environment.
- Email tool: Reconfigure the email tool to make sure necessary alerts are generated properly in the new environment.
- External REST tool: Reconfigure authentication settings for external REST API tools.

Migrate a Supervisor Agent from One Instance to Another

You can migrate an individual supervisor agent from one environment to another using the import and export options available in AI Agent Studio. For example, you can export from the test or staging environment and import into the production environment.

To migrate a supervisor agent:

1. From the source environment, go to **AI Agent Studio** and open the **Marketplace**  tab.
2. Go to the **Supervisor Agent** tab and locate the supervisor agent you want to export.
3.  Select **Export** to enter the export details and then select the **Export** button. A JSON file is downloaded to your local system. You can edit it to enter the export details.
4. From the target environment, go to **AI Agent Studio** and open the **Marketplace**  tab.
5. Go to the **Supervisor Agent** tab and select **Import** to upload the previously downloaded JSON file.
6. Publish the supervisor agent.

The imported supervisor agent will be available for use in the new environment only after you publish it in that environment. Also, if any of your supervisor agents use the following tools, make sure you complete the corresponding post-import tasks for each tool:

- Document tool: Upload and republish the document to ensure its content is correctly processed in the new environment.
- Email tool: Reconfigure the email tool to make sure necessary alerts are generated properly in the new environment.
- External REST tool: Reconfigure authentication settings for external REST API tools.

13 Use AI Studio Command Line Interface (CLI)

Overview of AI Studio CLI

You can use the AI Studio Command Line Interface (CLI) to scaffold projects, fetch server artifacts on request, create and modify local artifact files, validate files, run workflow and application tests, inspect debugger sidecar state, and save or publish to remote services when explicitly requested. Use the CLI directly when the next step is known, the command is mechanical, and you can supply the required options confidently. Use a code assistant such as Codex or Claude Code with the CLI when the work requires planning, interpretation, multifile changes, workflow or application design, validation repair, or careful sequencing. The content in this chapter is mainly about using code assistant with AI Studio CLI.

Use Case	Recommendation
User creating or redesigning a workflow or application from natural language, a screenshot, or a product specification	Code Assistant + AI Studio CLI
User adding a workflow node, application panel, action, communication, template, or test where the exact implementation isn't obvious	Code Assistant + AI Studio CLI
User fixing validation or test failures across related artifacts	Code Assistant + AI Studio CLI
User who wants visual editing plus source files	VS Code extension
User running a known create, inspect, validate, test, fetch, or save command	AI Studio CLI directly
User making a narrow local edit with known flags	AI Studio CLI directly

Direct CLI is recommended for these options:

- Checking version, identity, or command help.
- Validating a known file.
- Listing or fetching a known server object when you explicitly want server state.
- Running a focused workflow or app test suite.
- Saving or publishing a reviewed local artifact when the release path says to do so.

A Code assistant is recommended for these options:

- Deciding which artifact family and command sequence to use.
- Creating a workflow or application from a business goal.
- Choosing and wiring workflow nodes, applications, agents, widgets, actions, templates, and communications.
- Coordinating related application-backed workflow stages such as `InitDisplay`, `Query`, `InvokeAction`, `FillParameters`, Or `SendCommunication`.
- Handling validation errors, test sync, local judge attachments, and report summaries.

We recommend using the CLI from the AI Studio project root without assuming a globally installed `aistudio` executable. Run the bundled script by using this path:

```
node .agents/skills/aistudio/scripts/aistudio.js <command> [options]
```

Skill for AI Studio CLI

`aistudio` is a skill that you can run on the CLI. If the skill is installed outside the project, run that installed script while keeping the working directory at the project root:

```
node "C:\path\to\aistudio\scripts\aistudio.js" <command> [options]
```

Note: For readability, this topic uses `aistudio <command>` in examples. Consider `aistudio` as shorthand for the bundled `node ... \scripts\aistudio.js` command.

Commands in AI Studio CLI

Use these commands to view all available options in AI Studio CLI:

```
aistudio --help  
aistudio <command> --help
```

Prerequisites for Using AI Studio CLI

1. If you're using Codex to access CLI, you must have already procured access to OpenAI/Anthropic Codex through a subscription. The Codex API keys aren't included as part of Oracle product.

Note: Usage of Codex doesn't count toward any Oracle AI units, as it's a separate subscription.

2. If you're using Visual Studio Code, make sure that you've installed NodeJS and Visual Studio Code.

Set Up AI Studio CLI

Here are the steps to set up AI Studio CLI:

- *Enable CLI in AI Agent Studio*
- *Configure the VS Code Project Folder*

Enable CLI in AI Agent Studio

1. Go to **AI Agent Studio** and open the **Credentials**  tab.
2. Open the **AI Studio CLI** subtab.
3. Select **Enable AI Studio CLI**.
4. If your administrator has created the public client in Oracle IDCS, add the **Application ID**.
5. If your administrator hasn't created the public client in Oracle IDCS, contact your IDCS administrator to follow the steps in the **Create a public client in Oracle IDCS** section and then add the **Application ID**.
6. Save the configuration.

Configure the VS Code Project Folder

1. After your administrator has setup the public client in Oracle IDCS, copy the lines in the **env.properties** section from the AI Studio CLI tab, part of Credentials tab in AI Agent Studio.

The lines will be similar to these lines:

```
aistudio.fa-host = https://fa-eqkj-dev18-saasfademo1.ds-fa.oraclepdemos.com
aistudio.clientId=7e4190889ab144539f494768bd49f6fc
aistudio.orchestratorTokenUrl=https://fa-eqkj-dev18-saasfademo1.ds-fa.oraclepdemos.com/api/fusion-ai/
orchestrator/authz/v1/tokens
aistudio.primaryScope=urn:opc:resource:fusion:eqkj-dev18:fusion-ai/
aistudio.redirectUri=http://127.0.0.1:43188/callback
aistudio.tenant=https://idcs-2ccbf0b4d0654430ac37d210f6efd9a8.identity.oraclecloud.com:443
```

2. Paste the lines to the **env.properties** file in the project root folder of your VS Code project folder. If you don't see the file in your root project folder, create the file and paste the values.
3. In VS Code, run the **Show and Run Command** action and select **Fusion AI Studio: Configure Authentication**
 - a. Select OAuth / Token Authentication.
 - b. Sign in using your credentials.
4. You can also open a terminal and run the command `aistudio authenticate`.

Here are some common `auth` commands in `aistudio`:

- `aistudio authenticate`
- `aistudio whoami`

- `aistudio logout`

Create Layout for New Project

Use `init` when creating a blank project:

- `aistudio init`

Artifacts are available within these root folders:

Artifact	Default location
Workflow	<code>src/workflows/*.wf</code>
App	<code>src/apps/*.app</code>
Agent	<code>src/agents/*.agent</code>
Business Object	<code>src/bos/*.bo</code>
Deeplink	<code>src/deeplinks/*.dl</code>
Topic	<code>src/topics/*.topic</code>
Tool	<code>src/tools/*.tool</code>
Connector Definition	<code>src/connectorDefinitions/*.connectorDefinition</code>
Connector Instance	<code>src/connectorInstances/*.connectorInstance</code>
Approval Process	<code>src/approvals/*.approval</code>
Policy Store	<code>src/policies/*.policy</code>
Policy Template	<code>src/policyTemplates/*.policyTemplate</code>
Document Schema	<code>src/documentSchemas/*.documentSchema</code>
Function Template	<code>src/functions/*.function</code>

Development Flow in AI Studio CLI

For most artifact work, do these steps:

1. Inspect the project layout and the relevant files.
2. Run `aistudio --help` and `aistudio <command> --help` for commands you haven't used in the current context.
3. Don't hand edit the JSON files.
4. Create or modify local artifacts.

5. Run the matching `validate-*` command.
6. For workflows, run `do-prettyfy-workflow` after structural edits, then `validate-workflow`.
7. For material workflow edits, run the workflow test sync loop.
8. For material app edits with a top-level agent container, run the app test sync loop.
9. Save, publish, unpublish, or delete remote state only when explicitly requested.

Code Assistant Usage and Examples

When using a code assistant, ask it to inspect local context first and to use the CLI for supported artifact operations using a prompt as shown below:

```
Use the AI Studio CLI from the project root.
Inspect the project layout, command help, and relevant local artifact files before changing anything.
Use CLI create/modify/update commands instead of hand-editing JSON when a command supports the operation.
Create or modify artifacts locally only.
Validate all changed artifacts.
For material workflow changes, run the workflow test sync loop.
For material app changes with testable panels, run the app test sync loop.
Do not run do-save-*, do-publish-*, remote delete, or server fetch commands unless I explicitly ask.
Summarize what changed, what was validated, and any remaining blockers.
```

If you're using the installed `aistudio` skill, ask the code assistant to load the relevant skill references for the artifact type rather than calling unavailable prompt-listing commands. For example, workflow work should use the workflow references, application work should use the app references, and test work should use the workflow or application test authoring references. This section gives some example prompts that you can use in code assistants.

Build a Workflow With a Code Assistant

Scenario: Create a workflow for a salesperson to manage their opportunities.

Note: You can use the LLM to generate the prompt for you.

Here's the prompt that was generated by a code assistant.

```
Build a standalone AI Studio workflow that manages Opportunities based on a free-text question the customer provides when the workflow runs.
```

```
## 1. What the workflow must do
```

```
The customer supplies a single input message/question at run time. Read it from `$context.$system.$inputMessage` (treat that as the question). The workflow must:
```

1. **Classify intent** from the question. Supported intents (Opportunity management only):
 - ``LIST`` - show **all** of the user's opportunities, with no filter criteria (e.g. "show all opportunities").
 - ``FILTERED_LIST`` - show a **filtered subset** of the user's opportunities based on criteria expressed in the question (e.g. "show opportunities with a win probability of 70 or higher", "show opportunities with revenue > 100000"). The criteria can reference any queryable opportunity field with a comparison (`>=`, `>`, `<=`, `<`, `=`, `between`, `contains`, etc.) and may combine multiple conditions.
 - ``UPDATE`` - update an existing opportunity. This includes updating any updateable field(s) of the opportunity **and/or** creating opportunity **revenue** records against it.
 - ``CREATE`` - create a new opportunity, optionally **with** revenue records.
 - ``OUT_OF_SCOPE`` - anything that is not opportunity management.
2. **Guard scope.** If the intent is ``OUT_OF_SCOPE``, do not attempt any operation. Return a friendly message that states the purpose of the workflow: it only helps the user view, create, and update their Sales

Opportunities (including opportunity revenue records), and invites them to rephrase. Do not call any business object on this branch.

3. ****Perform the operation**** matching the detected intent using the Opportunity business object function(s).
4. ****Return a user-friendly response.**** The final user-facing text must be natural-language, confirm what was done (e.g. fields updated, opportunity created with its key/ID, revenue lines added), and for `'LIST' / 'FILTERED_LIST'` summarize the opportunities readably (for `'FILTERED_LIST'`, restate the filter that was applied and how many matched). Ground the response in real upstream node output – do not fabricate IDs, fields, or results.

2. Business Object identification (do this before wiring operation nodes)

We need to ****find, create, and update opportunities****, plus create ****opportunity revenue**** records. Reuse before you build:

1. Use `'search-business-objects'` to find an existing Opportunity business object and inspect its functions (query/list, create, update, and revenue-record creation). Prefer an existing reusable BO.
2. ****Verify it meets the requirement**** – it must expose functions to: query the user's opportunities, update updateable opportunity fields, create an opportunity, and create opportunity revenue records (a child/revenue function or a create that accepts revenue lines).
3. ****If a suitable BO exists, use it.**** Carry the selected `'search-business-objects'` result forward as `'businessObjectHint'` on every later BO tool call (`'get-business-object-functions'`, `'do-create-node'`, `'do-modify-node'`, etc.).
4. ****If no existing BO meets the need****, create a new Business Object source artifact (`'.bo'`) for Opportunities (and its revenue records) per the `'business-object-builder.md'` reference, then use it.

Do not guess `'businessObjectCode'` or `'functionName'`; select the real BO and the real functions. Let `'do-create-node' / 'do-modify-node'` resolve and persist each `'BO_FUNCTION'` node's `'outputSpecification'` (pass `'businessObjectHint'`, omit `'outputSpecification'`), then reference only fields that actually exist in the resolved schema.

3. Suggested topology (adapt to the real BO functions)

START

```
-> CLASSIFY_INTENT (LLM) // outputs intent token + extracted filter criteria as JSON
-> SWITCH on the classified intent
case LIST -> LIST_OPPORTUNITIES (BO_FUNCTION: query, no filter) -> FORMAT_LIST (LLM) -> END
case FILTERED_LIST -> BUILD_FILTER (CODE) -> QUERY_OPPORTUNITIES (BO_FUNCTION: query with filter) ->
FORMAT_FILTERED (LLM) -> END
case CREATE -> CREATE_OPPORTUNITY (BO_FUNCTION: create)
-> [if revenue requested] CREATE_REVENUE (BO_FUNCTION) -> FORMAT_CREATE (LLM) -> END
case UPDATE -> [resolve target opportunity]
-> UPDATE_OPPORTUNITY (BO_FUNCTION: update updateable fields)
-> [if revenue requested] CREATE_REVENUE (BO_FUNCTION) -> FORMAT_UPDATE (LLM) -> END
case OUT_OF_SCOPE -> RETURN (static purpose message)
```

Guidance for wiring:

- Use a ****SWITCH**** node whose `'caseExpression'` reads the intent produced by `'CLASSIFY_INTENT'`. Give each case its own dedicated path; do not collapse multiple intents into one terminal prompt.
- The `'CLASSIFY_INTENT'` LLM prompt must inject the question explicitly (`'{{context.$system.$inputMessage}}'`). It must output a single intent token (`'LIST' | 'FILTERED_LIST' | 'UPDATE' | 'CREATE' | 'OUT_OF_SCOPE'`) plus, for `'FILTERED_LIST'`, a structured list of the filter criteria the user expressed – each as field + operator + value (e.g. `'[{"field":"WinProbability","op": ">=", "value": 70}]'`). Give the node an `'outputSpecification'` that declares both the intent token and the criteria array so the SWITCH and the filter-builder can read them. Have the SWITCH match on the intent token field only.
- ****'FILTERED_LIST' path:**** the `'BUILD_FILTER'` CODE node converts the extracted criteria into whatever the chosen query function actually expects (e.g. an `'RSQL' / 'q'` filter string, a `'where' / 'finder'` parameter, or named bind parameters – match the real BO function signature). Inspect the query function's token parameters via `'get-business-object-functions'` and build the filter to fit. Then bind `'QUERY_OPPORTUNITIES'` filter input to `'{{context.$nodes.BUILD_FILTER.$output.result.<filterField>}}'`. If the chosen query function cannot accept server-side filters, fetch the user's opportunities and apply the criteria inside `'BUILD_FILTER'` (or a post-query CODE node) instead – but prefer server-side filtering when supported. Only reference opportunity field names that exist in the BO function's resolved `'outputSpecification' / 'parameter set'`; do not invent field names like `'WinProbability'` or `'Revenue'` without confirming the real ones.
- For `'UPDATE'` and revenue creation, extract the needed parameters (target opportunity identifier, the fields to change, revenue line details) from the question. If the BO update/create functions need a

```
`personId`/`assignmentId` or the current user's identity, insert the standard upstream identity BO function
(`Logged In User Assignment Info` / `fetch_loggedIn_user_assignmentId`) and bind those IDs before the
operation node.
- Add **CONDITION guards** before any node that depends on a first record (e.g. `...$output.items[0].Id`)
and route empty-data cases to a graceful no-data message instead of calling a downstream BO with missing
inputs.
- Only run the revenue-creation node when the question actually asks for revenue records (branch it behind
an IF/CONDITION); otherwise skip straight to the formatting node.
- Each formatting LLM node must inject the relevant upstream BO output expression (`{{$context.
$nodes.<NODE>.$output...}}`) and produce concise, friendly, accurate confirmation/summary text.
```

4. Constraints & reminders

```
- Expressions are `{{$...}}` with no inner spaces. No template block tags (`{{#if}}`, `{{#each}}`, etc.) -
use control-flow nodes for conditional behavior.
- `CODE` node business fields are read under `$output.result...`.
- Normalize `workflowCode` to uppercase with underscores (e.g. `OPPORTUNITY_MANAGEMENT`). Pick the workflow
family/product from the real LOV options for the Sales/Opportunity domain; if there is no exact match, ask
before guessing.
- Keep every node fully wired into the execution path in the same pass - no floating nodes.
```

5. Finish criteria (do not stop early)

After the structural batch:

```
1. Run `do-prettify-workflow --file <workflow-file>`.
2. Run `validate-workflow --file <workflow-file>` and fix anything it reports.
3. Since this is a newly created workflow, continue into the workflow **test sync** loop: start with `get-
workflow-test-sync-plan --file <workflow-file>` (per `workflow-test-authoring.md`) and run the tests. Do not
use remote judging unless explicitly asked.
4. Report the final suite/package summary, preserving the `Metrics:` and `Optimization:` lines.
```

If anything about the Opportunity BO functions or their parameters is genuinely ambiguous after searching,
ask one targeted question rather than inventing schema.

Instructions given to the code assistant

Here are the instructions given to a code assistant to generate the above prompt. You can review the output and confirm that the prompt has instructions to what you want as the output before it generates all the artifacts.

Use the /aistudio skill to create a prompt that I can use in a code assistant to achieve the below:

Use Case

```
* Create a workflow that can do the following based on the question asked by the customer.
* See all the users opportunities
* Update an opportunity
* The user may want to update any updateable fields of the opportunity
* The user may want to create opportunity revenue records
* Create an opportunity
* Create an opportunity with revenue records
* The user would provide an input message or a question when running the workflow.
* Use the question to figure out what the customer wants to do. We should only support management for
opportunities. If the question is for something other area then respond back with what the purpose of the
workflow is.
* Once you identify what the customer wants to do (intent of the user), perform the appropriate operation
and return the output to the user in a user friendly way.
```

Business Object Identification

For this workflow, we need to find, create and update opportunities. You need business objects for that.

```
* Find existing business object to query opportunities, update opportunities and create opportunity
* Make sure that business object meets our requirement
* If it meets the requirements, use the business object. If it does not meet the need, create a new business
object
```

The openAPI describe for the opportunity object has been added to the project folder.

Enhance the Workflow to Support a New Scenario

Scenario: `src/workflows/opportunity_management.wf` already exists, and you want to enhance it to add a team member with questions such as Add Elaine Benes to the opportunity 'Laptops for ABC Networks'.

Use the AI Studio skill at:
<AI Studio CLI location>

Enhance the standalone workflow:
`src/workflows/opportunity_management.wf`

Add support for adding an existing Fusion Resource as a team member to an existing opportunity.

Inspect:

- `src/workflows/opportunity_management.wf`
- `src/businessObjects/opportunities.bo`
- `resourecs-describe.json`
- `opportunities-describe.json`

Important runtime rules:

- A team member is an existing Fusion Resource. Never create a Resource record.
- Check if there are existing BOs for searching for Resources. If not, create a new BO.
- Use Business Object workflow nodes only; do not create or use an External REST tool.
- Opportunity references in user requests are full ``OptyNumber`` values, not internal ``OptyId`` values.
- Preserve the full supplied opportunity number exactly, including prefixes, underscores, letters, and leading zeroes. For example, ``CDRM_152867`` must remain ``CDRM_152867``.
- Do not strip, parse, normalize, convert, or look up an ``OptyNumber`` to obtain an ``OptyId``.
- Add the child resource using the full opportunity number directly in the ``OpportunityResource`` child-operation path.
- The team-member payload must use the selected resource's ``PartyId`` as ``ResourceId``. Do not use ``ResourceProfileId`` as ``ResourceId``.
- Do not include ``OptyId`` in the child request body.

Implement:

1. Extend ``CLASSIFY_INTENT`` with ``ADD_TEAM_MEMBER``.
2. For that intent, extract:
 - ``opportunityNumber``: the exact user-supplied full `OptyNumber`
 - resource lookup type and value: name, email, or `ResourceProfileId`
 - only explicitly supplied supported team-member attributes
3. Do not classify generic opportunity updates as ``ADD_TEAM_MEMBER``.
4. Add the intent route and input validation.
5. Find existing BOs that can help query resources using various fields such as resource id, name, resource number, email address or login
6. If there is no existing BO, create a BO that you can use to search for resources using the above attributes. The openapi describe for resources is in the folder.
7. Return clear messages for missing opportunity number, missing resource information, zero resources, multiple resources, and API failures.
8. On exactly one resource match, map ``PartyId`` to ``ResourceId``.
9. Add or update ``OPPORTUNITIES.CreateOpportunityResource`` as a BO function with:
 - POST ``/crmRestApi/resources/11.13.18.05/opportunities/{opportunityNumber}/child/OpportunityResource``
 - token parameters ``opportunityNumber`` and ``requestBody``
9. Make ``ADD_OPPORTUNITY_TEAM_MEMBER`` a ``BO_FUNCTION`` node using ``OPPORTUNITIES.CreateOpportunityResource``.
10. Bind the full ``opportunityNumber`` directly to that BO node. Do not introduce an opportunity-ID lookup node.
11. Wire all new nodes into the real execution path.
12. Preserve all existing list, filtered-list, create, update, revenue, and out-of-scope behavior.

After changes:

- Run ``do-prettyfy-workflow``.
- Run ``validate-workflow``.
- Run ``validate-bo`` for the Opportunities BO.

- Complete the workflow test-sync loop and add coverage for successful team-member addition with an opportunity number, missing number/resource, no resource match, multiple resource matches, and an existing update regression.
- Keep changes local unless I explicitly ask you to save the BO or workflow to the server.

In the final response, state the full-number behavior, the `PartyId` → `ResourceId` mapping, the BO function used, and validation/test results.

Instructions given to the code assistant

Use the code assistant to create the prompt that you can then pass to the assistant to build the agentic app. Here are the instructions used to generate the above prompt.

Review this workflow:

```
`src/workflows/opportunity_management.wf`
```

I want to enhance it to support adding an existing Fusion Resource as a team member to an existing opportunity.

Please generate a detailed implementation prompt for Codex. Instruct Codex to use the AI Studio skill at:

```
`<Path To>\aistudio\SKILL.md`
```

I have included the Fusion REST describe files:

- `resources-describe.json`
- `opportunities-describe.json`

Additional requirements to include in the generated Codex prompt:

- A team member is an existing Resource; never create a Resource record.
- Use Business Object workflow nodes only; do not use an External REST tool.
- Find existing BOs that can help query resources using various fields such as resource id, name, resource number, email address or login
- If there is no existing BO, create a BO that you can use to search for resources using the above attributes. The openapi describe for resources is in the folder.
- Opportunities are identified by their full Opportunity Number (`OptyNumber`), not `OptyId`.
- Use the full `OptyNumber` directly when adding the OpportunityResource child record. Do not look up or use an `OptyId`.
- Use the selected Resource's `PartyId` as the OpportunityResource `ResourceId`; do not use `ResourceProfileId`.
- The child POST should use the `OpportunityResource` child resource under the supplied opportunity number.
- Keep existing list, filter, create, update, revenue, and out-of-scope behavior unchanged.
- Include validation, deterministic no-match/multiple-match resource handling, workflow wiring, formatting, and test requirements.

Generate only the detailed prompt I can paste into Codex.

Build an Agentic Application with a Code Assistant

Scenario: Build a My Pipeline app for a salesperson.

Use the `aistudio` skill to build a local-only agentic app named "My Pipeline" in the current AI Studio project. Do not publish or save remote artifacts.

First, read the AI Studio skill instructions and all app/workflow/BO references needed for this work. Inspect the existing project layout and local artifacts. Do not run `init`.

Important: before creating or editing any AI Studio artifact, present the structured pre-build checkpoint requested below and wait for my explicit confirmation. This request is not itself authorization to create artifacts.

Goal

Build an agentic app for an individual sales representative. It must show only that rep's own open leads and opportunities, help them prioritize those records, and support the confirmed write actions.

Data sources

Create exactly these two Business Objects and two app-backed workflows unless I explicitly approve additional artifacts:

1. Leads BO and Leads workflow

- Describe contract: `leads-describe.json` in the project root; if absent, retrieve the contract using the project's configured Fusion authentication.
- REST resource: `leads`.
- Use the `MyLeadsFinder` finder, titled "My Open Leads," for the open-leads scope.
- Relevant fields:
 - `LeadId`
 - `Name`
 - `PrimaryContactPartyName`
 - `DealAmount` – deal-size ranking value
 - `EstimatedCloseDate`
 - `Score`
 - `Rank`
 - `OwnerId`, `OwnerPartyName`, `OwnerPartyNumber`
 - `StatusCode`
- Supported operations:
 - collection read: `GET`
 - item read/update: `GET` / `PATCH`
 - `POST acceptLead(leadId)` – this is the closest available operation to "qualify," but it is not named qualify.
 - `POST convertLeadAndGetOppty(leadId, opportunityName, opportunityOwnerNumber, enableCreateContact, enableCreateAccount, attributeMap)` – preferred conversion operation because it returns an object.
 - `POST convertLeadToOppty(...)` is also available.

2. Opportunities BO and Opportunities workflow

- Describe contract: `opportunities-describe.json` in the project root; if absent, retrieve it using the project's configured Fusion authentication.
- REST resource: `opportunities`.
- Use the `MyOpportunitiesFinder` finder, titled "My Open Opportunities," for the open-opportunities scope.
- Relevant fields:
 - `OpptyId`
 - `Name`
 - `TargetPartyName` – account
 - `PrimaryContactPartyName`
 - `Revenue` – amount
 - `WinProb` – writable integer win probability
 - `StatusCode`, `LookupCategory`, `StageStatusCd`
 - `EffectiveDate` – close date
 - `OwnerResourcePartyId`, `OwnerPartyNumber`, `PartyName1`
- Supported operations:
 - collection read: `GET`
 - item read/update: `GET` / `PATCH`
 - Change probability by PATCHing only `WinProb`.

Identity and record scope

- Scope every read to the current logged-in sales rep and open records only.
- Prefer the two "My Open ..." finders.
- Do not assume the service-account identity or `aistudio.fa-user` is the current rep.
- Determine how runtime user context maps to Fusion's resource/party identifier. If launch context already provides the resource party ID, use it. If not, identify a supported source for resolving it; if the contracts do not provide one, flag that as a dependency in the pre-build checkpoint and wait for my decision.
- Do not implement a hard-coded owner ID or a broad unfiltered query.

Ranking

- Leads: rank descending by `DealAmount`; default top N is 8.
- Opportunities: rank descending by calculated expected value:
`Revenue \* (WinProb / 100)`.
- Perform ranking after retrieval if the API cannot sort by the calculated expression.
- Handle null amounts/probabilities safely and display a clear fallback label.

App experience

Create a "My Pipeline" agentic app with:

- A short text summary at the top: open lead count and total deal amount, open opportunity count and total revenue/expected value, plus the single highest-priority lead and opportunity.
- Panel 1: "Sales Lead Summary"
 - Default top 8 leads by `DealAmount`.
 - A ranked horizontal bar chart using lead name and `DealAmount`.
- Panel 2: "Opportunity Summary"
 - Default top 8 opportunities by expected value.
 - A bubble/scatter chart: x = `WinProb`, y = `Revenue`, bubble size = `Revenue`.
- Expanded Leads list columns:
`Name`, `Primary Contact`, `Amount`, `Close Date`, `Score`, `Rank`.
- Expanded Opportunities list columns:
`Name`, `Account`, `Primary Contact`, `Amount`, `Win Probability`, `Status`, `Close Date`.

Use the AI Studio-supported widget and layout contracts. Inspect the relevant app/widget references rather than inventing widget IDs or output payload shapes.

App-backed workflow design

- Implement real app-stage routing using `\$context.\$app.\$OraMessageHint`.
- Provide separate, routable terminal paths for at least `InitDisplay` and normal query behavior. Do not implement multiple app stages in one shared terminal prompt.
- Fetch shared record data before the stage router where appropriate.
- Clarify in the checkpoint how the top summary will combine lead and opportunity facts while preserving the requested two-workflow scope. Do not silently create a third summary workflow.
- Add useful read-only free-form questions, such as "What should I focus on first?" and "Why is this opportunity ranked highly?"
- Do not add communications unless I explicitly approve them in the checkpoint.

Actions and side effects

1. Lead "Qualify"

- The qualify lead is PATCH operation on the lead where you set the `StatusCode` to `QUALIFIED`;
- Refresh the Leads list after the successful PATCH and verify that the persisted value remains after refresh.

2. Lead conversion

- Use `convertLeadAndGetOpty` after confirmation.
- Before execution, show the lead name and conversion inputs, explain that it creates an opportunity, and clearly state that the describe contract does not specify the source lead's resulting status or lifecycle.
- After conversion, inspect the returned object and refresh the source lead and opportunity lists. Report the actual observed result rather than claiming that the lead was deleted, closed, or otherwise changed without evidence.

3. Opportunity probability

- Offer +10% and -10% actions.
- Calculate `newWinProb = min(100, max(0, currentWinProb ± 10))`.
- Require confirmation before PATCHing the item with only `WinProb`.
- Refresh the opportunity list after the successful PATCH and verify that the persisted value remains after refresh.

Pre-build checkpoint

Before creating artifacts, present this concise structured checkpoint and wait:

Data source

- Confirm the two Fusion REST contracts, the My Open finder for each source, and the user-to-resource-party resolution approach.

App goal and user

- Individual sales representative; own open leads and opportunities.

First-load experience

- Top summary, lead panel, opportunity panel, top-N default of 8, ranking calculations, and chart plan.

Query behavior

- Read-only natural-language questions and which workflow handles them.

Priority actions

- `acceptLead` mapping decision, `convertLeadAndGetOpty`, and `PATCH WinProb`.

Communications

- Omit for now unless I request them.

Side effects

- Confirmation requirements, refresh behavior, source-lead post-conversion uncertainty, and error handling.

Launch context

- Exact current-user/resource-party fields required, or the unresolved dependency.

Validation

- Demonstrate: on load, the rep sees the top 8 open opportunities ranked by expected value; changing one opportunity from 40% to 50% persists after refresh; the approved lead action changes the observed status; and converting a lead creates and returns an opportunity while the app reports the source lead's actual post-conversion state.

Implementation and verification

After I approve the checkpoint:

- Create the two BOs, two app-backed workflows, and app artifacts through the bundled AI Studio CLI—not by hand-editing supported artifact structures.
- Validate every BO, workflow, and app.
- Run the required workflow/app test-sync process specified by the AI Studio skill.
- Do not claim validation passed unless command output proves it.
- End with a concise implementation summary, validation evidence, known limitations, and the exact unresolved behavior if the user-to-resource identity mapping or the Qualify-to-acceptLead mapping remains unapproved.

Note: You must provide a schema for the 2 business objects: Sales Lead and Opportunity. You can provide them as json files.

Instructions given to the code assistant

Use the code assistant to create the prompt that you can then pass to the assistant to build the agentic app. Here are the instructions used to generate the above prompt.

Use the /aistudio skill to create a detailed prompt that I can use in codex to achieve the below

Use Case

Build an agentic app called My Pipeline.

Purpose

- An agentic app that shows a sales rep their leads and opportunities and helps them progress both toward closure.
- Primary user: an individual sales rep. The app should scope to **that rep's own records** (see Record scope below).

Questions

If you have any questions on the use case, feel free to ask.

Data sources & authentication

Create two app-backed workflows and their corresponding business objects:

1. **Leads** — OpenAPI describe: `leads-describe.json`. This file can be found in the project folder.
2. **Opportunities** — OpenAPI describe: `opportunities-describe.json`. This file can also be found in the project folder.

```
Use the describe output as the source of truth for field names, resource operations, and any custom actions
(do not assume field or action names).

**Authentication:** Use the authentication already setup in VS code project for this.

## Record scope

- Show only records **owned by the current rep**, and **open** records only.
- The current rep is the logged in user

## "Most important" definition

- **Lead importance** = deal size / deal amount. Higher amount = more important.
- **Opportunity importance** = a blend of amount and win probability (WinProb). Rank by **expected value =
  Amount x WinProb** so both dimensions count, unless the describe surfaces a better native ranking field.

## Summary section

Show a short text summary of the rep's pipeline at the top (e.g. count and total value of open leads and
opportunities, and the single highest-priority item in each), then two panels.

### Panel 1 - Sales Lead summary

- Show the **top N (default 8)** most important open leads.
- Chart: a **ranked horizontal bar chart** of those leads by Amount.

### Panel 2 - Opportunity summary

- Show the **top N (default 8)** most important open opportunities.
- Chart: a **bubble/scatter chart** - x = Win Probability, y = Amount, bubble size = Amount - so amount and
  probability are both visible in one view.
## Panel expansion

### Panel 1 expanded - Leads list

Columns: **Name, Primary Contact, Amount, Close Date, Score, Rank**

Actions on a lead (confirm the exact operation from the describe; these are likely custom actions, not plain
field updates):
- **Qualify lead** - call the qualify operation. Ask for confirmation before writing.
- **Convert to opportunity** - call the convert operation. Ask for confirmation before writing, and state
  what happens to the source lead after conversion.

### Panel 2 expanded - Opportunities list

Columns: **Name, Account, Primary Contact, Amount, Win Probability, Status, Close Date**

Action on an opportunity:

- **Change win probability** - increment by **10%** or decrement by **10%**. Clamp the result to the
  **0-100** range. Confirm the write operation (likely a PATCH on the opportunity) and whether it needs
  confirmation before saving.

## Validation scenario

Prove the app works with this flow: `<e.g. "On load, the rep sees their top 8 open opportunities by
expected value. Nudging opportunity X from 40% to 50% persists on refresh. Qualifying lead Y changes its
status, and converting lead Z creates an opportunity.">>`

## Build expectations

- Confirm the two business objects and the exact read/write operations from the describe endpoints before
  wiring anything.
```

- Present your structured pre-build checkpoint (data source, app goal/user, first-load behavior, panels, actions, side-effect handling, launch context, validation) and wait for my confirmation before creating artifacts.

Before Creating the Prompt:

Before creating the prompt, confirm the business objects and API operations from the `/describe` endpoints, then present your prompt for my confirmation.

Add Or Modify One Application Panel Area

Scenario: `src/apps/my_pipeline.apps` exists and needs a converted leads panel backed by a workflow agent.

You are an AI Studio expert assistant. Using the `aistudio` skill, please add a new "Converted Leads" panel to the "My Pipeline" application that displays leads that have already been converted to opportunities.

Context & Target File

- Application: ``src/apps/my_pipeline.apps``
- Backing Workflow: ``src/workflows/my_pipeline_leads.wf``
- Business Object: ``src/businessObjects/my_pipeline_leads.bo``

Requirements

- Application Configuration (``src/apps/my_pipeline.apps``):
 - Add a new panel entry to the application with heading: "Converted Leads"

Data source

- Extend ``MY_PIPELINE_LEADS`` with a read-only query for converted lead records and their linked opportunity details.

App goal and user

- Sales representatives can review leads that have already become opportunities.

First-load experience

- A "Converted Leads" panel shows lead name/number, conversion date, and resulting opportunity name/number in a multi-record list.

Query behavior

- The panel answers questions using only the converted-lead result set.

Priority actions

- Omit for now.

Communications

- Omit for now.

Validation

- Verify a converted lead with a linked opportunity renders in the new panel.
- Validation & Quality Sync:
 - Run workflow prettify and validation
 - Validate the app
 - Synchronize workflow and app tests

Fetch Or Sync an Existing Application for Local Review

Use this pattern when you already have an application code and explicitly want local server state before editing.

Use AI Studio CLI read-only `fetch/search` commands to fetch the latest local artifact set for app ``<APP_CODE>``.

Rules:

```
- Do not edit, deploy, or delete anything.
- Prefer CLI fetch and search commands.
- Prefer DRAFT status when it exists; otherwise use PUBLISHED.
- Do not invent missing schemas, examples, topics, deeplinks, approval processes, or payloads.

Discover where supported:
- App
- Direct workflow agents
- Nested workflow agents
- Actions
- Communications
- Templates
- BOs
- Deeplinks
- Topics
- Approval processes
- Tools, connectors, policies, document schemas, and functions referenced by the app or workflows

Validate all the local files once fetched.

Final report:
- Artifact inventory.
- Source used for each artifact.
- DRAFT/PUBLISHED status used.
- Missing artifacts or unsupported fetch gaps.
- Validation errors.
- Confirmation that no artifact was updated or deleted.
```

Test a Workflow or Agentic Application Created or Edited with AI Studio CLI

You can test your workflows and agentic applications by working with a code assistant. You describe the behavior to be verified and the coding assistant handles the technical test set up and execution.

Understand Your and the Coding Assistant’s Roles

You provide	The coding assistant handles
The workflow or application to test.	Finds the correct local files and assesses current coverage.
Business scenarios and expected outcomes.	Creates, updates, and validates the required tests.
Safe sample records or required identifiers, when live data is necessary.	Records or prepares replay data and runs the focused tests.
Confirmation before tests could change business data.	Diagnoses failures and gives you a clear report.

Prepare to Request Tests

- Tell the assistant which workflow or application you changed.
- Explain the new or changed business behavior in plain language.
- Provide safe, nonproduction sample data if the behavior creates, updates, or deletes records.
- Say whether you want the assistant to only review the test plan, or to create and run the tests.
- If you created a brand new application or workflow then ask it to identify and create all the tests for the application or workflow.

Test a Workflow

Ask the code assistant to first inspect the workflow's test sync plan. This is a checklist of the business scenarios that are already covered and those that still need tests. For each missing scenario, the assistant will choose an appropriate data approach.

Approach	Description
Record now	Safely run the scenario once and save the returned data for later replay.
Record later	Create the test now and wait until you can supply safe data or access.
Reuse a baseline	Reuse recorded data from a similar test.
Use generated data	Prepare safe, coherent test data when live recording isn't appropriate.

After the tests are ready, the assistant runs the affected test and then the focused workflow suite. It should tell you how many tests passed, link the report, and explain any next step in natural language.

Identify Good Workflow Scenarios

Describe business cases, not node names. Here's a useful starter set.

- A normal successful request.
- An empty or “no matches found” result.
- A missing required input.
- A validation or service-error path.
- An out-of-scope request, if the assistant has scope limits.
- Each important create, update, or delete path—using safe test records.

The Opportunity Manager sessions used this approach. They covered listing, filtering, creating, updating, revenue actions, out-of-scope messages, and conditional routes covering 15 scenarios across two workflows.

Test an Application

An application test checks the panel experience: the application starts the panel, calls the backing workflow, checks the response, and can confirm that the expected widget or layout appears.

Ask the assistant to do the following steps:

1. Complete the relevant backing-workflow tests first.
2. Inspect which panel tests are missing or out of date.
3. Create and record the needed panel tests.
4. Run the application tests and summarize the results.

Conversation Workflows with HUMAN and WAIT Steps

If a workflow pauses for a person or a timed wait, ask the assistant for a multiturn conversation test. It will send an initial request, confirm the pause, and resume using the appropriate action.

- For a HUMAN step, test approval, rejection, and when applicable, feedback.
- For a WAIT step, test the timeout path.
- Create separate tests for approval and rejection so each result is easy to understand.
- If the workflow relies on earlier conversation context, include a follow-up request that uses a fact stated only in the first request.

Improve Cost and Quality through Testing

After the tests are stable, ask the code assistant to compare model choices for selected workflow nodes. It can test alternatives without changing the saved production workflow, then recommend a model placement only when the relevant end-to-end tests continue to pass.

In the Opportunity Manager sessions, this approach reduced reported AI Units while keeping all tests passing: the intent classifier stayed on a premium model, while several formatting and revenue-classification nodes moved to a basic model.

Sample Requests for a Coding Assistant

Here are some sample prompts that you can give to the code assistant:

- Review coverage only

```
Review the test coverage for my Opportunity Management workflow. Tell me which important business scenarios are missing, but do not create or run tests yet.
```
- Create and run workflow tests

Create and run the missing tests for my workflow. Cover listing, filtering, creating, updating, no-results, errors, and out-of-scope requests. Use safe replay data where possible and keep all changes local.

- Test a workflow that changes data

Create tests for the revenue update and delete paths. Use this safe test opportunity number and revenue record ID. Do not use production data. Run the focused suite and summarize the results.

- Test an application panel

Test the Opportunity Details panel and its backing workflow. Verify that the panel opens, returns the correct information, and displays the expected layout. Create or update the needed tests and provide a concise report.

- Check a model-cost change

Compare lower-cost model options for the formatting nodes in this workflow. Do not change the saved workflow. Recommend a model only if the relevant tests still pass.

Recommended Usage Flow for AI Studio CLI

1. Install or open the VS Code extension, or use the standalone bundled CLI from a normal terminal.
2. Start from the AI Studio project root.
3. Initialize only when creating a blank project: `init`.
4. Configure environment and authentication.
5. Ask a code assistant to inspect local files, project layout, and command help before changing artifacts.
6. Confirm real data sources before building BO-driven or data-dependent application behavior.
7. Run `do-prettify-workflow` and `validate-workflow` after material workflow structure edits.
8. Run workflow test sync after material workflow changes.
9. Run app test sync after material app changes when the app has top-level agent containers.
10. Review diffs and test reports.
11. Save, publish, unpublish, or delete remote state only through the approved release path or explicit user request.

This preserves the benefits of vibe coding while keeping local files, deterministic CLI operations, validation, tests, and reviewable diffs as the control points.

14 Integrate Applications and Workflows with Continuous Integration and Continuous Delivery (CI/CD)

Integrate Applications and Workflows with CI/CD

A continuous integration and continuous deployment (CI/CD) process maintains supported application and workflow source artifacts in Git. Changes are reviewed, packaged, deployed to a target environment, and tested. Version control provides a history of changes and supports collaborative development.

Key Capabilities

The CI/CD process includes the following activities:

- **Version-controlled source:** Maintain supported application, workflow, and related metadata files in the project Git repository.
- **Reviewed changes:** Use branches and merge requests to review changes according to your project's approval requirements.
- **Packaging and deployment:** Track package creation and deployment to each target environment separately.
- **Deployment verification:** Confirm that the expected artifacts and intended version are available in the target environment, then validate their behavior.

Prerequisites

Before you begin, confirm that you have access to the project repository and review process, the supported deployment tooling, and the target AI Agent Studio environment. Make sure the environment and tooling meet the requirements for your release and that the account used for testing can execute the relevant applications and workflows.

Follow your project's repository and approval conventions. Review pipeline triggers and target environments before running a deployment.

Scope

Include only artifact types supported by your release and deployment method. Review references between applications, workflows, and related artifacts, and ensure that required dependencies are available in the target environment.

Use CI/CD in AI Agent Studio

Here's how you can use CI/CD in AI Agent Studio:

1. Prepare the source change.

Create a branch for the change and update the source files using the required repository structure. Validate artifact syntax and required properties, review dependency references, and run the applicable tests before submitting the change for review. Commit the source changes and associated tests.

2. Submit and review the change.

Create a merge request for the target branch. Describe the application or workflow change, include validation results, and obtain the approvals required by your project before releasing the change.

3. Build and deploy the approved change.

Run the configured pipeline for the approved repository revision. Review build and deployment results for each target environment, including failed or skipped artifacts. Confirm that the expected artifacts are available and that the intended version is active. If a failure occurs, retain the deployment reference and relevant logs, correct the issue, and rerun the affected process.

4. Validate the deployed application.

Run representative application and workflow scenarios in the target environment. Verify required dependencies, permissions, live integrations, expected responses, and business actions. When automated tests replay recorded or synthetic data, perform separate checks of the current live integrations and data.

Best Practices

- Use short-lived branches, and describe merge requests in terms of their outcomes.
- Record the source revision, deployment reference, and test-report links with the change.
- Verify expected artifacts and runtime behavior, even when the pipeline reports success.
- If build or deployment fails, preserve the build number and console-output reference, correct the source or configuration, and run a new build.
- Review supported artifact types and dependencies before deployment.

15 Allow External Access to Fusion Applications Agents

Enable Applications to Access Fusion Applications Agents

You can access the agents of Oracle Fusion Cloud Applications from other Oracle applications and from external applications, using the `/invokeAsync` API. The access is based on the role assigned to you, and whether that role can access workflows. The roles that can access a workflow are specified in the Security tab when creating or editing the workflow in AI Agent Studio.

Access from Other Oracle Applications

To use `/invokeAsync` API, you must authenticate through Oracle Cloud Infrastructure Identity and Access Management (OCI IAM) using an OAuth 2.0 bearer token. For agent operations to function, integration must be bidirectional. Both systems must securely access each other's services.

As an administrator, you create the confidential application in Fusion Applications IAM, representing the other Oracle application service. Then create a matching confidential application in the other application service, representing the AI Agent Studio's service. These two applications form the trusted link for the two-way communication.

Set Up Two-Way Trust Between Applications

1. Create a confidential application in Fusion Applications IAM:

This application acts as a representation of the external Oracle application in Fusion Applications IAM. After creating the confidential application, enable access to the AI Agent Studio service so the application can call the `/invokeAsync` API exposed by this service.

- Identify the identity domain (authorization server) where you'll create your confidential application.
- Specify OAuth grant types. For example, a 2-legged OAuth.
- Define the access scope.
- Generate the access token and use it to make REST API calls.

For more information, see [Configure OAuth Using the Fusion Applications Identity Domain](#).

2. Create a confidential application in the external Oracle application's IAM:

Similar to the previous confidential application you created, create another confidential application that acts as a representation of the Fusion Application service in the external Oracle application's IAM. After creating the confidential application, configure it to allow AI Agent Studio to call the appropriate service endpoints.

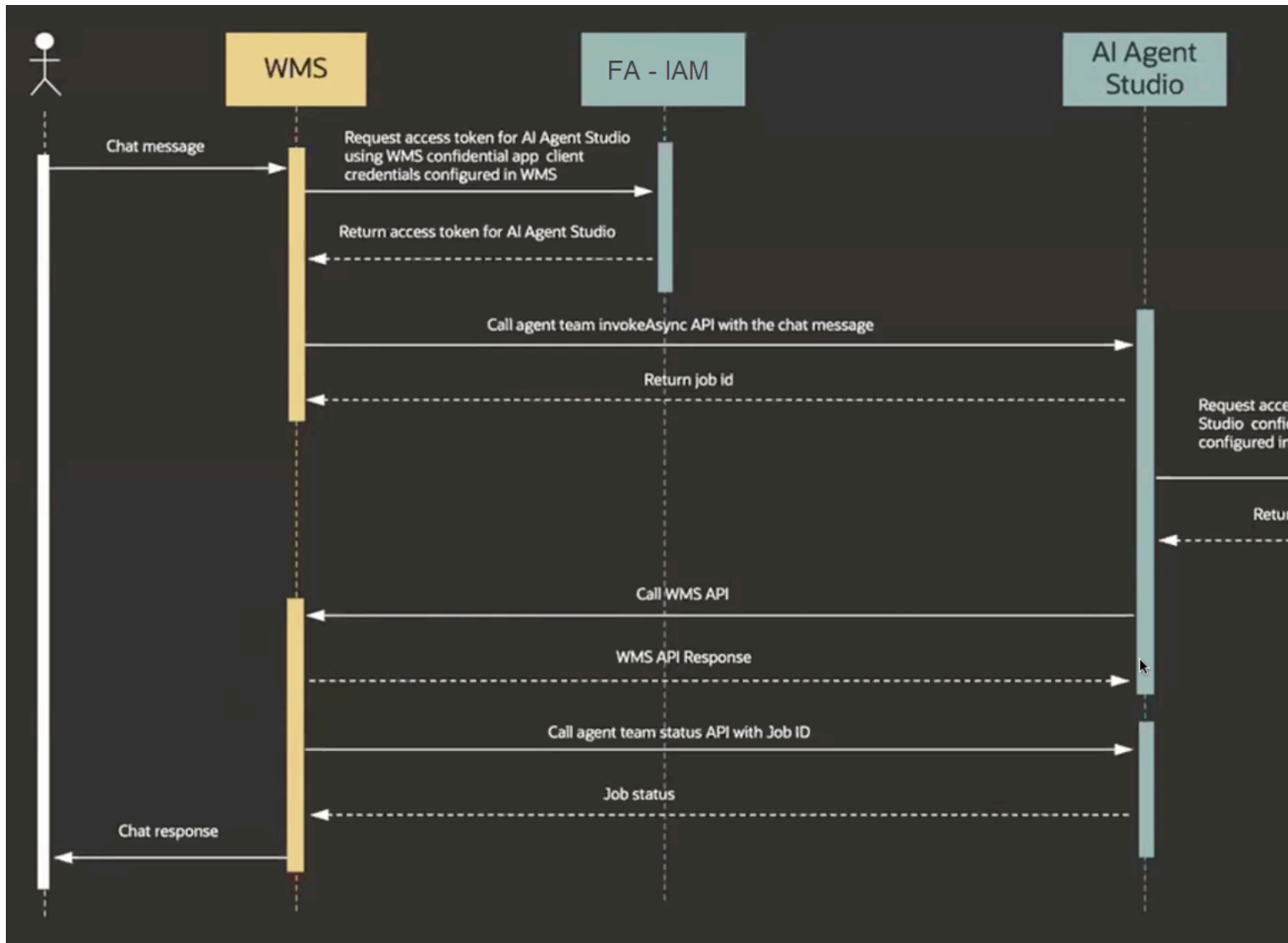
For more information, see [Configure OAuth Using the Fusion Applications Identity Domain](#).

3. Register the confidential application created in the external Oracle application's IAM (from step 2) in AI Agent Studio:
 - a. Go to **AI Agent Studio** and open the Credentials tab.
 - b. Go to the Data Source Applications subtab and add the details of the external confidential application.
 - Base URL: Enter the base URL of the external API
 - IDCS URL: Enter the URL of the Fusion Applications IAM, to enable AI Agent Studio to get the OAuth token
 - Scope: Specify what access needs to be given whenever that OAuth token is retrieved.
 - Public and Private keys: Enter the keys for signing authentication requests. Used by IDCS to verify that the request is from a trusted and authorized source.
4. Create a business object to access the external Oracle application:
 - a. In **AI Agent Studio**, open the Business Object tab.
 - b. Add a new business object with the **Resource Type** as **Other Data Source Application**.
 - c. From the **Oracle Data Source Application** list, select the data source application that you've registered in step 3.

Access Flow During Runtime

At runtime, the external Oracle application obtains an access token from Fusion Applications using the configured credentials. The application then uses this token to call the `/invokeAsync` API in AI Agent Studio, enabling access to the workflow.

This illustration shows the interactions between Warehouse Management System (WMS), which is an Oracle application outside Fusion Applications, Fusion Applications IAM, AI Agent Studio, and WMS IAM.



- The user interacts with the WMS application and sends a chat message.
- WMS requests an access token from Fusion Applications IAM using its configured credentials.
- WMS calls the `/invokeAsync` API in AI Agent Studio, passing the user's chat message and access token.
- AI Agent Studio returns a Job ID to WMS, which WMS uses to track progress and get the response.
- To call WMS API, AI Agent Studio requests an access token from WMS IAM.
- AI Agent Studio calls the WMS API using the access token.
- WMS returns the API response and requests for the job status.
- AI Agent Studio returns the job status.
- Final chat response is processed and delivered to the user.

Access from External Applications

Use `/invokeAsync` API to access the agents of Fusion Applications. For more information, see [Agent Team REST Endpoints](#).

16 Collaborate with AI Agents Across Platforms Using Agent2Agent (A2A) Protocol

Collaborate with AI Agents Across Platforms using Agent2Agent (A2A) Protocol

The published AI agents in Oracle Fusion Cloud Applications can collaborate with AI agents in other platforms using the A2A protocol. A2A is an open, industry-backed protocol that enables AI agents to discover, understand, and call other agents in a standardized way. In AI Agent Studio, A2A serves as the foundation for seamless cross-agent collaboration across Fusion Applications and external ecosystems, eliminating the need for custom point-to-point integrations.

Here are the key features of A2A:

- A standardized way to describe agents through an Agent Card
- A standard request-and-response model for agent interactions
- A shared lifecycle framework for managing long-running and asynchronous tasks

Here's an example scenario for using A2A. If your organization uses an Oracle AI agent for sales managers, that agent can reach out to a third-party agent to get travel options and cost quotes when considering the sales calls. Similarly, third-party platforms can call Oracle AI agents using A2A. Another example scenario for using A2A is when your organization has a general purpose AI chat assistant for employees. You can reach out to subject matter expert agents in Oracle Fusion Applications when customers ask about HR Sickness and Absence Policy or Vacation Bookings. Here's how the interaction using A2A works:

1. Each agent publishes an Agent Card that describes the agent to the third-party platform.
2. Third-party platforms can then call the Fusion Applications agent in a consistent and secure way.

The Fusion Applications A2A API provides clients a practical way to discover, call, poll, and stream workflow-backed agents using A2A-style endpoints and method names. As an authenticated, client-facing API, it offers the following key properties:

- A workflow-backed agent model
- A task-oriented asynchronous interaction model
- A pragmatic subset of broader A2A capabilities

Components in A2A

Here are the core components of A2A interactions:

- Agent Catalog or Agent Registry
- Agent Card

Agent Catalog (Registry)

The Agent Catalog, also referred to as the Agent Registry, acts as a searchable directory of available agents. It doesn't call agents or manage their internal logic.

Here are the key responsibilities of the agent catalog:

- Maintain the list of available agents.
- Index agent metadata for search.
- Enforce access rules.
- Return agent details, that is, Agent Cards.

Agents can be discovered in two main ways:

- Listing Available Agents: By passing an empty query, a client can retrieve a list of all agents it's allowed to access.
- Keyword Search: The agent registry supports keyword-based search across agent metadata. Typical search scenarios include finding an agent by name or description. The search results return agent references, that is, each result points to the agent's Agent Card.

Agent Card

An Agent Card is a machine-readable JSON document that describes an agent. It's the core building block that makes discovery and interoperability possible in A2A. You can consider an Agent Card as an agent's digital profile or business card.

An Agent Card allows a client to answer the following questions without prior knowledge of the agent:

- What's this agent?
- What capabilities or skills does it provide?
- How do I call it?
- What authentication is required?

Call Agents

After an agent is discovered and its Agent Card is retrieved, calling the agent follows a consistent pattern - using asynchronous calls or streaming. Streaming includes `message/send` and `message/stream` methods.

1. Discover the agent using the registry.
2. Retrieve the agent card.
3. Authenticate according to the agent card's requirements.
4. Call the agent using the A2A invocation API.

Authentication and Access for A2A

When using A2A, all endpoints require an authenticated caller. Here are some key points about the Fusion Applications A2A surface:

- It's externally exposed to authorized clients and integrations.
- It's not anonymous or publicly open without authentication.
- It's authenticated for discovery, invocation, status retrieval, and streaming.

The agent card advertises the **Bearer** authentication.

Authorization Behavior

All endpoints require authentication, but clients must also understand the distinction between discovery and execution behavior. A client might be able to discover an agent entry, but actual processing or status access might still depend on user-specific authorization. Status retrieval is user-scoped, and authorization to process is enforced at processing time.

Clients must therefore treat discovery as a way to find candidate agents, and processing as the point where final access is confirmed.

Auth, OAuth, Bearer Tokens, and Authorization

Authentication means proving the identity of the caller. Here **OAuth** is the standard token-based mechanism used to obtain access credentials. A **bearer** token is the token the client sends on the request, usually in the **Authorization** header. Authorization means determining what the authenticated caller is allowed to access or do.

A high-level overview of the authentication process includes the following:

- Clients authenticate by presenting a bearer token.
- That bearer token is obtained through an OAuth flow.
- Authentication gives access to the API.
- Authorization determines whether the client can run a workflow or retrieve a task.

A2A API Overview

The API consists of three endpoints:

1. **GET** /orchestrator/a2a/v1/{workflowCode}/.well-known/agent.json
2. **POST** /orchestrator/a2a/v1/agents/search
3. **POST** /orchestrator/a2a/v1/{workflowCode}

Together, these endpoints support the following actions:

- Agent specific discovery
- Registry-style agent search
- Streaming and nonstreaming invocation and task retrieval

Endpoint	When to use	Request	Response and how to use it	Notes
Agent Card Discovery GET /orchestrator/a2a/v1/{workflowCode}/.well-known/agent.json	Use this endpoint when the client already knows the target workflow code or when registry search has already returned a workflow-specific agent-card URL.	The request is identified by: workflowCode: the published workflow to discover No request body is required.	The response is an agent card for the specified workflow-backed agent. Clients must expect the card to provide the following details: • Agent identity information • Version information • Provider information • Authentication requirements • Capabilities such as streaming support • Default input and output modes • Skills associated with the agent How to use the response Clients must use the card to do these actions: 1. Confirm they have the correct agent. 2. Inspect the advertised authentication and capability model. 3. Determine the invocation path for subsequent calls.	• The card is agent-specific rather than host-wide. • The card model is intentionally narrower than a richer full-specification A2A card.
Registry Search POST /orchestrator/a2a/v1/agents/search	Use this endpoint when the client needs discovery rather than direct lookup. Here are some typical scenarios: • Browsing available agents • Searching by name or related query text • Paging through results	The request body supports the following options: • query: optional search text • skip: optional pagination offset • top: optional page size limit	The response is a registry-style result set containing these details: • Version metadata for the registry response • A collection of matching resources • A total count Each resource points to an agent card. How to use the response Here's the recommended client flow: 1. Call registry search. 2. Select a matching result.	• This is a Fusion-specific registry-style endpoint, not a generic standards-level registry service. • Results are agent-backed. • Search success must not be treated as a guarantee that subsequent execution will be authorized in every case.

Endpoint	When to use	Request	Response and how to use it	Notes
			- Fetch that result's agent card. - Call the selected agent.	
Nonstreaming Invocation POST /orchestrator/a2a/v1/{workflowCode}	Use this endpoint when the client wants a standard request/response interaction rather than a Server-Sent Event (SSE) stream. This endpoint supports the following operations: - Starting asynchronous work from a message - Starting or resuming task-oriented work - Polling the latest state of an existing task	The request body supports the following request patterns: message/send message/stream tasks/send tasks/get Request patterns message/send Use message/send to begin asynchronous work from a conversational message. Typical request content includes these details: - A message object - Message parts - Optional context information - Optional metadata message/stream The response is returned as an SSE stream. tasks/send Use tasks/send to start or continue task-oriented execution. Typical request content includes these details: - A task identifier when continuing or resuming work - A message payload - Optional context information - Optional metadata tasks/get Use tasks/get to retrieve the latest state of an existing task.	The endpoint returns a JSON-RPC-style response whose result is task-oriented. Clients must expect task-like responses representing these states: Submitted Working Input-required Completed Failed Completed and waiting tasks might include output artifacts. For message/stream, clients must expect these responses: - An initial event representing the initial task snapshot. - Additional events as task state changes. - Completion of the stream when execution finishes. How to use the response Here's a common client pattern: - Call message/send or tasks/send. - Receive a task-shaped response - Continue polling with tasks/get until the task reaches a terminal state. Use message/stream when streaming is preferred over polling. Prefer nonstreaming invocation plus tasks/get for these situations:	- Even message/send is task-oriented in practice. - Clients must expect asynchronous behavior rather than an immediate final answer. - The message/stream behavior can be considered as repeated task snapshots over SSE. - Clients must not assume a richer event-native A2A stream contract beyond the task-states.

Endpoint	When to use	Request	Response and how to use it	Notes
		Typical request content includes these details: - The task identifier - Optional history preferences	- Polling is simpler for the client. - The integration doesn't need live incremental updates.	

API Task Lifecycle

The API is asynchronous by design. The typical client lifecycle includes these steps:

1. Identify the target agent through direct card lookup or registry search.
2. Call API using `message/send`, `tasks/send`, or `message/stream`.
3. Receive a task-oriented result.
4. Continue through polling or streaming until the task completes.
5. Read final output from the resulting task state and artifacts.

Here are the available task states:

- `submitted`
- `working`
- `input-required`
- `completed`
- `failed`

Clients must be prepared for a task to enter the `input-required` state before reaching completion.

Input and Output Expectations

Here are the input and output expectations for the API.

Inputs

The API is primarily text-oriented. Clients must assume the following inputs:

- Text is the primary supported input mode
- The most important request content is the message text
- Metadata might be included when additional execution context is needed

Outputs

The API returns task-oriented outputs. Clients must expect the following outputs:

- Task identifiers
- Task state
- Timestamps
- Output artifacts when work completes or pauses for input

- Optional history when requested

Usage Guidance

For most client integrations, here's the recommended usage pattern:

1. Use registry search when discovery is needed.
2. Fetch the agent card for the selected agent.
3. Use `message/send` for standard asynchronous invocation.
4. Use `message/stream` when live progress updates are preferred.
5. Use `tasks/get` for polling-based status retrieval.

Error Handling and Client Expectations

Clients must handle the following classes of errors:

- Authentication failures
- Workflow-not-found cases
- Invalid method usage
- Task lookup failures
- Authorization failures during processing or status retrieval

Clients must also assume that invalid or incomplete requests might fail when validating the request. Task retrieval might fail if the task doesn't belong to the specified agent or accessible user context.

Scope and Constraints

The API is intentionally narrower than the broader A2A protocol. Here are some of the important constraints:

- The API is workflow-backed
- The primary response model is task-oriented
- Streaming is snapshot-based over SSE
- Discovery is implementation-specific
- Richer nontext and multipart input handling is limited
- The broader A2A feature set is only partially implemented

17 AI Agent Studio FAQs

Which large language models (LLMs) are currently used or supported for AI agents?

Oracle Fusion Cloud Applications support several LLMs from providers such as, Cohere, Meta, and OpenAI, for powering AI features including content authoring, summarization, and interactive Q&A. For the most current list of supported models, see [What Large Language Models \(LLM\) are in use or supported by Fusion SaaS Applications? \(KB796229\)](#).

Which scheduled processes do I need to run to make sure the features of AI Agent Studio work as expected?

Here are the scheduled processes required to ensure AI Agent Studio features work as expected.

Feature	Scheduled Process	Purpose	More Details
Access Requirements for AI Agent Studio	Import Resource Application Security Data	Imports resources and data from LDAP, and transfers the necessary information into the security tables of Fusion Applications.	Access Requirements for AI Agent Studio
	Import User and Role Application Security Data		
AI Agent Studio Assistant	Index AI Agent Studio Assistant Objects and Attributes	Indexes resources like workflows, agents, tools, business objects, and topics so you can search and receive AI-powered suggestions for relevant components to use in your agents and apps.	Access Requirements for AI Agent Studio
Monitor and Evaluate AI Agents	Aggregate AI Agent Usage and Metrics	Aggregates the metrics that are displayed in the Monitoring and Evaluation tab of AI Agent Studio.	Monitor and Evaluate AI Agents

Can I use any agent in my workflow?

No, you can't use just any agent in your workflow. You can use predefined agents and reuse agents created from the Agents tab in AI Agent Studio in other agent teams and workflows. However, if you create an agent directly within a workflow, that agent can't be reused and will only be available in the specific workflow where it was created.

Can I migrate any workflow or supervisor agent from one environment to another?

Yes. As an administrator, you can migrate (import) any individual workflow or supervisor agent from one environment to another. However, if you've imported a workflow or supervisor agent that belongs to a product family you don't have access to, you can't view the imported workflow or supervisor agent in the new environment. For example, you're an Oracle Fusion Cloud Human Capital Management (HCM) administrator and you import a workflow or supervisor agent that belongs to Oracle Fusion Cloud Supply Chain & Manufacturing (SCM). If your user doesn't have access to SCM product family in AI Agent Studio, you won't be able to view the imported workflow or supervisor agent in the target environment.

For more information, see these topics:

- [Migrate Agentic Applications or Workflows from One Instance to Another](#)
- [Migrate a Supervisor Agent from One Instance to Another](#)

How do I display a welcome message in a chatbot agent?

You can configure your welcome message and any interactive starter questions for your chatbot agent in AI Agent Studio. These messages are displayed during the user's first interaction. You can also define follow-up questions, which are perfect for keeping the conversation going by suggesting logical next steps after an initial inquiry is resolved.

These settings can be configured while creating a workflow or editing it later.

- If you're creating a new workflow, add the workflow from the **Workflows**  tab. Select **Settings**  and then go to the **Questions** subtab to configure the welcome message. For more information about the workflow settings, see [Create Workflows](#).
- If you're editing an existing workflow, go to the **Workflows**  tab and select **Edit**  to modify the workflow. Select **Settings**  and then go to the **Questions** subtab to configure the welcome message.

Note: If you see **View**  instead of **Edit** , it indicates that the workflow is either published or a preconfigured one, which can't be edited or directly modified. For more information, see [Edit a Workflow](#).

How do I make agents respond faster?

AI agent response time is directly linked to the combined length of input and output text, measured in tokens for each transaction. For example, 100 tokens is roughly 75 words, though this ratio varies by model, writing style, and language. By being deliberate with your instructions, you ensure that your agents reason clearly and avoid unnecessary processing.

Here are some recommendations to help your agents respond faster.

- Remove the summarization prompt when complex post-processing isn't required. For simple post-processing, include the instructions directly in the user prompt instead of using a separate summarization prompt. This approach reduces the number of LLM processing steps and improves response time.
- Minimize the use of input and output tokens with concise prompts. Avoid passing the entire output from upstream nodes to downstream nodes. Instead, use an output schema to pass only the data required for the next step.
- Specify response length to avoid overly verbose answers.
- Choose the smallest model that meets your requirements. Use lightweight models for routine tasks and larger models only when additional reasoning or higher quality responses are needed.
- Configure reasoning effort based on task complexity. Use Minimal or Low reasoning effort for routing, extraction, and simple question answering. Reserve higher reasoning effort for complex analysis, as higher settings can increase response time and token usage.
- Use Code nodes for deterministic tasks with predictable outcomes, such as filtering structured data, validating inputs, performing calculations, formatting outputs, and parsing or transforming structured data such as JSON. Use LLM nodes for tasks that require language understanding, interpretation, summarization, classification, extraction from unstructured content, or reasoning.
- Use smaller, specialized agents or workflow nodes to handle distinct tasks, such as research, sending communications, or Q&A. Where possible, execute independent tasks in parallel instead of building long sequential workflows.

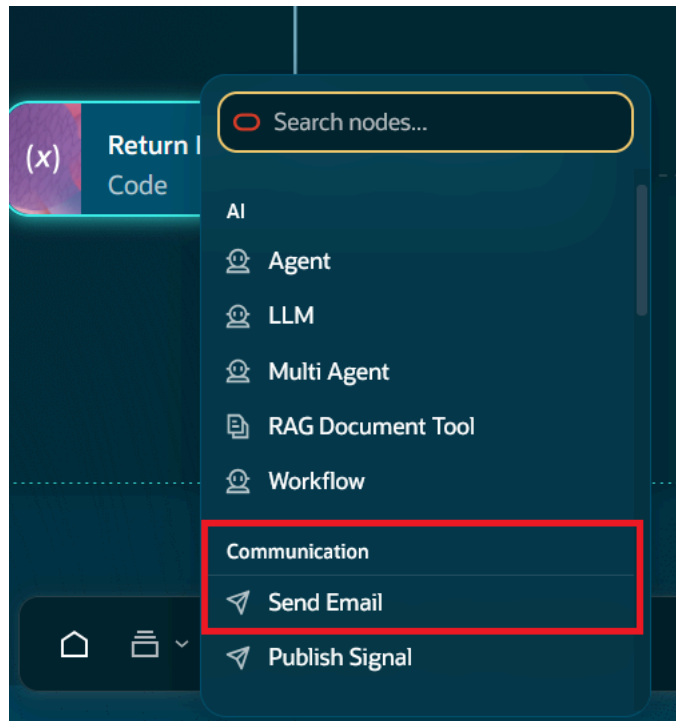
By carefully selecting relevant context and breaking complex tasks into smaller, focused agents, you can improve both the speed and efficiency of your AI agents.

How do I send an email using AI Agent Studio?

You can send emails using Workflows in AI Agent Studio.

- To enable the email functionality in a workflow, configure an email tool. For more information, see [Add Email Tool](#).

- To send emails from a workflow, use the **Send Email** node. You can add the node by dragging a connecting line from the last node and then selecting **Send Email** from the Communication section. For more information, see



Configure a Send Email Node in a Workflow.

How do I track the usage of AI Agents created in AI Agent Studio?

You can use the Monitoring and Evaluation tab within the AI Agent Studio to track real-time interactions, analyze usage patterns, and identify errors to improve agent accuracy and performance. For more information, see [Monitor and Evaluate AI Agents](#).

How can I best reduce the risk of my agent showing hallucinations?

Large language models (LLMs) can unintentionally introduce extra details when they lack sufficient information.

To mitigate this issue, consider these strategies:

- Validate your agent before deployment: Use Monitoring and Evaluation tab to evaluate your agent against representative user scenarios and edge cases. Review the evaluation results to identify hallucinations, grounding issues, and other quality gaps, then refine your prompts and knowledge sources before promoting your agent to production.

- Supply the necessary knowledge: Integrate relevant sources such as business objects, RAG documents, external and internal sources added through APIs or Connectors, ensuring the LLM has access to accurate and up-to-date information.
- Use explicit prompt instructions: Clearly instruct the LLM not to add or infer information beyond what's provided, reducing the risk of embellishment or fabrication.

How do I hide or restrict access to AI Agent Studio for specific users?

To restrict a user's access to AI Agent Studio, remove any related custom roles from the user's account.

1. Go to the Security Console, open the Users tab and search for the specific user.
2. From the search results, select the user, and edit the user account details.
3. In the Roles section of the Edit User Account page, select the custom roles associated with AI Agent Studio, and remove them using the **Delete**  icon. These roles were added to the user account when access was provided. For a list of these roles, see [Provide Access to Configure AI Agents in all Products](#).
4. Save your changes.

How can I provide access to an AI Agent for all employees?

To enable access to an AI agent for all employees, create or edit a role that includes the required privileges to access that AI agent, and assign that role to a common employee job role.

1. Go to the Security Console.
2. To use a new custom role, create it. To use an existing custom role, search for the custom role and edit it.
Note: Make sure that permission groups are enabled for the custom role.
3. Go to the Function Security Policies page and select **Add Function Security Policy**.
4. Add the Access Intelligent Agent Chat (HRC_ACCESS_AI_AGENT_CHAT_PRIV) privilege to the role and save it.
5. Go to the Role Hierarchy page and open the **Roles and Permission Groups** tab.
6. Add the FAI GenAI Agent Runtime Duty (ORA_DR_FAI_GENERATIVE_AI_AGENT_RUNTIME_DUTY) role.
7. Save the custom role and assign this custom role to the appropriate job roles.

All employees who have been assigned the job role can now access the AI agent.

Why can't I find the standard Employee role in the Security tab of Workflow Settings?

The standard Employee role isn't available in the Security tab because it's a predefined role and can't be modified to add new privileges or roles. Although permission groups can be enabled for predefined roles, you can't edit those roles to add additional privileges.

If you see an Employee-like role available in the Security tab, it's typically a custom role created by copying the predefined Employee role. Predefined roles generally have names that begin with the ORA_ prefix.

Can I rename the AI chat label to something other than Ask Oracle, or change the design of the chat panel?

No, you can't change the background image or rename 'Ask Oracle' within the active chat window or AI Agent Studio preview.

What's the difference between editing a topic versus editing a system prompt?

When you update a topic, the changes are available across agents because you can reuse topics across agents. When you're updating a system prompt, the changes are available only for the agent being edited.

Tip: We recommend updating topics instead of prompts, so that you can reuse the topics.

What file types, limits, and processing capabilities are supported in AI Agent Studio?

AI Agent Studio provides several options for processing files, including chat uploads, Document Tool ingestion, and background processing with Document Processor.

You can upload these types of files:

- Documents: PDF, DOCX, PPTX
- Text and structured text: TXT, HTML, MARKDOWN, JSON, XML
- Spreadsheets and tabular data: XLSX, CSV

- Images: JPG, PNG
- Multimedia
- Archives: ZIP

Note: ZIP files are supported only when using RAG Document Tool, and multimedia files aren't supported for RAG Document Tool.

This table outlines the recommended usage patterns for chat uploads, Document Tool, and the Document Processor. By understanding these capabilities, limitations, and best practices, you can optimize file processing performance and ensure reliable handling of uploaded content across different processing flows.

Recommended patterns

Goal	Option to Use	Best suited for	Key Limitations	Can you choose LLM?
Work with a file during a chat conversation	Chat file upload	Quick, one-time tasks such as summarizing a document, extracting a few values, reviewing an image, or analyzing a small set of files.	A maximum of five files can be uploaded per conversation, with a combined size limit of 50 MB across all uploaded files. Additionally, each individual multimedia file must not exceed 25 MB.	No, the default LLMs are used for processing.
Build reusable knowledge for retrieval	Document Tool	Guides, FAQs, handbooks, procedures, and other reference content that should be searchable and reusable by the agent over time.	A single file size limit of 25 MB. It's recommended to avoid adding more than 20 files to a single Document in the Document Tool. Instead, distribute the files across multiple Documents within the same Document Tool. For example, if you have 50 files, create three separate Documents within the same Document Tool instead of a single Document containing all 50 files. You can distribute the files as follows: • Document 1: 20 files • Document 2: 20 files • Document 3: 10 files	No, the default LLMs are used for processing.
Process business object attachments in the background	Document Processor	Asynchronous processing of files attached to business objects, automated ingestion, workflow-driven document handling,	A single file size limit of 25 MB.	• No, LLM can only be configured when LLM-based data extraction with Document Schema is enabled.

Goal	Option to Use	Best suited for	Key Limitations	Can you choose LLM?
		scanned PDFs, image-heavy PDFs, long DOCX files, and dense text files.		When it isn't enabled, the default LLMs are used for image and multimedia processing.
Extract structured fields from files	Document Processor with Document Schema	Deterministic field extraction, structured output, and cases where only specific values need to be extracted from a document.	- A Document Schema is required. - A single file size limit of 25 MB.	- Yes, when LLM-based data extraction with Document Schema is enabled. - For more information, see Add a Document Schema to a Document Processing Node of Workflows.

Best Practices

Follow these best practices to improve reliability, reduce processing failures, and choose the right file-handling approach for your use case.

- Keep files small and focused
Large files can fail even when they're within the recommended size limits. File size isn't the only constraint; extracted text, embedded images, OCR output, tables, and token expansion can also affect processing. Split large or dense files into smaller documents, especially long reports, large spreadsheets, scanned PDFs, and image-heavy PDFs.
- Use clear, readable source files
Fully scanned PDFs and image-heavy PDFs can be harder to process. Use text-based PDFs when possible. For scanned documents, make sure pages are upright, readable, high contrast, and not blurry or skewed.
- Use chat uploads for lightweight tasks
Use chat uploads for short, one-time tasks such as summarizing a small document, extracting a few values, reading an image, or reviewing a small set of attachments. Use another processing option when the content must be reused, searched, or processed as part of a background workflow.
- Use Document Tool for reusable knowledge
Use Document Tool when content should be searchable and reusable over time, such as guides, FAQs, handbooks, procedures, and reference documentation. Keep source documents focused by topic so retrieval returns more relevant results.
- Use Document Processor for background processing and extraction
Use Document Processor when files are attached to business objects, need asynchronous processing, or require structured extraction. For field extraction, provide a Document Schema and clear instructions so the processor returns only the required values.
- Avoid sending full extracted content to the LLM
Extracted content can be much larger than the original file and may increase cost, slow processing, or exceed model context limits. Send only the relevant sections, fields, pages, or chunks needed for the task. For large files, use retrieval, filtering, summarization, or a document schema.

- Simplify structured files before upload

Large spreadsheets, CSV, JSON, and XML files can contain hidden, unused, or deeply nested content that increases processing complexity. Remove unused columns, empty rows, hidden sheets, unnecessary formulas, formatting, and irrelevant records before processing.
- Review archive contents before upload

ZIP file processing depends on the files inside the archive. Include only supported and relevant files, and avoid unnecessary nested folders, duplicate files, or archives that expand into very large content.

Estimated Content Capacity

The following estimates provide a general guideline for the amount of content that can typically be processed within a 25 MB file size limit. Actual capacity may vary depending on factors such as document complexity, formatting, embedded media, and file structure.

Estimators

File Type	Estimated Capacity for a 25 MB File
PDF, text-based	300 to 1000+ pages.
PDF, image-heavy or scanned	50 to 200 pages.
DOCX	100 to 500+ pages.
PPTX	50 to 150 slides.
TXT / Markdown / HTML	5 to 15 million words.
CSV / JSON / XML	Highly variable and best kept smaller if dense or deeply structured.
XLSX	Tens of thousands to hundreds of thousands of cells, depending on formulas, formatting, and sheets.
Images	10 to 100 images, depending on resolution and compression.
Multimedia	About 15 to 60 minutes, depending on format and bit rate.
ZIP archives	Depends on the files inside and extracted content still needs to be processed.

Why do I see missing or misaligned structured document data in AI Agent Studio?

When processing structured documents in AI Agent Studio, some data may be missing or misaligned in the extracted output. This issue commonly occurs with structured PDFs containing multi-page tables, repeated headers or footers, line-level details, quantities, amounts, totals, or similar tabular data. It can occur when these PDFs are uploaded through

chat, processed by the document tool, or used as business object attachments. Although the documents might process successfully, the extracted data may not fully preserve the original document structure.

Resolution

Upgrade to the AI Agent Studio configurations or subscriptions included in 26B Cumulative Weekly Bundle (CWB) dated May 29, 2026 or later to benefit from the enhanced structured document parsing capabilities.

For more information, see *Fusion AI Agent Studio: Structured Document Data May Be Missing or Misaligned when Processed in AI Agent Studio (KB919394)*.

Why does the external REST tool only support HTTPS transactions?

The tool is limited to only HTTPS to maintain data privacy in transit.

Why is the Provider list in the LLM subtab empty?

If the Provider list is empty when creating a workflow, it's likely due to missing permissions. To resolve this, make sure the required permission groups with the **AllRowsAllFields** security view are added to the custom role assigned to the user, and run the necessary scheduled processes to synchronize security data. For more information, see *LLM Provider LOV Empty when Creating a new Agent in Ai Agent Studio (KB863374)*.

Why is my agent not fetching data from the document added to it?

Check if you've marked your document as published. If not, publish your document. To do so, go to the **Resources**  tab in AI Agent Studio, and open the **Tools** subtab. Edit the document tool and set the status of your document as **Ready to Publish**. Changing the status initiates the **Process Agent Documents** scheduled process. After the scheduled process completes, your document status is automatically changed to **Published**.

If your document status hasn't changed to **Published** even after five minutes, that means the scheduled process might have failed to trigger or complete. To resolve, do these steps:

1. Make sure your user has the **oraCommonFusionAISecurityAuthorization_create_Rag_OraResource** permission assigned.
2. From the Tools tab in AI Agent Studio, edit your document tool and change the status of your document from **Ready to Publish** to **Draft**.
3. Save the document tool.
4. Now, again change the status of your document back to **Ready to Publish**.

5. Save again to trigger the scheduled process.

For information about creating documents to be uploaded to the Document tool, see [Create Documents to Add to Document Tool](#).

Why can't I see the imported workflows or supervisor agents in the target environment?

You might not have access to the product family the imported workflow or supervisor agent belongs to. For more information, see [Provide Access to Configure AI Agents in all Products](#).

How can I move the AI Agents link to the top of the Quick Actions list?

For information about reordering links in Quick Actions, see [Reorder Navigation Entries](#).

Why is the Roles list in the Security tab empty?

That's probably because permission groups aren't enabled for your duty role. Security administrators can enable permission groups for the duty role assigned to you, in Security Console. For details, see [Access Requirements for AI Agent Studio](#).

How can AI agents be configured or instructed to respond in a user's preferred language?

Here's how you can enable responses to be returned in the user's preferred language:

- You can specify the response language preference at the time of setting up your workflow, or when adding individual LLM and agent nodes to the workflow.
 - To set up the response language preference at the time of setting up your workflow, do these steps:
 - i. Go to **AI Agent Studio**.
 - ii. Select the **Workflows**  tab.
 - iii. Select **Edit**  to modify the workflow.
 - iv. Select **Settings**  and open the **Chat Experience** tab.

- v. Enable the **Answer in user's language** option.

You can also update the language instructions in the respective prompt. If the language prompt isn't updated, the default language prompt is automatically applied. The default prompt is "You must respond only in \$context.\$system.\$responseLanguage. Do not use any other language."

- o To set up the response language preference for individual nodes, do these steps:
 - i. Go to **AI Agent Studio**.
 - ii. Select **Edit**  to modify the workflow.
 - iii. Select the nodes of type LLM or Agent, and enable the **Answer in user's language** option.
- You can specify to get the responses in your preferred language using natural language instructions when interacting with the agent.

Why is my agent not responding in my preferred language?

If your agent isn't responding in the preferred language, verify these steps:

- In your agent's settings, go to the **Chat Experience** tab and make sure that the **Answer in user's language** option is enabled.
- Check that all LLM and agent nodes added in the workflow have the **Answer in user's language** option enabled.

For more information, see [How can AI agents be configured or instructed to respond in a user's preferred language?](#)