Microsoft Certified: Agentic AI Business Solutions Architect Expert

# AB-100

## Agentic AI Business Solutions Architect

A detailed study guide for architects who design, deploy, and govern agentic-first business solutions across Microsoft 365 Copilot, Copilot Studio, Microsoft Foundry, Power Platform, and Dynamics 365. Built from the October 14, 2026 skills outline and Microsoft Learn.

| | | |
|---|---|---|
| **25–30%** | **25–30%** | **40–45%** |
| Plan AI-powered business solutions | Design AI-powered business solutions | Deploy AI-powered business solutions |

Edition: October 2026. Independent study material. Practice items are original and are not real exam questions. Product names, licensing, and preview status change frequently: confirm details against Microsoft Learn before exam day.

# Contents


**START HERE** — **The AB-100 exam and how to use this guide**

0.1 Who the exam is for
0.2 Skills measured and weighting
0.3 Exam mechanics
0.4 How this guide is organised
0.5 Five-week study plan

**FOUNDATION** — **The Microsoft agentic platform landscape**

F.1 The platforms
F.2 Agent types you must distinguish
F.3 Cross-cutting building blocks

**DOMAIN 1 · PART A** — **Requirements and AI strategy**

1.1 Analyze requirements for AI-powered business solutions
1.2 Design overall AI strategy for business solutions (part 1)
1.3 Additional practice questions

**DOMAIN 1 · PART B** — **Custom AI, prompts, Center of Excellence, and ROI**

2.1 Custom AI models and small language models
2.2 Prompt engineering and prompt libraries
2.3 The Microsoft AI Center of Excellence
2.4 AI solutions that span multiple Dynamics 365 apps
2.5 Selecting ROI criteria, including total cost of ownership
2.6 Creating an ROI analysis for a business process
2.7 Build, buy, or extend AI components
2.8 Implementing a model router
2.9 Key terms
2.10 Additional practice questions

**DOMAIN 2 · PART A** — **Design AI and agents for business solutions**

3.1 Copilot in Dynamics 365 customer experience and service apps
3.2 Agent types: prompt and response, task, and autonomous
3.3 Copilot Studio conversation and automation design
3.4 Data, Foundry Tools, and intelligent apps
3.5 Additional practice questions

**DOMAIN 2 · PART B** — **Extensibility and prebuilt agents and apps**

4.1 Design extensibility of AI solutions
4.2 Orchestrate configuration for prebuilt agents and apps
4.3 Additional practice questions

**DOMAIN 3 · PART A** — **Monitor, tune, and test AI solutions**

5.1 Analyze, monitor, and tune AI-powered business solutions
5.2 Manage the testing of AI-powered business solutions
5.3 Additional practice questions

**DOMAIN 3 · PART B** — **Application lifecycle management for AI solutions**

6.1 A cohesive ALM strategy for agentic-first solutions
6.2 A cohesive environment strategy for AI-powered solutions
6.3 Design the ALM process for data used in AI models and agents
6.4 Design the ALM process for Copilot Studio agents, connectors, and actions
6.5 Design the ALM process for Microsoft Foundry Agent Service
6.6 Design the ALM process for custom AI models
6.7 Design the ALM process for AI in Dynamics 365 apps for finance and supply chain
6.8 Design the ALM process for AI in Dynamics 365 apps for customer experience and service
6.9 Key terms
6.10 Additional practice questions

**DOMAIN 3 · PART C** — **Responsible AI, security, governance, risk, and compliance**

7.1 Design security for agents
7.2 Design governance for agents
7.3 Design model security
7.4 Analyse solution and AI vulnerabilities and mitigations, including prompt manipulation
7.5 Review the solution for adherence to responsible AI principles
7.6 Validate data residency and movement compliance
7.7 Design access controls on grounding data and model tuning
7.8 Design audit trails for changes to models and data
7.9 Additional practice questions

**CROSS-DOMAIN PRACTICE** — **Case study practice**

C.1 A method for architecture case studies
C.2 Case study: Contoso Industrial

**EXAM DAY** — **Exam technique and signal words**

E.1 Signal words that map to answers
E.2 Distractor patterns
E.3 Time management

**APPENDIX A** — **One-page cheat sheet**

**APPENDIX B** — **Glossary**

**APPENDIX C** — **Hands-on labs**

**APPENDIX D** — **Final readiness checklist**

**APPENDIX E** — **Official resources**

START HERE

# The AB-100 exam and how to use this guide

**Orientation, exam mechanics, and a study plan**

AB-100 is an expert-level architecture exam. It does not ask you to write code; it asks you to choose. Which platform should host this agent? Should we extend Microsoft 365 Copilot or build a custom agent? Is this a task for a prebuilt Dynamics 365 agent, a Copilot Studio autonomous agent, or a Foundry agent with a fine-tuned model? How do we prove ROI, test it, ship it through environments, monitor it, and keep it secure and compliant? Every item is a design decision with trade-offs.

| 700 | Expert | 40–45% | Annual |
|---|---|---|---|
| Passing score on a 1–1000 scaled score | Requires AB-100 plus one associate prerequisite | Weight of the Deploy domain | Free online renewal assessment |

## 0.1 Who the exam is for

The candidate is an accomplished solution architect who designs and delivers AI-driven business solutions that transform business processes. The profile emphasises agentic-first and multi-agent design, secure and scalable cross-platform solutions, comprehensive knowledge of Dynamics 365, Power Platform, Copilot Studio, Foundry Tools, and Foundry Models, open protocols (Agent2Agent and Model Context Protocol), the Microsoft Responsible AI Standard, securing models and data workflows, interpreting telemetry, and ROI analysis.

Key responsibilities include defining architecture strategy and the roadmap for agentic-first business processes, prototyping AI components, guiding end-to-end implementation, championing adoption, and creating cohesive **ALM** and **environment** strategies, including third-party AI.

### Certification requirements

To earn the Expert certification you must pass AB-100 **and** hold at least one prerequisite associate certification. The list includes, among others, PL-400 (Power Platform Developer), MB-230 (Customer Service), MB-310 (Finance), MB-330 (Supply Chain Management), MB-500, MB-800, MB-820, AI-103 (Azure AI Apps and Agents Developer), AI-300, AB-620 (AI Agent Builder), AB-210 (Dynamics 365 Sales AI Consultant), AB-410 (Intelligent Applications Builder), and AB-250 (Dynamics 365 Contact Center AI Engineer).

## 0.2 Skills measured and weighting

| Domain and skill groups | Weight | Relative emphasis | Chapter |
|---|---|---|---|
| **Plan AI-powered business solutions**<br>Analyse requirements; AI strategy (part 1) | 25–30% | | Chapter 1 |
| AI strategy (part 2); costs and benefits | | | Chapter 2 |
| **Design AI-powered business solutions**<br>Design AI and agents | 25–30% | | Chapter 3 |
| Extensibility; prebuilt agents and apps | | | Chapter 4 |
| **Deploy AI-powered business solutions**<br>Monitor, tune, and test | 40–45% | | Chapter 5 |
| ALM | | | Chapter 6 |
| Responsible AI, security, governance, risk, compliance | | | Chapter 7 |

**EXAM TIP**

The Deploy domain alone is worth up to 45%. Chapters 5 to 7 (monitoring, testing, ALM, and security and governance) deserve as much study time as the first four chapters combined.

## 0.3 Exam mechanics

| Item | What to expect |
|---|---|
| Scoring | Scaled 1 to 1000; 700 or greater passes. No penalty for wrong answers. |
| Case studies | Expect at least one multi-tab case study. You cannot return after leaving it. |
| Preview features | Mostly generally available features; commonly used preview features may appear. Many agent capabilities are in preview, so learn their status. |
| Localisation | English updated first; others about eight weeks later. Request 30 extra minutes if your language is unavailable. |
| Practice | Free Practice Assessment on AI Skills Navigator; exam sandbox at aka.ms/examdemo. |
| Renewal | Annual, through a free online assessment on Microsoft Learn. |

### Question formats

| Format | Technique |
|---|---|
| Multiple choice | Identify the decisive constraint (licensing, data location, autonomy, channel, latency) and eliminate options that violate it. |
| Multiple response | Check whether options are parts of one solution or complete alternatives. |
| Drag and drop / ordering | Common for ALM sequences, adoption phases, and matching platforms to scenarios. |

| Format | Technique |
|---|---|
| Hot area | Pick the right value per drop-down, usually contrasting pairs (managed vs unmanaged, task vs autonomous agent). |
| Case study | Number the requirements first; map each item to one requirement. |

## 0.4 How this guide is organised

- **Agentic platform orientation** maps Microsoft's agent platforms and when each is the right home for an agent.
- **Chapters 1 to 7** follow the official outline bullet by bullet: objective map, explanation, decision tables, architecture diagrams, callouts, original practice questions with answers, and key takeaways.
- **Case study practice and exam technique** combine all domains and list signal words.
- **Appendices** contain a cheat sheet, glossary, hands-on labs, a readiness checklist, and official resources.

**EXAM TIP**
Patterns that frequently decide an answer.

**COMMON TRAP**
Plausible distractors and misconceptions.

**PRODUCTION NOTE**
What changes in real enterprise programmes.

## 0.5 Five-week study plan

| Week | Read | Hands-on | Checkpoint |
|---|---|---|---|
| 1 | Orientation; Chapters 1 and 2 | Map one business process to agent opportunities; compare a Copilot Studio agent and a Microsoft 365 declarative agent for the same scenario; draft an ROI model. | Chapters 1 and 2 practice at 80%. |
| 2 | Chapter 3 | Build task, autonomous (trigger-based), and prompt-and-response agents in Copilot Studio; design topics with fallback; add a prompt action and an agent flow. | Chapter 3 practice at 80%. |
| 3 | Chapter 4 | Add an MCP server and a connector to a Copilot Studio agent; connect a Foundry agent; review Copilot features in a Dynamics 365 trial; explore AI hub. | Chapter 4 practice at 80%. |
| 4 | Chapters 5 and 6 | Enable Copilot Studio analytics and Application Insights; create test sets; move an agent through dev, test, and prod with solutions and pipelines. | Chapters 5 and 6 practice at 80%. |

| Week | Read | Hands-on | Checkpoint |
|---|---|---|---|
| 5 | Chapter 7; Case study; Appendices | Apply DLP and managed environment rules; review Purview audit for agent activity; configure Prompt Shields in Foundry. Take the free Practice Assessment. | 85% or better. |

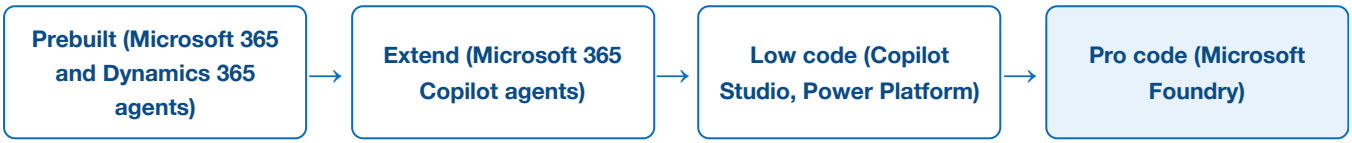
FOUNDATION

# The Microsoft agentic platform landscape

**Where an agent should live, and why**

Most AB-100 decisions start with one question: which platform is the right home for this capability? This chapter gives the map; Chapters 1, 3, and 4 go deep.

## F.1 The platforms




| Platform | Best fit | Typical signals in a question |
|---|---|---|
| **Prebuilt agents and Copilot features in Microsoft 365 and Dynamics 365** | Standard business scenarios already covered by Microsoft (sales, service, finance, supply chain, productivity) | "Fastest time to value", "minimal customisation", "use built-in capabilities" |
| **Microsoft 365 Copilot agents (declarative agents, extensions)** | Employee-facing assistants inside Microsoft 365 that reuse Copilot's orchestrator, security, and Microsoft Graph grounding | "Users work in Teams and Outlook", "use Microsoft 365 data with existing permissions" |
| **Copilot Studio** | Low-code agents with topics, knowledge, actions, agent flows, triggers for autonomous behaviour, many channels including Dynamics 365 Contact Center | "Makers", "connectors", "multiple channels", "autonomous on events", "governed by Power Platform admin" |
| **Power Platform (Power Apps, Power Automate, AI Builder, Dataverse)** | Business apps and processes with embedded AI, prompts, and data in Dataverse | "Canvas app", "AI Builder model", "Dataverse", "AI hub" |
| **Microsoft Foundry** | Pro-code agents and models: model choice, fine-tuning, custom models, evaluations, advanced orchestration, Foundry Tools | "Custom model", "fine-tune", "specific model", "complex orchestration", "developers" |

**COMMON TRAP**

Do not jump to Foundry because it is the most capable. Expert architects choose the **least complex platform that meets the requirement**: prebuilt, then extend, then low code, then pro code. Move right only when a requirement forces it (custom model, specific model control, deep orchestration, or unsupported scale).

## F.2 Agent types you must distinguish

| Type | Behaviour |
|---|---|
| Prompt and response agent | Answers questions and generates content when a user asks; no actions beyond the conversation. |
| Task agent | Completes defined tasks for a user using actions, tools, and flows, usually with the user in the conversation. |
| Autonomous agent | Runs on triggers or schedules without a user prompt, plans and acts within defined guardrails, and escalates when needed. |
| Multi-agent solution | Several agents coordinate: an orchestrating agent delegates to child or connected agents, possibly across platforms through Agent2Agent (A2A) or tools exposed through Model Context Protocol (MCP). |

## F.3 Cross-cutting building blocks

| Building block | Role in agentic solutions |
|---|---|
| Dataverse | Business data, security model, and solution container for apps, agents, and flows. |
| Microsoft Graph and Copilot connectors | Organisation content and external data indexed for Microsoft 365 Copilot with permissions respected. |
| Microsoft Fabric and OneLake | Analytical data made available to AI systems and agents. |
| Connectors, agent flows, MCP, A2A | How agents call systems and each other. |
| Solutions, environments, pipelines | ALM for Power Platform, Copilot Studio, and Dynamics 365 customisations. |
| Managed environments, DLP, Purview, Entra | Governance, data protection, identity, and audit. |
| Foundry Tools and Foundry Models | Prebuilt AI services and the model catalog behind pro-code solutions. |

### Microsoft Learn references

- Study guide for Exam AB-100
- Microsoft Certified: Agentic AI Business Solutions Architect Expert
- Microsoft Copilot Studio documentation
- Microsoft Foundry documentation

DOMAIN 1 · PART A

# Requirements and AI strategy

**Part of Domain 1: Plan AI-powered business solutions (25–30%)**

Domain 1 is where the exam checks whether you think like an architect before you think like a builder. The questions rarely ask how to click through a designer. Instead they describe an organisation, a business process, some data, a licensing position and a set of constraints, then ask which platform, which pattern or which governance decision is the best fit. This chapter covers the first half of the domain: analysing requirements (where agents add value, whether the grounding data is good enough, and how to expose business data to AI systems) and the first group of AI strategy skills (the Cloud Adoption Framework AI adoption process, platform strategy, multi-agent design, prebuilt agents, solution rules and constraints, Copilot Studio knowledge, and the extend versus build decision for Microsoft 365 Copilot).

Study it in three passes. First, learn the decision frameworks Microsoft publishes (the CAF AI decision tree, the AI agent decision tree, the single versus multi-agent criteria and the declarative versus custom engine comparison), because most scenario questions are built directly on them. Second, memorise the hard numbers that turn into distractors: Copilot Studio quotas and knowledge limits, Copilot Credit rates, the 125% overage rule, declarative agent manifest limits and Foundry quota scope. Third, practise reasoning about trade-offs: the "correct" answer is usually the option with the least operational responsibility that still meets every stated requirement.

| Official skill (Plan AI-powered business solutions) | Covered in |
|---|---|
| **Analyze requirements for AI-powered business solutions** | |
| Assess the use of agents in task automation, data analytics, and decision-making | 1.1.1 |
| Review data for grounding, including accuracy, relevance, timeliness, cleanliness, and availability | 1.1.2 |
| Organize business solution data to be available for other AI systems | 1.1.3 |
| **Design overall AI strategy for business solutions** | |
| Implement the AI adoption process from the Cloud Adoption Framework for Azure | 1.2.1 |
| Design the strategy for building AI and agents in business solutions | 1.2.2 |
| Design a multi-agent solution by using platforms such as Microsoft 365 Copilot, Copilot Studio, and Microsoft Foundry | 1.2.3 |
| Develop the use cases for prebuilt agents in the solution | 1.2.4 |
| Define the solution rules and constraints when building AI components with Copilot Studio, Microsoft Foundry and Microsoft Foundry Tools | 1.2.5 |
| Determine the use of generative AI and knowledge sources in agents built with Copilot Studio | 1.2.6 |
| Determine when to build custom agents or extend Microsoft 365 Copilot | 1.2.7 |

| Official skill (Plan AI-powered business solutions) | Covered in |
|---|---|
| Determine when custom AI models should be created; prompt library guidelines; customised small language models; prompt engineering guidelines; Microsoft AI Center of Excellence; AI solutions that use multiple Dynamics 365 apps | Domain 1 · Part B (next chapter) |
| **Evaluate the costs and benefits of an AI-powered business solution** | |
| ROI criteria and TCO; ROI analysis; build, buy, or extend; model router | Domain 1 · Part B (next chapter); cost drivers introduced in 1.2.2 and 1.2.5 |

**NOTE**

The skills outline used throughout this book is the version measured as of October 14, 2026. The change log on the study guide lists only a minor change to "Design overall AI strategy for business solutions" compared with the previous version. The study guide also states that most questions cover generally available (GA) features, but Preview features can appear if they are commonly used. This book marks Preview features with PREVIEW.

**NOTE**

Naming is moving quickly. Microsoft Foundry was formerly Azure AI Foundry, Foundry Tools were formerly Azure AI services, and Microsoft 365 Copilot connectors were formerly Microsoft Graph connectors. Some recently updated Learn pages (for example, the CAF AI adoption article and Microsoft 365 admin articles) refer to the licensed Microsoft 365 Copilot experience simply as "Microsoft Copilot". The exam outline uses "Microsoft 365 Copilot", and so does this book.

## 1.1 Analyze requirements for AI-powered business solutions

Requirements analysis for an AI solution has two halves that the exam treats separately. The first half is about the work: what the process is, how deterministic it is, how many systems it touches and how much autonomy is acceptable. The second half is about the data: whether the content the agent will rely on is trustworthy, current, clean, relevant and reachable under the right permissions, and whether it is organised so that more than one AI system can consume it without each team building its own integration.

**Agent** — A program that uses a generative AI model to interpret inputs (user messages, system events or messages from other agents), reason, and decide which knowledge and tools to use. The CAF agent guidance describes five components: the generative AI model (reasoning engine), instructions (scope and boundaries), retrieval (grounding data), actions (functions, APIs, systems) and memory (conversation history and state).

**Grounding** — Supplying trusted, domain-specific organisational data to the model at run time so that responses are based on that data rather than on the model's general knowledge.

**Retrieval-augmented generation (RAG)** — A deterministic retrieve-then-generate pattern. CAF distinguishes it from agents: an agent uses the model to decide which knowledge and tools to use at each step, which enables multi-step reasoning but introduces nondeterminism.

**Semantic index** — The Microsoft 365 index that maps content across Microsoft Graph into lexical and semantic representations for Copilot retrieval. It updates as content changes and respects

| | |
|---|---|
| | user permissions. |
| **Data product** | A governed, curated data set published by a data domain (typically in Microsoft Fabric OneLake) that AI and analytics workloads consume as a trusted input. |

### 1.1.1 Assess the use of agents in task automation, data analytics, and decision-making

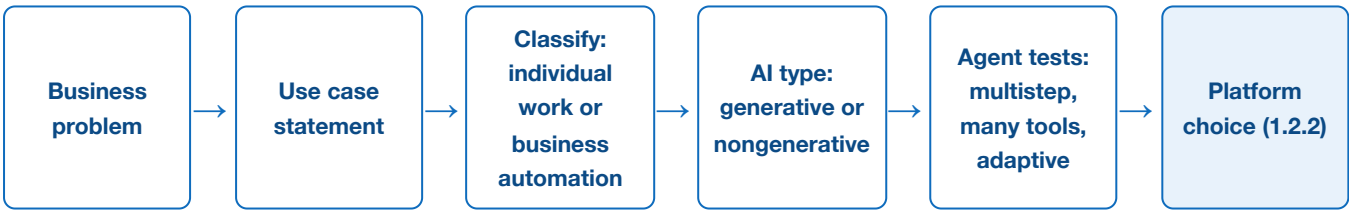
The exam expects you to separate three questions that business stakeholders often blur together: should this be AI at all, should it be an agent rather than a simpler generative pattern, and how much autonomy should the agent have. Microsoft's Cloud Adoption Framework (CAF) gives a clear sequence for the first two questions, and the AB-100 training module adds the three usage lenses named in the outline: task automation, data analytics and decision-making.

#### Step 1: Decide whether an agent is appropriate

The CAF "Business plan for AI agents" article starts with when *not* to use an agent. Two tests eliminate many candidate use cases:

- **Structured or predictable task.** If the steps are clear, repeatable and follow strict rules, use regular code or nongenerative AI models. They are faster, cheaper and more reliable for fixed workflows. Typical Microsoft options are Power Automate or code, Fabric data science, Foundry Tools prebuilt models and Azure Machine Learning.
- **Static knowledge retrieval.** If the goal is to answer questions or summarise content from a fixed document set, and no tools or multi-step reasoning are needed, a classic RAG solution is sufficient. FAQ bots and document search with summaries are the examples CAF gives.

If both answers are "no", CAF lists three signals that an agent adds real value: the task requires **multistep decisions** (read, evaluate, decide the next step, check its own work), it uses **many tools or systems** in a flexible order, or it needs **adaptive behaviour** because inputs are incomplete or ambiguous.




The CAF "Plan for AI adoption" article frames the earlier steps. Start with business problems rather than technology, translate each problem into a short use case statement (activity plus expected result), classify it as **individual work** (improving how people work inside existing tools, which aligns strongly to Microsoft 365 Copilot) or **business automation** (changing how the organisation operates, often needing integration with other systems), then form an initial view of the AI type. Generative AI is nondeterministic and suits unstructured inputs where some variation is acceptable; nongenerative AI is deterministic and suits structured inputs where the same input must produce the same result, such as prediction or anomaly detection.

### Step 2: Apply the three usage lenses

| Lens | What agents contribute | Typical Microsoft surfaces | Architect questions to ask |
|---|---|---|---|
| **Task automation** | Drafting documents and emails, summarising mail, meetings and chats, triggering multi-step processes (approvals, notifications, record updates), creating or updating records in systems of record. | Microsoft 365 Copilot; Copilot Studio agents with agent flows and connectors; Power Automate; Dynamics 365 agents such as the Case Management Agent. | Which steps are deterministic and better as a flow? Which actions change data, and do they need approval? What is the run volume (affects quotas and credits)? |
| **Data analytics** | Turning natural language questions into answers over governed data, identifying trends and outliers, generating visualisations, explaining dashboards and suggesting next steps. | Fabric data agents over lakehouses, warehouses, semantic models and KQL databases; Copilot in Power BI; the Sales Research Agent in Dynamics 365 Sales. | Is the data modelled and described well enough for query generation? Are answers read-only? Who certifies the semantic model? |
| **Decision-making** | Scenario recommendations from historical data, risk identification through pattern recognition, context summaries from documents, meetings and data sets, recommendations backed by enterprise knowledge. | Researcher in Microsoft 365 Copilot; Sales Opportunity Agent (risk visibility); Foundry agents with evaluation pipelines for higher-risk logic. | Who remains accountable for the decision? Where is the human-in-the-loop checkpoint? What evidence (citations) must accompany a recommendation? |

The AB-100 module adds five best practices that recur in scenario wording: start with the business outcome, use automation to reduce repetitive work rather than replace critical thinking, maintain the Responsible AI principles (fairness, reliability and safety, privacy and security, inclusiveness, transparency, accountability), monitor and refine prompts, workflows and data inputs, and train people to use Copilot effectively.

### Step 3: Decide the level of autonomy

CAF describes a spectrum of agent categories with increasing autonomy and governance burden:

| CAF category | Capabilities | Example | Governance implication |
|---|---|---|---|
| Productivity (retrieval) agents | Knowledge tools that retrieve and synthesise information | Internal knowledge assistant, service knowledge lookup | Focus on data access and citation quality |
| Action (task) agents | Knowledge tools plus action tools within defined workflows | Service ticket creation, record updates | Scope each action, log tool calls, approval for writes |
| Automation (autonomous) agents | Knowledge and action tools plus triggers that decide when to run, stop or escalate | Supply chain optimisation, autonomous case follow-up | Rigorous governance: escalation rules, monitoring, rollback |

**EXAM TIP**

When a scenario describes a rule-based, high-volume process with fixed inputs (for example, validating invoice totals against purchase orders using exact matching rules), the expected answer is usually *not* an agent. Look for answers that use a deterministic flow, code or a nongenerative model. Agents become the right answer when the wording includes ambiguity, judgement across several systems, or variable sequences of steps.

#### Step 4: Prioritise and set success metrics

CAF recommends scoring each candidate on a 1–5 scale across three areas:

- **Business impact:** executive strategy alignment, business value (reshape business processes, enrich employee experiences, reinvent customer engagement, accelerate innovation) and change management timeframe.
- **Technical feasibility:** implementation and operation risks, sufficient safeguards (never advance a use case with unclear safeguards) and technology fit with existing systems and data access patterns.
- **User desirability:** clearly defined personas, a convincing value proposition and low change resistance.

Validate value with a small pilot in Copilot Studio or Microsoft Foundry and, as CAF puts it, pilot the hardest steps first. Then define success metrics before development: set baseline KPIs (measure current performance where a process exists), use the metrics as go or no-go **decision gates** at each stage, and keep measuring after deployment so the solution can be refined, retired or replaced.

**PRODUCTION NOTE**

Capture non-functional requirements during this analysis, not later: expected messages per minute and per hour (Copilot Studio quotas are per Dataverse environment), peak concurrency, languages, channels, identity model (user-delegated or agent identity), data residency and the actions that need human approval. These directly determine platform choice, capacity planning and the governance controls described in 1.2.5.

### 1.1.2 Review data for grounding: accuracy, relevance, timeliness, cleanliness, availability

Agents synthesise information rather than create it, so their accuracy depends on the quality and accessibility of the sources behind them. The AB-100 module defines five dimensions of grounding data quality, and the exam will describe symptoms (wrong answers, stale answers, answers that ignore a document, answers that differ between users) and ask which dimension is failing and what to change.

| Dimension | Definition and indicators | Typical symptom | Architect controls on Microsoft platforms |
|---|---|---|---|
| **Accuracy** | Data is correct and verified: validated by subject matter experts, matched to authoritative sources, free of outdated assumptions. | Confident but wrong answers; conflicting answers from two documents. | Curate authoritative sources only; establish a "golden dataset" for testing (CAF govern guidance); require citations in agent instructions; in classic orchestration, mark highly trusted sources as official sources in Copilot Studio. |
| **Relevance** | Data aligns to the agent's intended task and domain. | Semantic search returns conceptually similar but contextually wrong content. | Scope knowledge to specific sites, folders, tables or connections; write precise knowledge source descriptions so generative orchestration selects the right source; give each subagent its own nonoverlapping knowledge. |

| Dimension | Definition and indicators | Typical symptom | Architect controls on Microsoft platforms |
|---|---|---|---|
| **Timeliness** | Data is current: modified dates, seasonal or compliance updates, refresh schedules. | Answers reflect last year's policy. | Remove superseded versions; understand refresh behaviour (the Microsoft 365 semantic index updates as content changes; Copilot Studio uploaded SharePoint and OneDrive sources synchronise every four to six hours; Copilot connectors sync periodically with admin-configurable frequency). |
| **Cleanliness** | Clear structure, no duplicates, minimal irrelevant metadata, stable formatting. | Low retrieval precision; tables or layouts parsed badly. | Consistent document templates; unique, human-readable field names and descriptions in semantic models; avoid unsupported page types (Copilot Studio does not use classic ASPX SharePoint pages or modern pages with SPFx components for answers). |
| **Availability** | Data is accessible, indexable and permitted for the user. | Agent answers for one user but not another; content shows "Ready" but never appears in answers. | Store authoritative content where it is indexed; verify permissions regularly; configure authentication correctly (for example, Dataverse knowledge requires "Authenticate with Microsoft"); check sensitivity labels and file limits. |

### Availability is a permissions question first

Microsoft 365 Copilot and Copilot Studio retrieve only what the signed-in user can access. The Microsoft 365 Copilot Retrieval API retrieves text passages from SharePoint, OneDrive and connected sources while honouring user permissions, and Copilot Studio knowledge sources such as SharePoint, Dataverse and Copilot connectors use the agent user's Microsoft Entra ID authentication. That is a security feature, but it means "the agent cannot find the document" is often an availability problem rather than a model problem. Several platform specifics are worth memorising because they produce exactly this symptom:

- In Copilot Studio uploaded SharePoint and OneDrive sources, documents labelled **Confidential** or **Highly Confidential**, or password-protected documents, can't be indexed. They show as ready but don't provide responses.
- For makers without a Microsoft 365 Copilot licence in the same tenant, generative answers can only use SharePoint files under 7 MB. With a Microsoft 365 Copilot licence in the tenant and **Tenant graph grounding with semantic search** turned on, SharePoint and connector files up to 200 MB are supported (512 MB for PDF, PPTX and DOCX).
- Dataverse knowledge requires **Dataverse search** to be enabled in the environment.
- Generative answers from SharePoint sources aren't available to guest users in SSO-enabled apps.
- Queries that depend on file names or folder metadata (for example, "how many files are in this folder") are not supported for SharePoint knowledge.

**COMMON TRAP**

Do not "fix" an availability problem by widening permissions on the source or by switching to a shared service identity just so the agent can see more content. That breaks least privilege and the permission trimming that makes grounded answers safe. The correct fixes are usually to move authoritative content to an indexed, correctly permissioned location, correct the authentication method, or adjust labels and file formats.

### A grounding data review checklist

1. Inventory each knowledge domain the agent needs and name the authoritative owner (CAF calls for documenting data sources, formats, quality and accessibility).
2. Remove or archive outdated and conflicting content before connecting it (the module's first best practice).
3. Store authoritative unstructured content in SharePoint or OneDrive so it becomes part of the semantic index.
4. Standardise formatting to improve chunking and retrieval precision.
5. Review permissions and sensitivity labels against the intended audience.
6. Build a golden question set with expected answers and sources, validated by SMEs, and keep it for regression testing.
7. Record refresh expectations per source (real time, hours, daily) and confirm the platform can meet them.

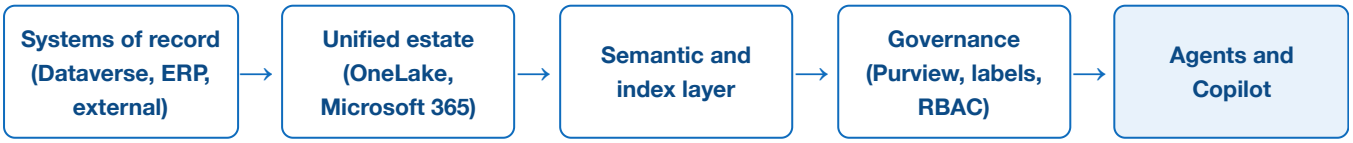
**PRODUCTION NOTE**

Copilot Studio documents a quirk worth planning for: after you add files or folders to an unstructured knowledge source, the status can show "Ready" immediately, then change to "In Progress", and only become usable when it returns to "Ready". Also, application lifecycle management (ALM) isn't supported for these unstructured sources: importing an agent into another environment doesn't automatically process its knowledge sources, so deployment runbooks must include re-adding or re-ingesting them.

## 1.1.3 Organize business solution data to be available for other AI systems

The outline wording matters: "available for *other* AI systems". The architect's job is not to wire one agent to one database, but to organise business data so that Microsoft 365 Copilot, Copilot Studio agents, Foundry agents, Fabric data agents and analytics can all consume the same governed data without each team building a private pipeline. CAF's "Data architecture for AI agents" article frames this as three moves: unify the data platform, prepare Microsoft 365 data, and define retrieval strategies for custom agents (built-in retrieval first, MCP for live data and actions), then document the retrieval decision for every data domain.

| Layer (AB-100 module) | Purpose | Microsoft examples |
|---|---|---|
| Operational databases | Store application and business data with structure | Dataverse (Dynamics 365 customer engagement apps, Power Apps), Dynamics 365 finance and operations databases, Azure SQL, Azure Cosmos DB, Azure Database for PostgreSQL |
| Analytical stores | Prepare curated data for AI and machine learning | Fabric lakehouse and warehouse in OneLake, Power BI semantic models |
| Intelligence layer | Grounding, retrieval, semantic and vector search | Microsoft 365 semantic index, Azure AI Search, Foundry IQ knowledge bases, Dataverse search |
| AI apps and agents | Consume the layers above | Microsoft 365 Copilot and declarative agents, Copilot Studio agents, Foundry agents, Fabric data agents, custom RAG apps |



### Microsoft Fabric and OneLake: the unified data platform

CAF positions Fabric OneLake as the central lake in which data domains publish governed **data products**, and states that agents consume these data products through Fabric IQ, Foundry IQ in Microsoft Foundry and Copilot Studio. Two OneLake mechanisms let you unify without copying everything:

- **Shortcuts** virtualise data in ADLS Gen2, Amazon S3, Google Cloud Storage, S3-compatible storage and Dataverse so it appears in OneLake without copying. They suit data mesh patterns where domains own their data and others discover it.
- **Mirroring** provides a query-optimised replica of operational databases where you need one.

For Dynamics 365 and Power Apps, **Link to Microsoft Fabric** makes Dataverse data available in OneLake with no export, no ETL pipelines and no partner tools: shortcuts point from OneLake into Dataverse, the data stays in Dataverse, and the link can include data from all Dynamics 365 apps, including finance and operations apps. Dataverse shortcuts are read-only regardless of the user's permissions.

### Fabric data agents and semantic models

A Fabric data agent is a configurable, standalone conversational Q&A artifact over OneLake data. It supports up to five data sources in any combination of lakehouses, warehouses, Power BI semantic models, KQL databases, ontologies and Microsoft Graph, and returns read-only answers without users writing SQL, DAX or KQL. Fabric data agents can be consumed from Copilot Studio (as a tool, or as a connected agent PREVIEW), from Foundry, and in Microsoft 365 Copilot PREVIEW. Prerequisites include a paid F2 or higher Fabric capacity (or Power BI Premium P1 or higher with Fabric enabled) and enabling the cross-geo processing and storing tenant settings for AI. Learn warns that when responses are consumed in Copilot Studio or Microsoft 365 Copilot, they might be sent outside Fabric's compliance boundary or geographic region, which is a data residency consideration you must raise during design.

When a data agent queries a semantic model, it generates DAX from the model's metadata. Learn's semantic model guidance is explicit that the DAX generation tool relies solely on the model metadata and **Prep data for AI** configuration, and ignores instructions added at the data agent level for DAX generation. Prep data for AI PREVIEW has three components, which also improve Copilot in Power BI:

- **AI data schema:** simplify the schema Copilot can use (remove unused objects, avoid duplicate field names across tables).
- **Verified answers:** map trigger phrases to an approved visual so common questions return a governed answer.
- **AI instructions:** business context and terminology for the model.

General modelling hygiene still applies: field descriptions, hidden technical columns, human-readable names and an optimised model, because a bloated model creates noise for DAX generation and slows conversational responses.

> **EXAM TIP**
> If a scenario says "the data agent returns wrong figures from the semantic model even though we added instructions to the data agent", the answer is to configure Prep data for AI on the semantic model (AI data schema, verified answers, AI instructions) and improve model metadata, not to keep editing data agent instructions.

### Dataverse: the system of record for business applications

Dataverse is both a source of grounding and a target for actions. For grounding, Copilot Studio can add Dataverse tables as knowledge. The constraints that appear in questions are: the environment must have Dataverse search turned on; the agent must use "Authenticate with Microsoft"; each knowledge source supports up to 15 tables; the Copilot Studio limits page lists a maximum of two Dataverse knowledge sources per agent; supported tables are standard or activity tables (plus virtual tables that use the finance and operations data provider); and synonyms and glossary definitions improve matching (in the newer agent experience, Dataverse sources rely on table metadata and column descriptions instead). Declarative agents in Microsoft 365 Copilot can also use a Dataverse capability, and Agent Builder exposes Dataverse knowledge through Business Applications in Work IQ, which an administrator must enable first.

For actions, Copilot Studio documents Microsoft-provided MCP servers for services including Dataverse, with access that enforces Dataverse role-based access control so agents act only within granted permissions.

### Microsoft 365 content and Copilot connectors

CAF's placement rule is simple: content that serves collaboration, reference or organisational knowledge belongs in Microsoft 365, where document-level permissions and retention govern it. Microsoft 365 Copilot agents retrieve through Microsoft Graph while enforcing permissions, sensitivity labels and tenant policies.

External content reaches Microsoft 365 Copilot through **Microsoft 365 Copilot connectors** (formerly Microsoft Graph connectors):

| Connector type | How data flows | Use when | Considerations |
|---|---|---|---|
| **Synced connectors (prebuilt or custom via the Copilot connectors API)** | Items with content, metadata and an access control list are ingested into the Microsoft Graph index and semantically indexed; periodic sync reflects adds, updates and deletes. | Large, relatively stable external knowledge bases (ticketing knowledge, wikis, product catalogues) that must be searchable across Microsoft 365 Copilot, Microsoft Search and agents. | Ingested items consume item quota; ACLs must be mapped correctly at ingestion; freshness depends on crawl schedule. |
| **Federated connectors** PREVIEW | Data stays in the source and is fetched in real time (citations refer to content returned from an MCP server); nothing is stored in Graph. | Data that must not be copied or must be real time. | Early access preview; read-only. |

### Search indexes, Foundry IQ and MCP for custom agents

For custom agents, CAF recommends a retrieval strategy that prioritises built-in Microsoft capabilities and approves custom solutions only when documented requirements exceed built-in limits. Built-in options it lists include Foundry IQ, Fabric IQ, Fabric data agents, MCP server connections, the SharePoint tool, Azure AI Search and OneLake search indexers. Foundry IQ is described as the default retrieval approach for Foundry-based agents: a unified knowledge base endpoint over one or more knowledge sources, using **agentic retrieval** (the system plans and iterates its searches rather than running one static query) with shared governance and consistent citations.

Choose **MCP** when agents must take actions or access real-time data, for example current stock levels or creating an incident. Decide early whether each system needs read-only or read-write access, require authentication for every tool call, and prefer identity passthrough where user-level permissions must persist

(Foundry supports key-based access, Microsoft Entra agent identity, project managed identity and OAuth identity passthrough).

| Requirement | Recommended way to expose the data |
|---|---|
| **Policies, procedures, project documents used by employees in Microsoft 365** | SharePoint or OneDrive (semantic index), scoped in declarative agents or Copilot Studio SharePoint knowledge |
| **External knowledge base (for example ServiceNow knowledge, Confluence) needed across Microsoft 365 Copilot** | Synced Microsoft 365 Copilot connector with correct ACLs |
| **Dynamics 365 or Power Apps records for grounding agents** | Dataverse knowledge (Dataverse search on) or Dataverse capability in declarative agents |
| **Dynamics 365 data for analytics, data agents and ML without ETL** | Link to Microsoft Fabric (Dataverse shortcuts in OneLake) |
| **Natural language analytics over curated data** | Fabric data agent over semantic models, lakehouses or warehouses, with Prep data for AI on semantic models |
| **Large or heterogeneous document corpus needing vector search and semantic ranking for custom agents** | Azure AI Search or Foundry IQ knowledge base |
| **Live transactional queries or record changes** | MCP server or API (connector, OpenAPI tool), with scoped identity |

Finally, CAF asks leaders to **document retrieval decisions**: list each data domain, classify its retrieval method (search, API or both), and record governance details (certified sources, which APIs require authentication, which permissions apply). Its example is a support agent where product FAQs use a knowledge index, order tracking uses an API to the order system and return labels use an API to the shipping system. The AB-100 module adds the governance layer: centralise data in Azure, Dataverse or Fabric, normalise schema, naming, metadata and taxonomy, provide multiple access paths (APIs, search indexes, RAG pipelines, connectors, SQL endpoints), implement governance early with Microsoft Purview (access policies, sensitivity labels, lineage, data quality rules), and keep data authoritative and current.

**COMMON TRAP**

Copying Dataverse or ERP data into a separate store "for the AI team" is rarely the best answer when a governed, non-copy option exists. On the exam, prefer Link to Microsoft Fabric, OneLake shortcuts, Copilot connectors with ACLs, or built-in knowledge sources over bespoke extract pipelines, unless the scenario states a requirement those options can't meet.

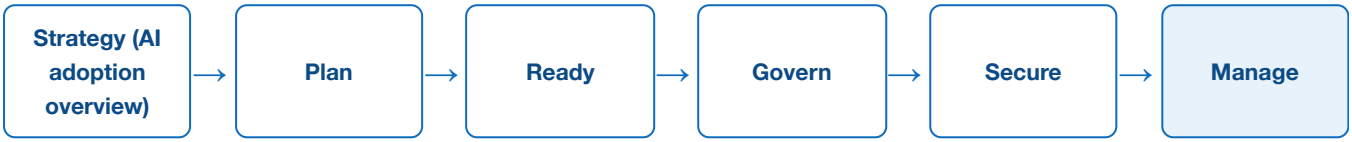
## 1.2 Design overall AI strategy for business solutions (part 1)

The strategy skills test whether you can turn a portfolio of use cases into a coherent, governed platform plan. Microsoft publishes two complementary bodies of CAF guidance that the AB-100 training explicitly unifies: the **AI adoption** scenario (organisation-wide AI) and the **AI agent adoption** guidance (the operating model for agents). Learn both, and learn how they map onto each other.

### 1.2.1 Implement the AI adoption process from the Cloud Adoption Framework for Azure

#### Current shape of the CAF AI guidance

The current CAF "AI adoption" article is an executive overview that covers what AI is (traditional, generative and agentic AI), the foundation of AI adoption, which AI to use (with the Microsoft AI decision tree), how to prepare an AI data foundation, and how to adopt AI. Its process diagram shows five phases: **Plan, Ready, Govern, Secure, Manage**. The strategy work that older material called a separate "AI strategy" phase now sits in the overview article itself (the "strategy" URL resolves to the same AI adoption article). The AB-100 module therefore describes the CAF AI phases as Strategy, Plan, Ready, Govern, Secure and Manage, and maps them to the agent lifecycle phases Plan agents, Govern and secure agents, Build agents and Operate (manage) agents.




The CAF AI agent adoption guidance uses four phases with named substeps:

- **Plan for agents:** business plan, technology plan, organisational readiness, data architecture.
- **Govern and secure agents:** Responsible AI, governance and security, prepare environment.
- **Build agents:** single and multi-agent systems, process to build agents.
- **Manage agents:** integrate agents, operate agents.

#### Foundation actions for leadership

Before the phases, the AI adoption article asks leaders to do three things: **lead AI adoption** (define expected business outcomes, sponsor them visibly, and document priorities, decision rights and value measurement in an AI strategy or charter), **centralise AI governance** (central ownership of organisation-wide policies and standards, for example through an AI Center of Excellence that defines Responsible AI requirements and common boundaries, so business groups can innovate within them) and provide **workforce access** to approved AI paired with training. It also states the selection principle that drives many exam answers: approve the solution that meets requirements with the **least operational responsibility**, because greater customisation requires more specialised skills and more operational ownership.

#### Phase mapping and deliverables

| CAF AI phase | Agent lifecycle phase | Key activities | Typical outputs |
|---|---|---|---|
| Strategy | Plan agents | Document business outcomes, prioritise use cases, decide whether agents are appropriate and which platform (SaaS or custom); draft data and Responsible AI strategies. | AI strategy brief; agent technology plan (decision tree, platform choice, guardrails) |
| Plan | Plan agents (continued) | Assess and acquire skills, access AI resources, prioritise use cases, run proofs of concept, define agent readiness criteria (data availability, governance readiness, identity model, connectors). | AI adoption plan; PoC report; agent readiness assessment |

| CAF AI phase | Agent lifecycle phase | Key activities | Typical outputs |
|---|---|---|---|
| Ready | Govern and secure agents (foundation) | Extend Azure landing zones, connectivity and governance boundaries for AI; prepare data architecture for agents; define agent governance. | AI landing zones, policy assignments, network segmentation, agent governance charter, data access model |
| Govern and Secure | Govern and secure agents (enforcement) | Policy-based governance for AI platforms, documented AI policies, risk monitoring, controls for agent behaviour, data access and audit. | AI and agent policy set, risk register, security controls for data, models and endpoints |
| (Ready to adopt on Azure) | Build agents | Standardise the build process across Copilot Studio and Foundry, evaluation gates, environment strategy, reference architectures. | Agent templates, evaluation gates, CI/CD with policy guardrails |
| Manage | Operate (manage) agents | Deployment authority, monitoring and SLOs, cost control, rollout patterns, lifecycle and deprecation. | AI operations baseline; agent operations playbook |

### Plan: details that become exam distractors

- **AI maturity levels.** CAF defines four levels by skills and data readiness. Level 1 (basic AI understanding, enterprise data available) supports any Copilot solution and quickstarts. Level 2 adds custom analytical workloads with Foundry Tools, generative chat without RAG, automated ML and fine-tuning. Level 3 adds RAG applications in Foundry and training ML models. Level 4 adds large generative or nongenerative workloads on VMs, AKS or Container Apps. Match ambition to maturity.
- **Access requirements.** Microsoft 365 Copilot requires a Microsoft 365 licence plus a Copilot licence; Copilot Studio requires a standalone or add-on licence; in-product Copilots require the primary product; Microsoft Foundry requires an Azure account.
- **PoCs.** Start with internal, non-customer-facing projects. Microsoft Copilots typically show return in days to weeks, while custom Azure AI and ML workloads take weeks to months. Add 20–30% contingency to estimates.
- **Responsible AI from the start,** using tools such as the HAX Toolkit and the Responsible AI Maturity Model, and operating practices such as GenAIOps or MLOps.

### Ready: AI uses your existing landing zone

The CAF Ready guidance is emphatic that AI workloads don't require a different landing zone architecture or governance model. Platform teams extend the existing governance baseline. Points to remember:

- Don't create AI-specific management groups. Place internet-facing AI workload subscriptions under the **Online** management group and internal ones under **Corp**.
- Add Azure Policy definitions for approved AI services (Foundry model deployment policies, Foundry Tools, Azure AI Search) through the existing management group hierarchy.
- Use Azure DDoS Protection for internet-facing AI workloads; choose ExpressRoute or VPN Gateway for on-premises data according to performance and availability needs.
- Each workload gets its own workload landing zone with subscriptions as boundaries for quota, cost, governance and security.
- Plan regions explicitly: model availability and capacity vary by region, retrieval data should sit near consuming AI services when latency matters, residency rules apply, and failover designs must include model deployments and retrieval data in the secondary region.

### Govern, Secure, Manage

CAF's AI governance process follows the NIST AI Risk Management Framework: assess AI organisational risks (using the six Responsible AI principles as a lens, plus external dependency and integration risks), document AI governance policies (model selection and onboarding, third-party tools and data, golden datasets, monitoring and retraining, regional compliance, user conduct, integration and replacement), enforce policies (automate with Azure Policy and Microsoft Purview where possible, manual audits and training where not) and monitor risks (for example, quarterly assessments for high-risk workloads and annual for lower risk, with independent review). The AB-100 module summarises Secure as implementing platform security for models, data, keys and endpoints with a complete AI asset inventory, and Manage as defining deployment authority within governance boundaries, comprehensive telemetry and SLOs, cost and incident runbooks, and agent lifecycle management.

| Activity (module RACI example) | Solution architect | Platform team | Security and compliance | Data or knowledge owner |
|---|---|---|---|---|
| Strategy and use case selection | A/R | C | C | C |
| Agent technology plan (SaaS or custom) | A/R | C | C | C |
| Landing zone and policies | C | A/R | C/R (controls) | C |
| Agent governance and SDLC | R | A | A/R (policy) | C |
| Build agents (Foundry or Copilot Studio) | A/R | R (platform services) | C | C/R (grounding data) |
| Operate agents and telemetry | A/R | R | C | C |

**EXAM TIP**

If asked which CAF activity comes first for a new AI initiative, the answer is almost always identifying use cases from business problems and defining measurable outcomes (strategy and plan), not provisioning landing zones or choosing a model. If asked how AI changes the landing zone, the answer is "extend the existing baseline", not "build a separate AI landing zone hierarchy".

## 1.2.2 Design the strategy for building AI and agents in business solutions

The CAF technology plan for agents reduces platform selection to one primary question: **does a SaaS agent meet the functional requirements?** If yes, adopt it. If not, build a custom agent and choose between Copilot Studio (SaaS, low-code), Microsoft Foundry (PaaS, low-code to pro-code) or GPUs and containers (IaaS or PaaS, full control). The AB-100 module restates this as a tiered approach: SaaS (prebuilt) agents for immediate value with minimal customisation, low-code agents in Copilot Studio for moderate customisation and rapid deployment, and pro-code agents in Foundry or custom hosting for maximum flexibility.

### Platform selection matrix

| Dimension | Microsoft 365 Copilot (prebuilt and extended) | Copilot Studio | Microsoft Foundry | GPUs and containers |
|---|---|---|---|---|
| **Service model** | SaaS | SaaS, low-code | PaaS, declarative (prompt agents) to code-first (hosted agents) | IaaS or PaaS (AKS, Container Apps) |
| **Best for (CAF)** | Personal productivity with minimal customisation | Process transformation with fast development and SaaS security | Strategic transformation with deep integration and custom logic | Compliance-sensitive or highly customised workloads needing full stack control |
| **Agent types (CAF)** | Retrieval, task | Retrieval, task, autonomous | Retrieval, task, autonomous | Retrieval, task, autonomous |
| **Orchestrator and models** | Copilot orchestrator and models (declarative agents); own engine for custom engine agents | Managed generative orchestration; model selection; Foundry models can be brought in | Your choice from the Foundry model catalog; own orchestration in hosted agents | Your own models and runtime |
| **Typical builders** | Business users (Agent Builder), pro developers (Agents Toolkit) | Makers and business analysts, fusion teams | Professional developers, AI engineers | Platform and ML engineering teams |
| **Integration** | Copilot connectors, MCP and API plugins (actions) | Prebuilt and custom connectors, agent flows, MCP, A2A, Azure AI Search | Tool catalog, toolboxes, MCP, OpenAPI, Azure Functions, Logic Apps, A2A | Anything you build |
| **Main cost factors** | Licence and metered usage | Copilot Credits based on agent usage | Model usage (tokens) and tool usage; container compute for hosted agents | Compute, storage, data transfer, model usage |
| **Governance burden** | Lowest; inherits Microsoft 365 compliance and RAI | Platform-enforced (environments, data policies) | Architect-led; evaluations, identity, networking you configure | Highest; you own everything |

### Microsoft Foundry building blocks to know

- **Agent types in Foundry Agent Service:** *prompt agents* (instructions, model and tools defined by configuration; Foundry runs them with no code or containers), *voice-based prompt agents* PREVIEW (real-time spoken interactions through Voice Live) and *hosted agents* (your own code and framework, such as Microsoft Agent Framework, LangGraph or Semantic Kernel, packaged as a container that Foundry runs with a managed endpoint, scaling and a dedicated Entra identity). You can also call the Responses API directly from code you already host, which creates an "ephemeral" agent defined in your application.
- **Toolboxes** curate tools (web search, file search, code interpreter, MCP servers, custom functions) once and share them through a single managed MCP-compatible endpoint with central authentication, governance and versioning.

- **Setup options:** the *basic* setup suits rapid prototyping and lacks network isolation; the *standard* setup gives control over data, security and networking, using public networking for nonconfidential workloads or private networking for confidential ones, and lets you bring your own storage, Azure AI Search and Azure Cosmos DB.
- **Enterprise capabilities:** per-agent Microsoft Entra identity, RBAC, guardrails and content filters, virtual network isolation, versioning, publishing to stable endpoints, and distribution to Microsoft 365 Copilot, Teams and the Entra Agent Registry.

#### A hybrid is normal

CAF explicitly recommends combining Copilot Studio's low-code interface with Foundry's models for sophisticated cases while keeping SaaS security, and notes that organisations often use multiple approaches. Validate choices with **time-boxed experiments** of one to two weeks per option, document findings for stakeholder review, proceed with low-code if it meets functional and security requirements, and only then shift to pro-code or adjust scope.

> **COMMON TRAP**
> "Most flexible" is not the same as "best". If Copilot Studio meets every stated requirement, an answer that proposes hosted agents on Foundry or a custom AKS deployment is wrong because it adds operational responsibility without a requirement that justifies it. Look for explicit triggers that push you up the stack: bring-your-own model, custom orchestration logic, private network isolation for the runtime, custom protocols, or code-level control.

### 1.2.3 Design a multi-agent solution by using Microsoft 365 Copilot, Copilot Studio, and Microsoft Foundry

#### Single agent first, unless a hard boundary exists

CAF gives three criteria for **starting** with a multi-agent architecture: the use case crosses **security and compliance boundaries** (for example, separation of duties where one agent prepares a transaction and another validates it), **multiple teams** own distinct knowledge areas and release cycles, or **future growth** across many functions is planned (CAF cites solutions spanning more than three to five distinct functions). The AB-100 module adds a fourth: the requirements dictate a series of actions with dependencies across two or more workstreams. Otherwise, test a single agent first. Role separation, speed to market, cost, large data volumes, high demand and multiple modalities do *not* automatically justify multiple agents; persona switching, better retrieval, policy controls or a larger context window often solve the problem.

| Approach | Use when | First step |
|---|---|---|
| **Single agent** | Narrow domain, unified context, speed and cost priority | Deploy one agent with required tools and iterate |
| **Comparative prototype** | Architecture unclear; need evidence on context handling, role separation or performance | Controlled tests of both architectures against success metrics |
| **Multi-agent** | Hard security, compliance or organisational boundaries, or guaranteed multi-domain scaling | Design isolated agents with scoped access and explicit interface contracts |

Know the trade-offs CAF lists for multi-agent systems: every handoff adds latency and requires state management; each agent needs its own prompt engineering, monitoring and debugging; credential and data transit points expand the security surface; and costs rise because agents process redundant context. The

benefit is least privilege by design, which limits blast radius.

### Assign each role to the right platform

| Role in the solution | Best-fit platform | Why | Typical orchestration |
|---|---|---|---|
| **Domain assistant in the flow of work** | Microsoft 365 Copilot (prebuilt or declarative agents) | Immediate value inside Teams, Outlook, Word; inherits Microsoft 365 security | Handoff, group chat |
| **Business workflow agent** | Copilot Studio | Rapid iteration, connectors, guardrails, agent flows | Sequential, handoff |
| **Integration or orchestration agent** | Microsoft Foundry | Pro-code tools, complex flows, custom evaluations | Concurrent, sequential, Magentic |




### Multi-agent options in Copilot Studio

A Copilot Studio agent can add other agents in several ways: **child agents** (lightweight agents inside the main agent), **connected Copilot Studio agents** in the same environment, and external agents: agents available over the **Agent2Agent (A2A) protocol**, **Microsoft Foundry agents** PREVIEW, **Fabric data agents** PREVIEW and agents built with the **Microsoft 365 Agents SDK** PREVIEW. Under generative orchestration, the main agent selects child and connected agents from their descriptions, just as it selects tools and topics.

| Use a child agent when | Use a connected agent when |
|---|---|
| **A single team owns the whole solution** | Different teams or developers manage different agents |
| **You want to group tools, instructions and knowledge logically inside one agent** | The main agent's ability to choose correctly degrades (rule of thumb: more than 30–40 choices of tools, topics and agents, or several items with similar descriptions) |
| **No separate settings, authentication or deployment are needed** | The subagent needs its own settings, such as its own model |
| **The subagent won't be published or reused independently** | The agent must be published on its own channels and reused by several main agents |
| **Shared ALM with the parent is fine** | Independent ALM is required |

Limitations worth remembering: an agent that already has connected agents can't itself be used as a connected agent by another main agent (agents without connected agents can be reused by many main agents); citations might not survive when outputs pass back to the calling agent; child and connected agents respect the main agent's general knowledge setting; and redirecting to Fabric data agents from a topic isn't currently supported. Microsoft's multi-agent guidance also notes that a child agent always receives the parent's context, whereas a connected agent has a context-inclusion setting that controls whether it receives conversation history, so you must confirm it rather than assume.

Copilot Studio's published best practices for multi-agent instructions are frequently paraphrased in scenarios: only one agent should respond to the user per turn (subagents return findings, the parent responds), subagent instructions must state that they are subagents, use strong directive language (MUST, NEVER, ONLY), give each subagent a distinct nonoverlapping knowledge source, write accurate and distinct subagent descriptions, define the full orchestration pattern in the parent (invoke, wait, combine, respond), test with domain-mismatch queries, and prefer "ask" over "inform" when a follow-up answer is expected.

### Protocols: A2A, MCP, Activity Protocol

| Integration need | Recommended approach | Notes |
|---|---|---|
| **Delegate a task to another agent with its own reasoning (built on another framework or hosted outside Copilot Studio)** | A2A connection | Supports multiturn interactions and rich contextual metadata; agent card at the agent endpoint plus `/.well-known/agent.json`; authentication None, API key or OAuth 2.0; uses custom connector infrastructure, so it can reach agents on-premises or in a virtual network. |
| **Use tools or resources from a server** | MCP server | Copilot Studio supports MCP tools and resources; server changes are reflected dynamically. Add through the MCP onboarding wizard (recommended) or a custom connector. |
| **Call a basic API or HTTP service** | Custom connector or HTTP tool | Deterministic call, no agent reasoning on the other side. |
| **Integrate an agent built with the Microsoft 365 Agents SDK** | Activity Protocol | Also used when Foundry agents connect to Copilot Studio and publish to Microsoft 365. |

On the Foundry side, Agent Service supports the OpenResponses and Activity protocols for Microsoft 365 publishing, an Invocations protocol for custom endpoints, and the A2A protocol for agent-to-agent communication (Learn states A2A v1.0 is generally available and v0.3 remains in preview). A newly created Foundry agent exposes only the Responses and A2A endpoints by default. To connect it to Copilot Studio as a connected Foundry agent, you must enable the **Activity protocol** endpoint programmatically (REST API or Python SDK); otherwise the call fails at runtime with an HTTP 400 error stating that the endpoint does not support activity. Only agents created in the new Foundry portal can be connected.

### Orchestration patterns for pro-code multi-agent systems

The AB-100 module highlights the Microsoft Agent Framework orchestration patterns and warns against brittle prompt-to-prompt daisy chains. Treat orchestration as a workflow with state, branching and error handling.

| Pattern | Use when |
|---|---|
| Sequential | Staged, deterministic pipeline (plan, enrich, verify, act) |
| Concurrent | Independent subtasks run in parallel, then results are aggregated and reconciled |
| Group chat | Agents contribute proposals in a mediated conversation and a moderator arbitrates |
| Handoff | Context and control move to a specialist agent or a human when a rule or threshold triggers escalation |
| Magentic | A central orchestrator dynamically pulls in the right expert agents at runtime |

Foundry also provides **workflows** for sequential logic, conditional branching and state management across agents. In Microsoft 365, the declarative agent manifest has a `worker_agents` property that lets one declarative agent use other declarative agents.

**PRODUCTION NOTE**

Design reviews for multi-agent solutions should check: one clear responsibility and minimum tools per agent, security scopes and data boundaries enforced per agent, graceful degradation if one agent fails, observability at every handoff (handoff latency, tool failure rate, decision quality), minimal handoff payloads (identifiers rather than raw content), and approval gates for high-risk actions. Also confirm that calling a connected agent can't bypass restrictions the parent has, for example a delete capability the parent is not allowed to use.

### 1.2.4 Develop the use cases for prebuilt agents in the solution

Prebuilt (SaaS) agents are the first stop in the CAF decision tree. They deliver value quickly because no custom development is needed, and they inherit the governance of the product that hosts them. The architect's task is to find where they fit the process, confirm feasibility, and decide what remains for extension or custom build.

#### Microsoft 365 Copilot prebuilt agents

Microsoft deploys **Researcher** and **Analyst** to users with Microsoft 365 Copilot licences as Microsoft-installed agents. Researcher handles complex, multi-step research and produces source-cited reports that draw on the web and the work content the user can access (files, emails, meetings and chats). Other agents reach users as admin-installed or user-installed agents, and administrators govern all of them in the Microsoft 365 admin center. The CAF technology plan also lists agents such as App Builder and Workflows in Microsoft 365 Copilot, and in-product SaaS agents across the stack: Fabric data agents, Azure Copilot agents, Dynamics 365 agents, Security Copilot agents and GitHub Copilot agents. Role-based experiences include the Sales agent in Microsoft 365 Copilot, which lets sellers search, synthesise and act on CRM data from Outlook, Teams and other Microsoft 365 apps.

#### Dynamics 365 prebuilt agents (selection)

| App | Agent | What it does |
|---|---|---|
| Customer Service and Contact Center | Case Management Agent | Automates the case lifecycle: create, update, resolve, follow up and close |
| | Customer Knowledge Management Agent | Analyses closed cases, fills knowledge gaps and creates or manages knowledge articles |
| | Quality Evaluation Agent | Autonomous and AI-assisted assessments of cases and conversations against a supervisor-defined framework |
| | Customer Intent Agent | Discovers common intents and suggests new ones; supports intent-driven routing and self-service |

| App | Agent | What it does |
|---|---|---|
| Sales | Sales Qualification Agent | Researches leads, assesses fit, sends outreach and engages leads autonomously |
| | Sales Opportunity Agent | Researches opportunities, surfaces emerging risks and promising deals |
| | Sales Close Agent | Manages the sales cycle through to closure with templated personalisation |
| | Sales Research Agent | Answers complex business questions through dialogue with sales data |
| Field Service | Scheduling Operations Agent | Optimises resource schedules interactively or in batches with custom goals |
| Finance | Account Reconciliation Agent PREVIEW | Reconciles subledger and ledger data earlier than period end (production ready preview) |
| Supply Chain Management | Supplier Communications Agent PREVIEW | Automates vendor follow-up and purchase order updates (production ready preview) |
| Project Operations | Time Entry, Expense and Approvals agents PREVIEW | Draft time entries, extract receipts into expense reports, review approvals (production ready preview) |
| Business Central | Sales Order Agent, Payables Agent | Capture sales orders from customer email; process incoming vendor invoices |

Dynamics 365 apps also inherit platform capabilities such as the agent feed PREVIEW for human-agent collaboration and the Power Apps MCP Server PREVIEW, and finance and operations apps expose a Dynamics 365 ERP MCP server for agents to work with ERP data and business logic.

### Building the use case

The AB-100 module identifies high-value prebuilt agent use cases through four criteria: **task repetitiveness** (frequency per user per week), **knowledge intensity**, **pain points** (delays, errors, bottlenecks) and the **ROI KPIs** affected (hours saved, ticket volume, cycle time). It then maps process steps to capabilities (searching documents to retrieval, drafting to summarisation and generation, policy questions to knowledge Q&A, daily insight gathering to synthesis) and confirms feasibility: the required data exists in Microsoft 365, the interaction fits a conversational pattern, no advanced multi-agent orchestration is needed, and accuracy expectations suit retrieval-first answers. The Microsoft Scenario Library helps map functional areas to Copilot capabilities.

| Situation | Prebuilt | Extend | Build custom |
|---|---|---|---|
| **Standard process the product already supports (case follow-up, lead qualification, research reports)** | Yes, configure and govern | Only if extra knowledge or actions are needed | No |
| **Prebuilt agent fits but needs company knowledge or a line-of-business action** | Base | Yes: declarative agent, connectors, MCP or API plugin, or Copilot Studio extension | No |
| **Unique process, complex rules, external channels, own models or proactive behaviour** | No | Partial at best | Yes: Copilot Studio or Foundry custom agent |

**EXAM TIP**

When a scenario names a Dynamics 365 process that matches a shipped agent (for example "automatically draft knowledge articles from resolved cases" or "autonomously research and qualify inbound leads"), the expected answer is to enable and configure that prebuilt agent, not to build a Copilot Studio agent from scratch. Watch for "production ready preview" status if the scenario requires GA-only components.

### 1.2.5 Define solution rules and constraints for Copilot Studio, Microsoft Foundry and Foundry Tools

"Rules and constraints" covers two things: the hard platform limits you must design within (quotas, capacity, file and knowledge limits), and the rules you impose (data access, behaviour envelopes, DLP, content moderation, networking, human review). The AB-100 module recommends defining **one set of rules** and applying it differently per platform.

| Rule category | Copilot Studio application | Foundry application |
|---|---|---|
| **Data access** | Strict connector scoping, data policies, user authentication | Full control; explicit boundaries via RBAC, identity and network |
| **Actions** | Light task automation through tools and agent flows | Deep tool integration and orchestration |
| **Governance** | Platform-enforced | Architect-led and customisable |
| **Risk level** | Low to medium | Medium to high |
| **Evaluation** | Built-in safety and agent evaluation | Requires evaluation pipelines you design |

#### Behaviour envelopes and data rules

Write explicit allowed and disallowed behaviours into instructions and design ("agent may summarise", "agent may not decide", "agent may not execute financial transactions"). Give agents only the data they need, mask sensitive fields, limit grounding to curated sources, define memory policy (ephemeral or persistent), restrict cross-domain access (for example HR, finance, legal), require human review for high-risk tasks such as financial approvals, legal drafting or health decisions, and enable auditing for tool invocation. For Foundry and Foundry Tools, implement least privilege per tool, use evaluation tools to test failure cases, define escalation rules for deferring to humans, and document every tool integration with its security and audit considerations.

#### Copilot Studio quotas

Quotas limit how often agents can receive messages and are applied per Dataverse environment, in addition to purchased capacity.

| Quota (per Dataverse environment) | Value |
|---|---|
| Messages to an agent (paid plan) | 8,000 requests per minute |
| Generative AI messages, 1–10 prepaid packs | 50 RPM / 1,000 RPH |
| Generative AI messages, 11–50 prepaid packs | 80 RPM / 1,600 RPH |

| Quota (per Dataverse environment) | Value |
|---|---|
| Generative AI messages, 51–150 prepaid packs | 100 RPM / 2,000 RPH (plus 1 RPM / 20 RPH per extra 10 packs above 150) |
| Trial or developer environments | 10 RPM / 200 RPH |
| Pay-as-you-go environments; Microsoft 365 Copilot users | 100 RPM / 2,000 RPH |

Generative AI messages include generative orchestration, agent actions, AI tools, agent flow actions and generative answers. When the quota is reached, users see a failure notice. Learn's throughput guidance warns that customer-facing agents, autonomous agents and agents that trigger many flows can exceed per-minute or per-hour limits even when monthly volume looks manageable.

### Copilot Studio limits

| Item | Limit |
|---|---|
| Knowledge sources per agent | 500 across all types |
| Instructions for an agent published to Microsoft 365 Copilot | 8,000 characters |
| Connector payload | 5 MB (450 KB in GCC) |
| File upload size / number of uploaded files | 512 MB / 500 |
| Topics per agent (Dataverse environment) / trigger phrases per topic | 1,000 / 200 |
| SharePoint site URLs with generative orchestration | 25 |
| Uploaded SharePoint or OneDrive source | Up to 1,000 files, 50 folders, 10 subfolder levels; 512 MB per file; sync every 4–6 hours |
| Model response stream timeout | 5 minutes (not configurable) |
| Teams plan (select Microsoft 365 subscriptions) | 10 sessions per user per 24 hours across agents; 6,000 Power Platform requests per 24 hours |

### Capacity and licensing messages: Copilot Credits

Copilot Studio usage is measured in **Copilot Credits**, pooled across the tenant and allocatable to environments, with an optional pay-as-you-go meter. Rates to know:

| Feature | Rate | Microsoft 365 Copilot licensed user |
|---|---|---|
| Classic answer | 1 credit | No charge |
| Generative answer | 2 credits | No charge |
| Agent action (triggers, deep reasoning, topic transitions; also Computer-Using Agents) | 5 credits | No charge (CUA not included) |
| Tenant graph grounding for messages | 10 credits | No charge |

| Feature | Rate | Microsoft 365 Copilot licensed user |
|---|---|---|
| Agent flow actions | 13 credits per 100 actions | No charge only for the "When an agent calls the flow" trigger |
| Text and generative AI tools (basic / standard / premium) | 1 / 15 / 100 credits per 10 responses | No charge |

The Microsoft 365 Copilot inclusion applies to employee-facing scenarios where the agent operates with the authenticated, licensed user's identity. Reasoning models add the premium tools rate on top of the feature rate. When consumption reaches **125% of prepaid capacity**, custom agents are disabled (current conversations finish; new attempts are rejected) and administrators are notified by email and in the Power Platform admin center. Capacity allocated to an environment is honoured even when the tenant pool is in overage, and environments with pay-as-you-go aren't affected. Agent flows are enforced separately: when prepaid capacity is fully consumed, new agent flow runs are blocked while the agent continues for non-flow interactions. Administrators can set monthly consumption limits per agent in the Power Platform admin center.

**COMMON TRAP**

Power Automate cloud flows use Power Automate licensing, not Copilot Credits, and aren't subject to Copilot Studio enforcement. Agent flows managed in Copilot Studio do consume Copilot Credits. Do not confuse the two when a scenario asks why flows stopped running.

### Data loss prevention (data policies)

Data policies in the Power Platform admin center govern how agents connect to data and services. Since early 2025, data policy enforcement is in effect for all tenants and exemptions are no longer supported. Connectors are classified into **Business**, **Non-business** and **Blocked** groups, and connectors used together must be in the same group. Newer Copilot Studio connectors often land in the default Non-business group. Common rules:

| To prevent makers from... | Block (or filter) this connector |
|---|---|
| **Publishing agents that don't require authentication** | Chat without Microsoft Entra ID authentication in Copilot Studio |
| **Using local file uploads as knowledge** | Knowledge source with documents in Copilot Studio (does not block SharePoint or OneDrive files) |
| **Using SharePoint or OneDrive uploaded sources** | Knowledge source with SharePoint and OneDrive in Copilot Studio (supports endpoint filtering) |
| **Using public websites as knowledge** | Knowledge source with public websites and data in Copilot Studio (supports endpoint filtering) |
| **Making HTTP requests** | HTTP (supports endpoint filtering) |
| **Using Azure Bot Service skills** | Skills with Copilot Studio |
| **Publishing to specific channels** | Channel connectors such as Direct Line channels, Microsoft Teams + M365 Channel, SharePoint, WhatsApp, Facebook, Omnichannel |
| **Adding event triggers or running authenticated automated evaluations** | Microsoft Copilot Studio |

Blocking Power Platform connectors used as tools also blocks MCP tools that rely on those connectors. Use **endpoint filtering** instead of a full block when only specific sites or endpoints should be allowed.

#### Content moderation

Copilot Studio moderation can be set at agent level (Generative AI settings), topic level (generative answers node) and prompt level (prompt tool). Levels range from Lowest to Highest, the default is **High**, and the topic-level setting takes precedence at runtime. Lower levels produce more answers but may allow more harmful content.

#### Microsoft Foundry and Foundry Tools constraints

- **Model quota** is assigned per subscription, per region, per model and per deployment type in tokens per minute (TPM), with corresponding requests per minute (RPM). Quotas aren't enforced at tenant level. Each deployment receives part of the quota; exceeding TPM or RPM returns HTTP 429.
- **Agent Service limits** include 128 tools per agent, 10,000 files per agent or thread, 512 MB per uploaded file for prompt agents, 100,000 messages per thread and 1,000 versions per agent. Concurrent hosted agent sessions are limited per subscription per region (2,000 in some regions, 1,000 elsewhere), and tool availability varies by region and model.
- **Guardrails:** default safety policies apply to all Foundry models except transcription models, covering content harm categories (hate, sexual, self-harm, violence), Prompt Shields for user prompt attacks and document (indirect) attacks, block lists and more. Guardrails for agents are in PREVIEW.
- **Networking:** use the standard setup with private networking for confidential workloads; private networking is available for prompt agents and hosted agents support bring-your-own virtual network.
- **Policy:** govern Foundry model deployments with Azure Policy through the landing zone management group hierarchy (CAF Ready).

> **PRODUCTION NOTE**
> Separate development, test and production with dedicated configuration on every platform: Copilot Studio environments are isolated per environment within the tenant boundary; Foundry needs an architected deployment environment (virtual network, private endpoints, region constraints). Establish SLOs for reliability, latency and throughput, agent health monitoring, incident response and rollback procedures before go-live.

### 1.2.6 Determine the use of generative AI and knowledge sources in Copilot Studio agents

#### Generative versus classic orchestration

The orchestration mode decides how much the model, rather than authored topics, drives the conversation.

| Behaviour | Generative orchestration | Classic orchestration |
|---|---|---|
| **Topic selection** | By the description of the topic's purpose | By matching trigger phrases |
| **Child and connected agents** | Selected by description | Not applicable |
| **Tools** | Agent chooses tools by name and description | Called explicitly from topics |
| **Knowledge** | Agent proactively searches knowledge | Fallback when no topic matches, or explicit generative answers node |

| Behaviour | Generative orchestration | Classic orchestration |
|---|---|---|
| **Combining components** | Combines topics, tools, agents and knowledge in one plan | Selects a single topic |
| **Asking for missing input** | Generates questions automatically | Question nodes authored in topics |
| **Responding** | Generates the response from everything used | Message nodes authored in topics |

Use classic orchestration (or tightly authored topics) where exact, controlled wording is required, for example regulated disclosures; use generative orchestration for flexible, multi-intent conversations, autonomous agents and multi-agent designs. Classic answers are also cheaper (1 credit versus 2 for generative answers).

### Knowledge sources and their limits

| Source | How it works | Generative mode | Classic mode | Authentication |
|---|---|---|---|---|
| Public website | Bing search restricted to listed sites | 25 websites | 4 URLs | None |
| Documents (uploaded) | Files stored and indexed in Dataverse | All documents | Limited by Dataverse file storage | None |
| SharePoint | Graph search over SharePoint URLs | 25 URLs | 4 URLs per generative answers node | User's Entra ID |
| Dataverse | RAG over Dataverse tables | Unlimited | 2 sources, 15 tables each | User's Entra ID |
| Enterprise data via Copilot connectors | Connector content indexed by Microsoft Search | Unlimited | 2 per agent | User's Entra ID |

Other options: **unstructured data sources** (SharePoint and OneDrive files and folders, Salesforce, ServiceNow, Confluence and Zendesk) are ingested, chunked and indexed with per-user permission checks and four-to-six-hour sync; **Azure AI Search** brings an existing vector index with semantic ranking and key, certificate or Entra ID authentication; and **Azure OpenAI, Bing Custom Search and custom data** are available only through the "Classic data" option of a generative answers node, because generative orchestration doesn't support custom data or Bing Custom Search as agent-level knowledge. In generative orchestration, an internal model filters knowledge sources when there are more than 25 (uploaded files don't count toward that limit).

### Generative AI settings that change grounding behaviour

- **Allow ungrounded responses** (requires generative orchestration). When off, a response is blocked in any turn where the agent used no knowledge source or tool, and the Fallback topic triggers. It doesn't guarantee that general knowledge is never blended with retrieved content. Answers without an in-text citation can be withheld, so instruct the agent to always cite sources and avoid instructions (such as "respond only in JSON") that suppress citations.
- **Use information from the web / Web Search** (requires generative orchestration) searches all Bing-indexed public sites using Grounding with Bing Search, in parallel with configured website sources.

- **Tenant graph grounding with semantic search** uses the Microsoft 365 semantic index for better SharePoint and connector retrieval. Users with Microsoft 365 Copilot licences already get it; enabling it for unlicensed users costs 10 credits per message. It requires a Microsoft 365 Copilot licence in the tenant and "Authenticate with Microsoft".
- **Official sources** mark a strictly verified source as authoritative, but the feature isn't compatible with generative orchestration.
- **Moderation** as described in 1.2.5.

| Requirement | Knowledge choice |
|---|---|
| **Structured business records with relationships** | Dataverse knowledge (with synonyms, glossary and good column descriptions) |
| **Large document estate needing vector search and high precision ranking** | Azure AI Search knowledge source |
| **Internal policies maintained in SharePoint, permission-trimmed** | SharePoint knowledge, ideally with tenant graph grounding |
| **Knowledge articles in ServiceNow, Salesforce, Confluence or Zendesk** | Unstructured data source or Copilot connector |
| **Public product documentation** | Public website knowledge |
| **Answers must come only from approved content** | Generative orchestration with "Allow ungrounded responses" off and web search off; citation instructions |
| **Exact, legally reviewed wording** | Authored topics (classic answers) |

### 1.2.7 Determine when to build custom agents or extend Microsoft 365 Copilot

Microsoft 365 Copilot can be extended in several ways, and the first decision is where people will use the experience: inside Microsoft 365 Copilot experiences (declarative agents, connectors, skills and MCP-based tools packaged as plugins for Microsoft 365 Copilot) or in apps and custom agents you host (Copilot APIs, Work IQ APIs, custom engine agents). A single solution can combine both, but each part keeps its own identity, authentication, deployment and governance requirements.

#### Extension building blocks

| Building block | What it adds | Key facts |
|---|---|---|
| Declarative agent | Custom instructions, knowledge and actions on top of Copilot's orchestrator and models | Built with Agent Builder, Copilot Studio or Microsoft 365 Agents Toolkit; no extra hosting; inherits Microsoft 365 security, compliance and RAI |
| Microsoft 365 Copilot connector | External content in the Graph index for grounding | Synced with ACLs, or federated PREVIEW |
| MCP and API plugins | Actions against MCP servers or REST APIs described by OpenAPI | Supported as actions within declarative agents, not as standalone experiences in Microsoft 365 Copilot |
| Custom engine agent | Your own orchestrator and models | Hosted in Copilot Studio or externally (for example Azure and Foundry); supports proactive messaging, group collaboration and external channels; you own compliance, RAI and security |

| Feature | Declarative agents | Custom engine agents |
|---|---|---|
| **Use case** | Focused scenarios in Microsoft 365 Copilot | Complex workflows or advanced integrations |
| **Sharing and access** | Designed for individuals | Individual and group collaboration (for example, a Teams channel or meeting) |
| **Customisation** | Limited to Copilot's orchestrator and models | Full choice of models (LLM, SLM, fine-tuned) and orchestration |
| **Proactive interactions** | Not supported; user-initiated | Can trigger actions without user input |
| **Channels** | Microsoft 365 apps | Microsoft 365 and external apps |
| **Hosting** | Microsoft 365 | Copilot Studio or external services such as Azure |
| **Compliance and security** | Inherited from Microsoft 365 | Your responsibility |

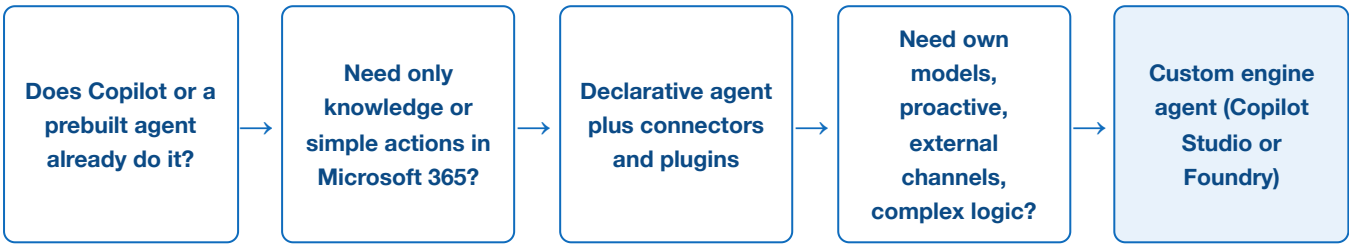
Learn's guidance condenses to these triggers. **Build a declarative agent** when you want Copilot's orchestration and models for consistent security and compliance, faster low-code delivery, and users who work inside Microsoft 365 apps (for example, an IT help desk agent responding to @mentions in Teams). **Build a custom engine agent** when you need custom orchestration with precise business rules and multiple system integrations (Learn's example is a loan approval agent with multiple credit check systems), your own or domain-specific models, group productivity, an existing assistant to bring into Microsoft 365, availability outside Microsoft 365, or proactive messaging. The AB-100 module adds a scope view: simple retrieval, summarisation and productivity-only contexts favour extending Copilot; complex multi-step workflows, high-volume automation, data outside Microsoft 365, strict grounding control and autonomy favour custom agents, which in turn require custom governance, monitoring and evaluation.

**Listing 1.1: Minimal declarative agent manifest (schema v1.8) scoped to one SharePoint library and one Copilot connector**

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/copilot/declarative-
agent/v1.8/schema.json",
  "version": "v1.8",
  "name": "Contoso Policy Assistant",
  "description": "Answers HR and travel policy questions for Contoso employees.",
  "instructions": "Answer only from the configured knowledge. Always cite the source document. If
the answer is not in the knowledge, say so and suggest contacting HR.",
  "capabilities": [
    {
      "name": "OneDriveAndSharePoint",
      "items_by_url": [
        { "url": "https://contoso.sharepoint.com/sites/HR/Policies" }
      ]
    },
    {
      "name": "GraphConnectors",
      "connections": [ { "connection_id": "travelKnowledgeBase" } ]
    }
  ],
  "conversation_starters": [
    { "title": "Travel", "text": "What is the hotel limit for London?" }
  ],
  "behavior_overrides": {
    "special_instructions": { "discourage_model_knowledge": true }
```

```
  }
}
```

Manifest facts that make good exam distractors: `instructions` is limited to 8,000 characters, `name` to 100 and `description` to 1,000; up to 12 conversation starters; `actions` references 1–10 plugins; omitting the item list in `OneDriveAndSharePoint` (or the connections list in `GraphConnectors`) gives the agent access to all of those sources the user can reach, so scope explicitly. Capabilities other than web search are available to users only if the tenant allows metered usage or the user has a Microsoft 365 Copilot licence. Version 1.8 added the `EmailActions` and `MeetingActions` capabilities, and the manifest supports `sensitivity_label`, `disclaimer`, `editorial_answers` and `worker_agents`.




**COMMON TRAP**

"Declarative" appears in two products. A Microsoft 365 Copilot *declarative agent* runs on Copilot's orchestrator and models. A Foundry *prompt agent* (described in CAF as declarative, prompt-based) runs on Foundry Agent Service with a model you choose. If a scenario requires choosing the model or keeping the runtime in your Azure network, the Microsoft 365 declarative agent is not the answer.

**EXAM TIP**

Proactive behaviour (the agent starts a workflow or sends a message without the user asking), group collaboration in a Teams channel, channels outside Microsoft 365, and bring-your-own model are the four strongest signals for a custom engine agent. Inheriting Microsoft 365 compliance with minimal engineering is the strongest signal for a declarative agent.

## 1.3 Additional practice questions

**Practice 1.1** MULTIPLE CHOICE

Fabrikam's accounts payable team matches supplier invoices to purchase orders. Every invoice arrives as a structured EDI message, and the matching rules (exact PO number, quantity within 2%, price equal to contract price) never change. Volume is 40,000 invoices a month. The CFO asks for "an AI agent" to do the matching. You need to recommend the most appropriate approach that minimises cost and maximises reliability. What should you recommend?

A. A Copilot Studio autonomous agent with generative orchestration and a Dataverse knowledge source
B. A deterministic automation (for example, a flow or code) that applies the matching rules, with exceptions routed to people
C. A Foundry hosted agent that uses a reasoning model to evaluate each invoice
D. A declarative agent in Microsoft 365 Copilot that the AP team prompts for each invoice

**Practice 1.2** MULTIPLE RESPONSE

A Copilot Studio agent grounded on a SharePoint site answers travel policy questions. Users report two problems: the agent often quotes last year's per diem rates, and some answers mix rules from the UK and US policies. Both policy versions and both countries' documents are in the same library. You need to improve grounding data quality. Which two actions should you take? Each correct answer presents part of the solution.

A. Remove or archive superseded policy versions from the knowledge location
B. Turn on "Use information from the web" so the agent can check current rates
C. Separate UK and US policies into distinct, clearly described knowledge sources or scopes aligned to the user's context
D. Lower the content moderation level to Lowest
E. Grant the agent maker's account Full Control on the library

**Practice 1.3** MULTIPLE CHOICE

Contoso runs Dynamics 365 Sales and Dynamics 365 Finance. The data team wants a Fabric data agent and Power BI semantic models over both apps' data. Security requires that data isn't exported through custom ETL pipelines and that the CRM remains the system of record. You need to make the data available in OneLake. What should you use?

A. Azure Data Factory pipelines that copy Dataverse tables into a Fabric warehouse nightly
B. Link to Microsoft Fabric from Power Apps, which creates Dataverse shortcuts in OneLake
C. A Microsoft 365 Copilot synced connector for Dataverse
D. Export to Excel and upload the files to a lakehouse

**Practice 1.4** MULTIPLE CHOICE

A Fabric data agent uses a Power BI semantic model as its only source. Users report incorrect totals when they ask about "net revenue". The data agent's instructions already define net revenue precisely, but the problem persists. You need to improve the accuracy of answers over the semantic model. What should you do?

A. Add more examples to the data agent instructions
B. Configure Prep data for AI on the semantic model (AI data schema, verified answers and AI instructions) and improve model metadata
C. Replace the semantic model with a KQL database
D. Increase the Fabric capacity SKU

**Practice 1.5** MULTIPLE CHOICE

Northwind is following the CAF AI adoption guidance. It is deploying its first internet-facing AI workload (a Foundry-based customer agent) into an existing Azure landing zone. The platform team proposes a new "AI" management group with its own policies. You need to recommend where to place the workload subscription. What should you recommend?

A. Under a new AI management group created for all AI workloads
B. Under the existing Online management group, extending the baseline with Azure Policy for approved AI services
C. Under the Corp management group, because Foundry is a PaaS service
D. In the platform landing zone's connectivity subscription

**Practice 1.6** MULTIPLE CHOICE

An HR team owns a Copilot Studio employee assistant. The IT department wants to add IT support capabilities. IT will maintain its own agent with a different model, its own ALM pipeline, and will also publish it directly to Teams for IT staff. You need to design how the HR assistant uses the IT capabilities. What should you use?

A. A child agent inside the HR assistant
B. A connected Copilot Studio agent owned by IT
C. A new topic in the HR assistant with IT trigger phrases
D. An HTTP request node that calls the IT ticketing API

**Practice 1.7** MULTIPLE CHOICE

A logistics partner exposes a route-planning agent built with LangGraph and hosted in its own cloud. The agent publishes an agent card and supports multiturn task delegation. Your Copilot Studio agent must delegate route planning to it and pass the conversation history. You need to choose the integration method. What should you use?

A. An MCP server connection
B. An A2A agent connection
C. A Microsoft 365 Copilot connector
D. A custom connector that calls a REST endpoint

**Practice 1.8** MULTIPLE CHOICE

A contact centre uses Dynamics 365 Customer Service. Managers want knowledge articles to be drafted automatically from resolved cases, with knowledge gaps identified, and they want to avoid custom development. You need to recommend a solution. What should you recommend?

A. Build a Copilot Studio agent with a Dataverse knowledge source over the case table
B. Enable and configure the Customer Knowledge Management Agent
C. Deploy a Foundry hosted agent that summarises cases nightly
D. Create a declarative agent in Microsoft 365 Copilot that reads case emails

**Practice 1.9** MULTIPLE CHOICE

A security review finds that makers have published Copilot Studio agents to a public website with "No authentication". Policy requires that every agent user is authenticated with Microsoft Entra ID. You need to prevent this configuration tenant-wide with the least administrative effort. What should you do?

A. Remove the Environment Maker role from all users
B. Create a data policy that blocks the "Chat without Microsoft Entra ID authentication in Copilot Studio" connector
C. Block the HTTP connector in a data policy
D. Set content moderation to Highest on each agent

**Practice 1.10** MULTIPLE RESPONSE

Tailspin has a tenant-wide prepaid pool of Copilot Credits shared by all environments. A business-critical customer service agent in the Production environment was disabled after the tenant's consumption reached 125% of prepaid capacity, mainly because of experimental agents in other environments. You need to ensure the critical agent keeps running in future months, even if other environments overconsume. Which two actions should you take? Each correct answer presents a complete solution.

A. Allocate dedicated Copilot Credit capacity to the Production environment
B. Link a pay-as-you-go billing plan to the Production environment
C. Convert the agent's agent flows to Power Automate cloud flows
D. Switch the agent from generative to classic orchestration
E. Increase the Production environment's Dataverse storage

**Practice 1.11** MULTIPLE CHOICE

A bank wants a loan pre-assessment agent available in a Teams channel where several underwriters collaborate. The agent must use a fine-tuned model hosted in Foundry, apply complex credit rules across three systems, and post a proactive alert to the channel when a bureau report arrives. You need to choose the agent type for Microsoft 365. What should you choose?

A. A declarative agent with an API plugin
B. A custom engine agent
C. A declarative agent with a Copilot connector
D. Researcher in Microsoft 365 Copilot

**Practice 1.12** MULTIPLE RESPONSE

A pharmaceutical company builds a Copilot Studio agent with generative orchestration that answers questions about approved product labelling stored in SharePoint. Compliance requires that the agent never answers from the model's general knowledge or from the public internet. You need to configure the agent. Which two settings should you configure? Each correct answer presents part of the solution.

A. Turn off "Allow ungrounded responses"
B. Turn off "Use information from the web"
C. Mark the SharePoint source as an official source
D. Turn on "Tenant graph grounding with semantic search" for all users
E. Set content moderation to Lowest

## Answers and explanations

**1.1: B.** CAF's business plan guidance says structured, predictable, rule-based tasks should use regular code or nongenerative models because they are faster, cheaper and more reliable. Inputs are structured and rules are fixed, so an agent adds cost and nondeterminism. A and C add generative reasoning (and credits or tokens) where none is required; D doesn't automate anything at volume.

**1.2: A, C.** Stale rates are a timeliness problem, fixed by removing superseded versions. Mixed UK and US rules are a relevance problem, fixed by scoping and clearly describing distinct sources so retrieval selects the right one. Web search (B) adds unvetted content; moderation (D) controls harmful content, not accuracy; widening permissions (E) breaks least privilege and doesn't affect relevance or timeliness.

**1.3: B.** Link to Microsoft Fabric makes Dynamics 365 data, including finance and operations apps, available in OneLake through shortcuts with no export or ETL, and the data stays in Dataverse. A violates the no-ETL requirement; C targets Microsoft 365 Copilot grounding, not OneLake analytics; D is manual and ungoverned.

**1.4: B.** For semantic model sources, the DAX generation tool relies only on model metadata and Prep data for AI configuration and ignores data agent-level instructions for DAX generation. Adding more agent instructions (A) won't help; changing the source type (C) or capacity (D) doesn't address the ambiguity.

**1.5: B.** CAF Ready guidance says AI workloads use the existing landing zone and management groups; internet-facing workloads generally belong under Online, internal ones under Corp, and platform teams extend the baseline with Azure Policy for approved AI services rather than creating AI-specific management groups.

**1.6: B.** Connected agents suit separate team ownership, different settings such as the model, independent ALM, independent publishing to channels and reuse. Child agents (A) are for a single team with no separate settings or publishing. A topic (C) or HTTP call (D) doesn't provide a separately owned agent.

**1.7: B.** A2A is designed for delegating tasks to external agents with their own reasoning, supports multiturn interactions and passes rich metadata including chat history. MCP (A) exposes tools and resources rather than a reasoning agent; a REST custom connector (D) is a plain API call; Copilot connectors (C) ingest content for search.

**1.8: B.** The Customer Knowledge Management Agent analyses closed cases, fills knowledge gaps and creates or manages knowledge articles. The CAF technology plan says to adopt a SaaS agent when it meets functional requirements; the other options require custom development.

**1.9: B.** Blocking the "Chat without Microsoft Entra ID authentication in Copilot Studio" connector in a data policy prevents makers from publishing agents that don't require authentication, leaving only "Authenticate with Microsoft" or manual authentication. A blocks all making, C blocks HTTP calls, and D is unrelated to authentication.

**1.10: A, B.** Copilot Studio honours capacity allocated to an environment even when the tenant pool is in overage, and an environment with a pay-as-you-go meter isn't affected by tenant overage enforcement. Moving to cloud flows (C) doesn't change agent enforcement; orchestration mode (D) changes rates but not enforcement; Dataverse storage (E) is unrelated.

**1.11: B.** Custom engine agents support your own models, custom orchestration with complex rules across systems, group collaboration in Teams channels and proactive messaging. Declarative agents (A, C) use Copilot's orchestrator and models and don't support proactive interactions; Researcher (D) is a prebuilt research agent.

**1.12: A, B.** Turning off "Allow ungrounded responses" blocks responses in turns where no knowledge source or tool was used, and turning off "Use information from the web" stops Bing web search. Official sources (C) aren't compatible with generative orchestration; tenant graph grounding (D) improves retrieval quality but doesn't restrict sources; moderation (E) concerns harmful content.

### Key takeaways

- Decide in order: is it AI, is it an agent (multistep, many tools, adaptive), how autonomous, and which platform. Rule-based work belongs in deterministic automation; static Q&A belongs in classic RAG.
- Score use cases on business impact, technical feasibility and user desirability, pilot the hardest steps, and use baseline KPIs as go or no-go decision gates.
- Grounding quality has five dimensions: accuracy, relevance, timeliness, cleanliness and availability. Availability is usually a permissions, authentication, label or file-limit issue, not a model issue.
- Organise data once for many AI consumers: OneLake data products, Link to Microsoft Fabric for Dataverse, Copilot connectors with ACLs for external content, Prep data for AI on semantic models, Foundry IQ or Azure AI Search for custom agents, and MCP for live data and actions.
- CAF AI adoption shows Plan, Ready, Govern, Secure, Manage, with strategy in the AI adoption overview; the agent guidance adds Plan, Govern and secure, Build, Manage. AI extends the existing landing zone; no AI-specific management groups.
- Choose the option with the least operational responsibility that meets every requirement: SaaS agent, then Copilot Studio, then Foundry, then GPUs and containers.
- Start with a single agent unless security boundaries, multiple teams or planned growth require multi-agent. In Copilot Studio, use child agents for one-team grouping and connected agents for separate ownership, settings, ALM and reuse; use A2A for external agents and MCP for tools.
- Know the numbers: 8,000 RPM messages per environment, generative AI RPM/RPH tiers, 500 knowledge sources, 25 SharePoint URLs in generative mode, credit rates (1, 2, 5, 10, 13 per 100 flow actions) and the 125% overage rule.
- Data policies are enforced for all tenants; block specific Copilot Studio connectors (for example, unauthenticated chat) or use endpoint filtering. Foundry quotas are TPM per subscription, region, model and deployment type.
- Extend Microsoft 365 Copilot with declarative agents, connectors and MCP or API plugins when Copilot's orchestrator suffices; build custom engine agents for own models, proactive behaviour, group collaboration, external channels or complex orchestration.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Exam AB-100: Agentic AI Business Solutions Architect
- Analyze requirements for AI-powered business solutions (training module)
- Assess the use of agents in task automation, data analytics, and decision-making
- Review data for grounding accuracy, relevance, timeliness, cleanliness, and availability
- Organize business solution data for AI systems
- Design overall AI strategy for business solutions (training module)
- Implement AI adoption process with Azure
- Design AI agents for business solutions
- Design a multi-agent solution
- Develop use cases for prebuilt Microsoft 365 Copilot agents
- Define solution rules and constraints for AI components
- Determine generative AI knowledge sources for agents built in Copilot Studio
- Determine when to build custom agents or extend Microsoft 365 Copilot
- AI adoption (Cloud Adoption Framework)
- Plan for AI adoption

- Architecture and governance for AI
- Guidance to set up your organization's AI governance process
- AI agent adoption
- Business plan for AI agents
- Technology plan for AI agents
- Data architecture for AI agents
- Single agent or multiple agents
- Knowledge sources summary (Copilot Studio)
- Add other agents overview (Copilot Studio)
- Connect an agent available over the Agent2Agent (A2A) protocol
- Connect to a Microsoft Foundry agent (preview)
- Multi-agent orchestration patterns and best practices
- Quotas and limits (Copilot Studio)
- Billing rates and management (Copilot Studio)
- Configure data policies for agents
- Add Dataverse tables as a knowledge source
- Extend your agent with Model Context Protocol
- Compare declarative and custom engine agents
- Ways to extend Microsoft 365 Copilot
- Declarative agent schema 1.8 for Microsoft 365 Copilot
- MCP and API plugins for declarative agents
- Microsoft 365 Copilot connectors overview
- Get started with Researcher agent
- Agent installation in Microsoft Copilot
- Agents in Microsoft Foundry
- Foundry Agent Service limits, quotas, and regional support
- Manage Azure OpenAI in Microsoft Foundry Models quota
- Default Guardrail policies for Microsoft Foundry
- Agents, Copilot, and AI capabilities in Dynamics 365 apps
- Fabric data agent concepts
- Semantic model best practices for data agent
- Prepare your data for AI to improve Copilot results
- Link your Dataverse environment to Microsoft Fabric
- Create a Dataverse shortcut

DOMAIN 1 · PART B

# Custom AI, prompts, Center of Excellence, and ROI

**Part of Domain 1: Plan AI-powered business solutions (25–30%)**

This chapter completes the *Design overall AI strategy for business solutions* objective group and covers the whole *Evaluate the costs and benefits of an AI-powered business solution* group. Where Chapter 1 asked "which platform and which agent pattern?", this chapter asks the questions a steering committee asks next: do we need our own model or will prompts and grounding do; how do we make prompting repeatable across thousands of users; who governs all of this; how do we design across several Dynamics 365 apps at once; and, above all, is it worth the money?

Expect scenario questions where several answers are technically possible and only one is the *least effort, least operational responsibility* option that still meets the requirement. Microsoft's guidance is consistent on this point: start with prompt engineering, add retrieval for knowledge, fine-tune only with evidence, adopt SaaS before you build, and prove value with a measured baseline. Study the decision tables in this chapter until you can apply them without looking.

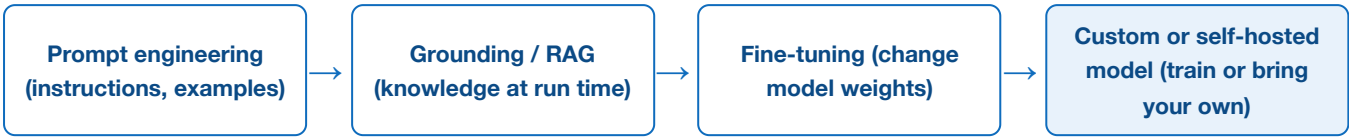
| Official skill (as of October 14, 2026) | Covered in |
|---|---|
| Determine when custom AI models should be created | 2.1.1, 2.1.2, 2.1.3 |
| Develop the use cases for customized small language models for the solution | 2.1.4, 2.1.5 |
| Provide prompt engineering guidelines and techniques for AI-powered business solutions | 2.2.1, 2.2.2 |
| Provide guidelines for creating a prompt library | 2.2.3, 2.2.4 |
| Include the elements of the Microsoft AI Center of Excellence | 2.3.1 to 2.3.4 |
| Design AI solutions that use multiple Dynamics 365 apps | 2.4.1 to 2.4.4 |
| Select ROI criteria for AI-powered business solutions, including the total cost of ownership | 2.5.1 to 2.5.4 |
| Create an ROI analysis for the proposed AI solution for a business process | 2.6.1 to 2.6.4 |
| Analyze whether to build, buy, or extend AI components for business solutions | 2.7.1 to 2.7.3 |
| Implement a model router to intelligently route requests to the most suitable model | 2.8.1 to 2.8.5 |

## 2.1 Custom AI models and small language models

### 2.1.1 The customisation ladder

Microsoft Foundry documentation (Microsoft Foundry was formerly Azure AI Foundry) describes three complementary techniques for adapting a pretrained language model to a task or domain: **prompt engineering**, **retrieval augmented generation (RAG)**, and **fine-tuning**. They are not mutually exclusive, and

production solutions often combine them. Beyond these sits a fourth rung that the exam also cares about: training a **custom model** that is not a large language model at all, such as an AI Builder prediction model or an Azure Machine Learning model trained on your own historical data.




Each rung adds cost, skills, and operational ownership. The Cloud Adoption Framework (CAF) AI adoption guidance states the principle directly: approve the solution that meets requirements with the least operational responsibility, because greater customisation requires more specialised skills and greater operational ownership. The CAF AI maturity table makes the same point from the skills side: fine-tuning a generative model appears at maturity Level 2, RAG apps and training a small model on Azure virtual machines at Level 3, and training and running large models on your own infrastructure at Level 4.

**EXAM TIP**

When a stem says "the model gives answers that are out of date" or "must answer from company policy documents", the answer is grounding (RAG or knowledge sources), not fine-tuning. Foundry guidance is explicit that fine-tuning does not replace retrieval for current information or application-level safety controls.

### 2.1.2 Determine when custom AI models should be created

Treat "custom model" as an evidence-based escalation, not a starting point. Foundry guidance on fine-tuning lists the conditions under which it helps when you have high-quality, task-specific data: improving accuracy on task-specific patterns that need more examples than fit in a prompt; reducing prompt overhead (fewer examples per call lowers token cost and latency); aligning style and structure to a consistent tone, format, or schema; improving tool selection and argument accuracy; teaching the model to use retrieved context effectively; and specialising a smaller model using teacher-generated examples. It also warns that benefits are not guaranteed and that held-out evaluation against a baseline is required.

| Requirement signal | Prompt engineering | RAG / knowledge sources | Fine-tune a foundation model | Custom non-generative model |
|---|---|---|---|---|
| **Need answers from private or frequently changing documents** | Insufficient alone | **Best fit** | Wrong tool (knowledge goes stale, no citations) | Not applicable |
| **Need a consistent tone, format, or JSON schema** | **Start here** (system message, output contract, examples) | No effect on style | Use if prompts become too long or still inconsistent | Not applicable |
| **Prompt carries hundreds of examples; token cost and latency too high** | Already exhausted | Partial (retrieve examples dynamically) | **Best fit**; can also let a smaller model match a larger one | Not applicable |
| **Agent picks wrong tools or fills arguments badly** | Improve tool names, descriptions, instructions first | Not applicable | **Good fit** when descriptions are already good | Not applicable |

| Requirement signal | Prompt engineering | RAG / knowledge sources | Fine-tune a foundation model | Custom non-generative model |
|---|---|---|---|---|
| **Predict a numeric or yes/no outcome from historical tabular data (churn, fraud, late payment)** | Unreliable, non-deterministic | Not applicable | Not appropriate | **Best fit**: AI Builder prediction, Azure Machine Learning, Fabric data science |
| **Extract fields from document layouts unique to the business** | Possible with multimodal prompts | Not applicable | Rarely needed | **Good fit**: AI Builder document processing (custom) or a custom document model in Foundry Tools |
| **Detect your own products in images** | Not applicable | Not applicable | Not applicable | **Best fit**: AI Builder object detection (custom) |
| **Strict rules, repeatable steps, same input must give same output** | Avoid generative AI; the CAF agent business plan recommends regular code or non-generative models for structured, predictable tasks | | | **Code or deterministic model** |

Four further questions settle the decision in practice:

- **Is there a baseline?** Foundry guidance stresses that without a baseline of the un-tuned model's performance you cannot tell whether fine-tuning helped, and fine-tuning on bad data makes the base model worse.
- **Is the data good enough and lawful to use?** Inspect coverage, duplicates, bias, and sensitive information before upload; separate training, validation, and test sets to avoid evaluation leakage.
- **Can we afford to run it?** Fine-tuning has up-front training costs and ongoing hosting costs for the deployed custom model, plus repeated experiments, grading, and evaluation.
- **Can we maintain it?** A fine-tuned adapter is tied to its base model version; when the base model approaches retirement you may need to retrain and redeploy. CAF management guidance asks you to track retirement dates and keep rollback mechanisms.

**COMMON TRAP**

"We need the model to know our product catalogue" is not a fine-tuning requirement. Product facts change; put them in a knowledge source (Dataverse, SharePoint, Azure AI Search) and ground the model. Fine-tune for *behaviour* (format, style, tool use, efficiency), ground for *knowledge*.

### 2.1.3 Custom model options across the Microsoft stack

The architect's job is to pick the lowest-effort platform that can produce the custom model the requirement needs.

| Platform | Custom model types | Typical consumer | Notes for the architect |
|---|---|---|---|
| AI Builder (AI hub in Power Apps and Power Automate) | Custom: document processing, prediction, object detection, category classification, entity extraction PREVIEW. Prebuilt: invoice, receipt, ID, contract, sentiment, translation, and others. | Power Apps, Power Automate, Copilot Studio | Use custom models for data unique to your business; use prebuilt models for scenarios common to all businesses (AI Builder guidance). Low-code, Dataverse-backed, solution-aware. |
| Microsoft Foundry managed fine-tuning | Supervised fine-tuning (SFT), direct preference optimisation (DPO), reinforcement fine-tuning (RFT); distillation through teacher-generated SFT data | Foundry agents, apps, Copilot Studio prompts via bring-your-own-model | Foundry runs the job on managed infrastructure; you supply data, supported hyperparameters, and graders. Method support is model-specific. |
| Foundry interactive training PREVIEW | Custom training loops: SFT, reinforcement learning, preference learning, distillation, custom losses | Experienced ML practitioners | Requires access approval; you own driver, checkpoints, and recovery logic. |
| Foundry Tools (formerly Azure AI services) | Custom document models, custom language and vision models | Pro-code apps and agents | Analytical, non-generative AI with predictable outputs. |
| Azure Machine Learning or Microsoft Fabric | Classical ML trained on your own data | Data science teams | CAF decision tree: use Fabric if you already work there, otherwise Azure Machine Learning. |
| Azure infrastructure (Container Apps serverless GPU, AKS, VMs) | Bring your own open model or runtime | Highly regulated or specialised workloads | CAF: not the default; only when managed platforms cannot provide the control required. |

Foundry also separates **training types** from **deployment types**, and both are governance decisions:

| Training type | Behaviour | Choose when |
|---|---|---|
| Standard | Trains in the Foundry resource's region; data residency guaranteed | Data must stay in a specific region |
| Data zone | Trains within a data zone (currently only the US data zone) | Data must stay within that zone |
| Global | Cheaper; uses capacity beyond your region; data and weights are copied to the training region | Residency is not a constraint and you want faster queues |
| Developer | Uses idle capacity; preemptible; no latency, SLA, or residency guarantees | Experiments and price-sensitive trials |

For serving, fine-tuned models can be deployed serverless (Foundry hosts and bills per token, with hosting charges where applicable), on managed compute PREVIEW with LoRA adapters on dedicated GPUs, or through Fireworks on Foundry PREVIEW. A **Developer** deployment is for evaluation only: it lasts 24 hours and has no availability SLA. Provisioned Throughput deployments give predictable throughput billed in PTUs.

**PRODUCTION NOTE**

For a European customer with residency obligations, "Global" training is usually the wrong answer even though it is cheaper, because training data and weights are copied to wherever training runs. Check the regional support tables before promising a training type for a specific model.

### 2.1.4 Use cases for customised small language models

Azure documentation defines small language models (SLMs) as generally having fewer than 10 billion parameters and large language models (LLMs) as having more. Microsoft's Phi-4 family illustrates the boundary: Phi-4-mini has 3.8 billion parameters, Phi-4 has 14 billion. SLMs are faster and cheaper to train and run, easier to deploy with smaller memory footprints, less prone to overfitting, and easier to interpret. They suit limited data or resources, well-defined narrow tasks, high-precision and low-recall work, and sensitive or regulated tasks where transparency matters. LLMs suit broad, exploratory, or complex reasoning work and noisy inputs.

| Scenario | Prefer SLM (possibly fine-tuned) | Prefer LLM / frontier model |
|---|---|---|
| **Classify incoming emails into 12 fixed case categories at very high volume** | **Yes**: narrow, repetitive, cost-sensitive; fine-tune or distil from a larger model | Over-specified and costly |
| **Field technicians need summarisation on laptops with intermittent connectivity** | **Yes**: on-device with Foundry Local; works offline, no per-token cost | Requires connectivity |
| **Extract a fixed JSON schema from structured supplier messages** | **Yes**: format-bound and predictable | Possible but expensive |
| **Multistep planning across many tools with ambiguous user intent** | Risky | **Yes**: needs reasoning depth and robust tool use |
| **Open-ended research synthesis across many sources** | Insufficient breadth | **Yes** |
| **Mixed traffic: mostly simple prompts, some hard ones** | Consider **model router** (Section 2.8) rather than choosing one model for everything | |

There are three routes to a "customised SLM" in the Microsoft stack:

- **Distillation in Foundry.** Generate high-quality outputs from a large "teacher" model, then use them as SFT data for a smaller "student" model. Foundry documents distillation through teacher-generated SFT data as a managed fine-tuning pattern. The Foundry customisation article gives the canonical example: an IT department fine-tuned GPT-4o mini on hundreds of natural-language-to-SQL examples and obtained better results than the base model at lower cost and latency than GPT-4o.
- **Fine-tune an open SLM** from the Foundry catalogue where the supported-models table allows it (the list includes small models such as Ministral-3B for SFT; always check the current table, because support varies by model, method, training type, and region).
- **Run an SLM at the edge.** Foundry Local is an on-device runtime with a curated catalogue (including Phi models) and an OpenAI-compatible API. Data never leaves the device, it works offline, there are no per-token costs, and no Azure subscription is required. It runs on Windows, macOS (Apple silicon), and Linux, but it is designed for single-user devices, not as a multi-user server inference stack. For server-side SLM hosting with minimal operations, CAF suggests Azure Container Apps with serverless GPU.

SLMs from the Foundry catalogue can also back a Copilot Studio prompt: the bring-your-own-model feature lets makers connect a Foundry model deployment (chat completions endpoint) to a prompt, and the documentation lists Phi multimodal and vision models among models that accept image input. The connection is governed by the "Azure AI Foundry" connector in Power Platform data policies.

**COMMON TRAP**

Do not assume every capability exists on every model. Copilot Studio documents that the GPT-5 family and later models are not currently supported for bring-your-own-model in prompts, and that the connection must target a `/chat/completions` endpoint (a Responses API endpoint fails). Similarly, check model cards for tool calling and context length before selecting an SLM for an agent.

### 2.1.5 Governing custom models and SLMs

- **Baseline and quality gates.** CAF management guidance requires performance criteria to promote trained, fine-tuned, and retrained models between environments, version control, and rollback for underperforming versions.
- **Golden datasets.** Maintain representative golden datasets to test models consistently across environments and versions.
- **Sensitivity changes.** When data is reclassified (for example with Microsoft Purview labels), identify every model trained on it and retrain without the reclassified data.
- **Lifecycle.** Monitor base-model retirement and plan retraining; an adapter is not interchangeable across similarly named base models.
- **Safety.** Keep application-level safety controls (content filters, prompt shields, human review) even after fine-tuning, and evaluate the final serving runtime.

## 2.2 Prompt engineering and prompt libraries

### 2.2.1 Prompt engineering techniques

Prompt engineering is the starting point for every generative AI solution and the cheapest lever to pull. Foundry describes a prompt in terms of components: **instructions**, **primary content**, **examples**, **cue**, and **supporting content**. In chat APIs these are carried by system, user, and assistant messages.

| Technique | What it does | When to recommend it |
|---|---|---|
| Clear, specific instructions first | States the task before context and examples | Always; be specific, restrict the operational space |
| Few-shot learning | Includes example input/output pairs in the prompt | To prime format and behaviour without training |
| Repeat instructions at the end | Counters recency bias in long prompts | Long grounding content between instruction and answer |
| Prime the output (cue) | Ends the prompt with the start of the desired output | Force a list, a single query, or a fixed opening |
| Clear syntax | Separators, headings, Markdown or XML sections | Multi-part prompts; makes outputs parseable |

| Technique | What it does | When to recommend it |
|---|---|---|
| Break the task down | Splits work into explicit steps | Extraction followed by generation; complex tasks |
| Chain-of-thought prompting | Asks the model to reason step by step | **Non-reasoning models only**; not for reasoning models |
| Specify output structure and citations | Defines a schema; asks for inline citations | Reduces fabrication; inline citations work better than end-of-text citations |
| Provide grounding context | Supplies source data the answer must use | Any non-creative business answer |
| Give the model an "out" | For example "respond with 'not found' if the answer isn't present" | Reduce invented answers |
| Temperature and top_p | Control randomness | Lower for legal or factual outputs, higher for creative work |
| Space efficiency | Tables instead of verbose JSON, careful whitespace | Large context payloads |

**COMMON TRAP**

The Foundry prompt engineering techniques article states that these techniques are not recommended for reasoning models such as gpt-5 and the o-series, and chain-of-thought prompting is applicable only to non-reasoning models. Attempting to extract model reasoning by other means than the reasoning summary parameter is unsupported and may violate the Acceptable Use Policy. A guideline that says "always ask the model to think step by step" is therefore wrong for a reasoning model.

For the **system message** (system prompt, metaprompt), Foundry's design checklist is: start with the assistant's job; define boundaries; specify the output format (a small, stable output contract); add a "when unsure" policy for ambiguous, out-of-scope, or unanswerable requests; and test with realistic and adversarial prompts. Common pitfalls are conflicting instructions, overly long system messages that consume context, and hidden requirements. A system message influences but does not guarantee behaviour, so layer it with filters and evaluation.

**Listing 2.1: A system message with role, scope, output contract, and fallback**

```
You are the Contoso returns assistant for retail customers.
Scope: answer only questions about returns, refunds, and exchanges
using the RETURNS POLICY section below. Do not give legal advice.
Output: plain text, at most 120 words, then "Source:" with the
policy clause number used.
If the policy does not cover the question, say you can't confirm
and offer to create a case. Never invent clause numbers.
---
RETURNS POLICY
{retrieved_policy_chunks}
---
```

### 2.2.2 Prompt guidance by surface

An architect issues different guidance to different audiences, because each surface handles prompts differently.

| Surface | Who writes the prompt | Guidance to give |
|---|---|---|
| Microsoft 365 Copilot and Copilot Chat | End users | Use the four elements: **goal** (what you want), **context** (why, who is involved), **source** (which files, data, or web), **expectations** (format, audience, tone). Iterate with follow-ups; check citations. Reference files with `/`, meetings or topics with `#`, people with `@` (organisational references need a Microsoft 365 Copilot licence). |
| Copilot Studio agent instructions (generative orchestration) | Makers | Ground instructions in tools and knowledge actually configured; reference objects with `/` using exact names; number steps for multistep work; add guardrails for when not to respond; specify response format; keep instructions short. Put precision into tool and topic *descriptions*, which the orchestrator uses to select them. Instructions cannot change search retrieval logic, Adaptive Card triggering, or the fallback message (edit the Fallback system topic instead). |
| Prompts (prompt builder) in Copilot Studio, Power Apps, Power Automate | Makers | Reusable prompts with input variables and knowledge (for example Dataverse) supplying run-time context; pick the model (managed models or Foundry models); test before sharing; human review for consequential output. |
| Foundry agents and apps | Developers | System message checklist, few-shot, output contracts, grounding, evaluation in CI; model-specific guidance for reasoning models. |

**PRODUCTION NOTE**

Copilot Studio guidance warns that triggers on autonomous agents can carry jailbreak instructions in the payload. Instructions should limit which tools may run after reading knowledge and constrain tool parameters (for example, only email a specified list of recipients). This is prompt engineering as a security control.

### 2.2.3 Guidelines for creating a prompt library

A prompt library turns individual prompting skill into an organisational asset. Microsoft's agent-value guidance lists "build shared prompt libraries and update them monthly based on the themes observed in Copilot Studio Analytics" among the adoption levers that move ROI. There are three places to host a library, matched to the three prompt audiences:

| Library | Content | Governance features | Use when |
|---|---|---|---|
| **Organizational prompts (Microsoft 365 admin center > Copilot > Prompts)** | Curated end-user prompts shown in Copilot Chat, Microsoft Edge, and Microsoft Teams | Managed by AI Administrator or Search Editor; up to 1,000 per tenant; about 3 hours to appear; pin up to four as suggested prompts; bulk import by UTF-8 CSV (up to 100 prompts and 5 MB per file); export; analytics (active users, submissions over 7, 14, or 28 days); no automatic translation | Tenant-wide, approved prompts for Microsoft 365 Copilot users |
| **Prompt Gallery personal and team prompts (shown in the prompt lab)** | Microsoft-curated, user-saved, and team-shared prompts | Stored in Microsoft 365 Substrate collections in the tenant boundary; admins can export prompts users saved, liked, or shared | Grass-roots sharing within teams |

| Library | Content | Governance features | Use when |
|---|---|---|---|
| **Prompts in Power Platform / Copilot Studio** | Reusable maker prompts used as agent tools, flow actions, or app functions | Private by default; share as **User** (run only) or **Co-owner** (edit, share, delete); stored in Dataverse, solution-aware for ALM | Prompts embedded in business processes |

Guidelines to give the CoE and prompt owners:

1. **Organise by task and department.** The organisational prompt schema already has *Task type* (for example Analyze, Create, Edit) and *Department*; use them as the taxonomy so users can filter.
2. **Write prompts to the four elements** and include placeholders the user must complete, because selecting an organisational prompt places it in the input box without sending it.
3. **One prompt per language.** Automatic translation is not available, so create localised variants deliberately.
4. **Assign an owner and a review date** to every entry (use the internal Description field, which users do not see).
5. **Test before publishing** against representative data and the Copilot surfaces listed in *Supported apps*. For maker prompts, the Copilot Agent Kit Prompt Advisor can score a draft and suggest optimised versions.
6. **Measure and prune.** Use the analytics to retire unused prompts and promote high performers; refresh monthly from analytics themes.
7. **Keep secrets and personal data out of prompts**, and do not encode access decisions in prompts; permissions come from the data source.
8. **Version maker prompts with ALM.** Promote Power Platform prompts in solutions through dev, test, and production rather than editing in production.

**EXAM TIP**

If a question asks how to make approved prompts appear as suggestions for all Copilot Chat users, the answer is organizational prompts in the Microsoft 365 admin center (with pinning for the top four), not Teams channel sharing and not a SharePoint list. If it asks who can manage them, think AI Administrator or Search Editor.

### 2.2.4 Prompt library template and lifecycle

| Field | Example | Purpose |
|---|---|---|
| ID and title (≤ 35 characters for org prompts) | SALES-012 Account brief | Unique reference; label on prompt cards |
| Display prompt (≤ 132 characters) | Brief me on [account] before my meeting | Short text for pills and autosuggest |
| Prompt text (≤ 8,000 characters) | Goal, context, source, expectations with [placeholders] | The prompt sent to Copilot |
| Task type / department / language | Analyze / Sales / en-GB | Filtering and localisation |
| Surface and licence prerequisite | Copilot Chat (work); requires Microsoft 365 Copilot licence for CRM and email grounding | Prevents broken experiences |

| Field | Example | Purpose |
|---|---|---|
| Grounding sources | Outlook, Teams meetings, Sales in Microsoft 365 Copilot | Documents data dependency |
| Model notes | Tested on current default model; do not add chain-of-thought instructions | Model-specific behaviour |
| Risk rating and human-review rule | Medium: output shared externally only after review | Responsible AI control |
| Owner, version, review date | Sales enablement lead, v1.3, quarterly | Accountability |
| Success metric | Submissions per active user; thumbs-up rate | Prune or promote |




## 2.3 The Microsoft AI Center of Excellence

The CAF defines an AI Center of Excellence (AI CoE) as an internal team of experts who drive successful and valuable AI outcomes, prevent fragmented or ungoverned adoption, and provide business and technical consultation. The CAF AI adoption guidance places central ownership of organisation-wide AI policies and standards (including Responsible AI requirements) with such a function, while business groups pursue opportunities within those boundaries. The Microsoft Learn AI Center of Excellence learning path adds workload operations guidance and AI landing zones.

### 2.3.1 Building the AI CoE team

1. **Secure executive sponsorship.** Sponsorship provides budget, authority, and credibility; form a steering committee of business and IT leaders with monthly reviews and direct access to C-level decision makers.
2. **Appoint an AI CoE leader** as the single point of contact for AI strategy implementation.
3. **Assemble a multidisciplinary team.** Business leaders identify use cases and data and judge effectiveness; technical experts handle data management, model design, training, adaptation, and selection; include data scientists, ML engineers, AI governance experts, AI security specialists, and AI operations professionals.
4. **Decide placement.** If a Cloud Center of Excellence already exists, integrate AI practices into it. Create a standalone AI team only if existing teams cannot support AI adoption or critical risks exist.
5. **Define the operating model.** Start centralised to consolidate expertise; move to an advisory model as maturity grows.

### 2.3.2 Responsibilities of the AI CoE

| Area | What the AI CoE does | Typical AB-100 artefact |
|---|---|---|
| Define AI strategy | Align AI with business goals, identify use cases with business leaders, use the Microsoft AI decision tree, develop Responsible AI policies | AI charter, decision tree for SaaS vs Copilot Studio vs Foundry |

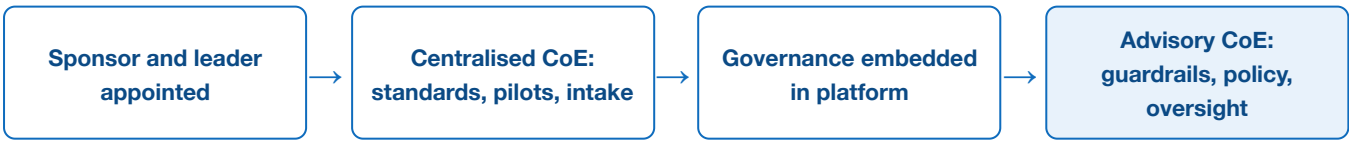
| Area | What the AI CoE does | Typical AB-100 artefact |
|---|---|---|
| Develop AI skills | Assess skills, provide learning pathways and hands-on experimentation | Role-based enablement plan, maker certification |
| Lead pilot projects | Run proofs of concept prioritised by impact and feasibility; feed results back into processes | PoC charter with go/no-go metrics |
| Define and enforce AI standards | Governance and security standards for data quality and model lifecycle; bias and transparency reviews; compliance audits | Environment strategy, data policies, model promotion gates |
| Intake and prioritisation | Structured intake with consistent criteria for value, feasibility, and resources; prioritised backlog | Use-case intake form and scoring grid |
| Develop reusable assets | Compliance checklists and shared assets on an internal platform | Prompt library, agent templates, reference architectures |
| Measure and report outcomes | KPIs such as adoption rates, compliance levels, cycle times; regular reporting to leadership | Executive value scorecard (Section 2.5) |
| Manage AI services (optional) | Operate and govern deployed AI services and models; templates and code repositories | Model registry, approved model list, router policy |

### 2.3.3 Roles, responsibility model, and evolution

The Learn module *Introduction to the AI Center of Excellence* frames a three-way responsibility model: the **AI CoE** sets strategy, standards, guardrails, reusable patterns, intake and prioritisation, and skilling; **platform teams** provide governed foundations (identity, networking, policy, observability, approved services); **workload and business teams** own use cases, domain data, integration, delivery, and business outcomes. Capabilities to staff, either as dedicated roles or across virtual teams, include an executive sponsor (an existing CIO, CTO, CDO, or a Chief AI Officer are alternative archetypes, not mandatory titles), business and product owners, AI application and agent engineers, data engineers and stewards, Responsible AI and risk specialists, AI security engineers (prompt injection, jailbreak, red teaming), evaluation and agent-optimisation specialists, and AI operations (GenAIOps, LLMOps, AgentOps).

Note that the module treats **prompt engineering as a cross-cutting capability** across engineering, product, design, and business roles rather than as a durable standalone role, with prompt patterns validated by evaluations and safety checks before reuse. That aligns with the prompt library guidance in Section 2.2.

The CAF describes when to shift from centralised control to advisory: watch for approval delays, knowledge bottlenecks, and teams debating priorities with the CoE instead of delivering value; embed AI delivery and governance into platform operations; then replace the gatekeeper model with an advisory group that sets guardrails while frontline teams own delivery.




**COMMON TRAP**

An answer option that creates a brand-new, separate AI organisation alongside an existing Cloud CoE is usually wrong. The CAF recommends integrating AI expertise into the existing CCoE unless current teams cannot support AI adoption or critical risks exist.

### 2.3.4 Connecting the AI CoE to Power Platform and Copilot Studio governance

In a business-applications estate, the AI CoE must work with the Power Platform CoE that already governs makers, environments, and data policies. Microsoft describes a Power Platform CoE as a team that balances innovation and control, drives cultural change, scales citizen development, and connects low-code initiatives to measurable outcomes and ROI. Tooling has moved: the **Power Platform CoE Starter Kit is no longer actively maintained**, and its core capabilities are part of the Power Platform admin center (managed environments, tenant-level analytics, data policies). Start with these supported, out-of-the-box capabilities.

For agents specifically, the **Copilot Agent Kit** (formerly Copilot Studio Kit) is maintained by the Power Customer Advisory Team and adds tools such as Agent Inventory (tenant-wide view of custom agents), Compliance Hub (evaluates agent configurations against configured controls, with manual review, quarantine, or delete actions), Agent Review Tool (anti-pattern checks), Agent Value (classifies agents by type, behaviour, and value), Agent Library (reusable agent templates), Power Shield (approval workflow for connector access in data policies), Conversation Analyzer, rubrics refinement for evaluating AI responses, and Prompt Advisor (scores a prompt and suggests optimised versions). Copilot Studio governance principles include ALM across dev, test, and production, connector governance, and environment-level policies (data policies, role-based access, auditing).

| Element of a Microsoft AI CoE | Microsoft capability that implements it |
|---|---|
| Strategy and decision rights | CAF AI adoption and AI agent adoption guidance; AI decision tree |
| Guardrails for makers | Managed environments, data policies, connector governance, sharing limits |
| Guardrails for pro-code | Azure Policy for model deployment, approved model subsets, content filters |
| Inventory and review | Power Platform admin center analytics; Copilot Agent Kit Agent Inventory, Compliance Hub, and Agent Review Tool |
| Reusable assets | Organisational prompts, shared Power Platform prompts, agent templates in solutions |
| Value reporting | Copilot Studio analytics and savings, Viva Insights Copilot reports (Section 2.5.4) |

## 2.4 AI solutions that span multiple Dynamics 365 apps

### 2.4.1 The shared platform

Dynamics 365 Sales, Customer Service, Field Service, Customer Insights, and Project Operations are model-driven apps on Microsoft Dataverse. Because of this shared foundation, Copilot and AI capabilities available in model-driven apps are automatically available across them: Microsoft 365 Copilot in model-driven apps PREVIEW, agent feed PREVIEW for supervising agent activity, the Power Apps MCP Server PREVIEW, form fill assistance, natural-language find and visualise data in views, row summaries, and timeline highlights. Some features need to be enabled per app.

Finance and operations apps (such as Finance, Supply Chain Management, and Commerce) share their own capabilities: generative help and guidance, workflow history summaries, chat with finance and operations data PREVIEW, Immersive Home PREVIEW, and the **Dynamics 365 ERP MCP server**, which gives agents a framework to perform data operations and access ERP business logic.

Designing across apps therefore means designing **one data and identity plane** (Dataverse plus ERP), **one agent supervision surface** (agent feed, Microsoft 365 Copilot, Teams), and **clear handoffs** between app-specific agents.

### 2.4.2 Prebuilt agents by app

| App | Agents and AI features to compose |
|---|---|
| Dynamics 365 Sales | Sales Qualification Agent, Sales Opportunity Agent, Sales Close Agent, Sales Research Agent, Recommended Actions Agent, AI-powered data enrichment; Sales agent in Microsoft 365 Copilot |
| Dynamics 365 Customer Service / Contact Center | Case Management Agent, Customer Knowledge Management Agent, Quality Evaluation Agent, Customer Intent Agent (and for voice), intent-driven routing, rollout manager, agent insights dashboard |
| Dynamics 365 Field Service | Scheduling Operations Agent |
| Dynamics 365 Project Operations | Time Entry Agent, Expense Agent, Approvals Agent (production ready preview) |
| Dynamics 365 Finance | Account Reconciliation Agent (production ready preview) |
| Dynamics 365 Supply Chain Management | Supplier Communications Agent (production ready preview) |
| Copilot apps | Sales in Microsoft 365 Copilot and Service in Microsoft 365 Copilot (the exam outline uses the names Microsoft 365 Copilot for Sales and for Service): CRM-enriched email summaries and replies, opportunity and case summaries, saving Outlook activities to CRM |

### 2.4.3 Integration patterns for cross-app AI

| Pattern | What it does | Use for AI when | Watch out for |
|---|---|---|---|
| **Dual-write** | Out-of-box, tightly coupled, near-real-time, bidirectional sync between finance and operations apps and Dataverse (for example Customer to Account) | Overlapping master data (accounts, products, sales orders) must exist locally in both apps and be grounding data for CRM-side agents | One-to-one mapping between a finance and operations environment and a Dataverse environment; schema changes such as company and party concepts |
| **Virtual entities (virtual tables)** | Expose finance and operations data in Dataverse without copying it | Read access to ERP reference data as an agent knowledge source | Latency and query patterns; not a copy for offline analytics |
| **Fin & Ops Apps (Dynamics 365) connector** | Create, read, update, delete on ERP entities from agents | Copilot Studio agents that act in ERP (create vendor, update credit limit, trigger processes) | Least-privilege identity, approval for writes |
| **Dynamics 365 ERP MCP server** | MCP tools for ERP data operations and business logic | Pro-code or Copilot Studio agents that need standardised tool access | Tool governance, auditing |

| Pattern | What it does | Use for AI when | Watch out for |
|---|---|---|---|
| **Business and data events** | Event notifications from finance and operations apps | Triggering autonomous agents from ERP state changes | Trigger payload hardening against injection |

Chat with finance and operations data PREVIEW documents both knowledge routes explicitly: add virtual entities as knowledge sources, or synchronise data to native Dataverse tables with dual-write and use those tables as knowledge. The resulting knowledge can serve Copilot for finance and operations apps, custom agents, Microsoft 365 Copilot, or other agents that support knowledge sources.

### 2.4.4 Worked design: service-to-cash across four apps

Scenario: a manufacturer wants a customer complaint about a faulty machine to be triaged, fixed in the field, invoiced correctly, and fed back to sales, with minimal manual coordination.




- **Prefer prebuilt agents per app**, then orchestrate the gaps with a Copilot Studio agent or agent flow (for example, a parts-check tool using the Fin & Ops Apps connector).
- **Ground on one customer identity.** Dual-write the Customer/Account and product master; expose inventory and price lists as virtual entities rather than copying them.
- **Supervise in one place.** Use agent feed in the model-driven apps so humans can review and take over agent actions; route exceptions to queues.
- **Secure each hop.** Each agent acts with its own identity and least privilege; ERP writes require approval steps.
- **Test end to end.** Cross-app scenarios need end-to-end test cases that span apps and agents (covered in the Deploy domain chapters).
- **Measure per stage.** Case resolution time, first-time fix rate, days sales outstanding, and churn signals give an ROI chain from service to revenue.

**EXAM TIP**

If the requirement is "sales staff and CRM agents must work with customer master data locally in Dataverse, and edits must flow back to Finance in near real time", the answer is dual-write for the customer master. If the requirement is "show ERP inventory to a service agent without copying data", choose virtual entities.

## 2.5 Selecting ROI criteria, including total cost of ownership

### 2.5.1 Define value before you build

Microsoft's Copilot Studio guidance on measuring agent business value starts before any configuration with four discovery questions, answered in sequence:

1. **Strategic goal:** a board-visible objective such as revenue growth, cost optimisation, customer experience, or compliance posture.
2. **Stakeholder map:** sponsor, operator, end user, and governor, each with a different KPI; resolve conflicts before launch.
3. **Success metrics:** a quantitative before-state, for example "handle time is 8 minutes, measured over 90 days, with a P90 of 16 minutes". "Our handle time is too long" is not a baseline.
4. **Build readiness:** only with a shared definition of success and a documented baseline should technical work start.

Use cases are then placed on an impact and effort grid. **Impact** is the projected annual value at target adoption, estimated from volume touched, share resolved or accelerated, unit value per task, and an attribution discount. **Effort** is build cost plus change burden (integration complexity, knowledge depth, orchestration pattern, enablement size).

| Quadrant | Action | Funding |
|---|---|---|
| Quick wins (high impact, low effort) | Build first; three to five in parallel so measurement has data within two quarters | Operating budget |
| Strategic investments (high impact, high effort) | One or two at a time, after a quick win reaches baseline | Dedicated programme funding; these justify an AI CoE |
| Nice to have (low impact, low effort) | Citizen-maker community or internal agent store | No central funding |
| Reconsider (low impact, high effort) | Park; revisit in six months | None |

The CAF agent business plan offers a complementary 1–5 scoring across **business impact** (strategy alignment, business value, change management timeframe), **technical feasibility** (risks, safeguards, technology fit), and **user desirability** (personas, value proposition, change resistance), and recommends using business metrics as go/no-go decision gates at each stage.

**PRODUCTION NOTE**

Microsoft's guidance asks you to store at least 90 days of operational data as the baseline, captured in a telemetry system, ticketing export, or ERP extract. Surveys can supplement a baseline but cannot replace it. Capture volume by category and channel, median/P90/P99 cycle time, error rate by type, fully loaded cost per transaction, a voice-of-customer signal, and productive-hour value.

### 2.5.2 ROI criteria: the four value drivers

| Value driver | What it measures | How to price it | Example metrics |
|---|---|---|---|
| Efficiency | Productive hours returned and reinvested | Hours returned × fully loaded hourly value | Hours saved, Agent Assisted Hours, cycle time, touchless rate, cost per transaction |
| Quality | Error reduction, consistency, compliance | (Error rate before – after) × volume × cost per error | Resolution rate, first-contact resolution, escalation and abandon rates, groundedness |

| Value driver | What it measures | How to price it | Example metrics |
|---|---|---|---|
| Revenue | Top-line lift from retained, expanded, or new business | Conversion or deflection delta × volume × unit revenue × attribution discount | Conversion lift, retention delta, cross-sell rate, advisor capacity |
| Strategic | Decision velocity, employee confidence, optionality, resilience | Option premium, talent retention value, resilience value | New capabilities shipped, workflows redesigned, employee sentiment on AI |

Pair each driver with a **leading** indicator (for example routine adoption at 30 days, rubric-graded groundedness in testing, containment trend) and a **lagging** indicator (annualised ROI, error-rate delta, retention delta, EBIT contribution). Microsoft warns against three post-launch failure patterns: measurement that stops at pilot, activity metrics that do not tie to outcomes, and the **time-savings trap**, claiming value from theoretical time savings without a chain of evidence from adoption through operational KPIs to business outcomes.

Fully loaded hourly cost is typically base salary multiplied by 1.3 to 1.5 to account for benefits, payroll taxes, tools, training, and overhead. Where role-specific data is missing, Microsoft's Copilot Studio agents report in Viva Insights uses a default of $72 per hour.

### 2.5.3 Total cost of ownership components

CAF cost planning asks you to estimate costs from a defined architecture (services, tiers, regions), include operational costs such as skills and process change, validate with test deployments, and record a cost baseline. The Microsoft 365 Copilot extensibility planning guidance lists the cost categories to record for any agent or plugin. Combined, they give the TCO checklist below.

| TCO category | Components | Microsoft-specific cost driver |
|---|---|---|
| Licences | User and admin licences for target experiences | Microsoft 365 Copilot add-on vs Copilot Chat; Copilot Studio licence or capacity; Dynamics 365 app licences and add-ons |
| Consumption | Metered agent, connector, API, and model usage | Copilot Credits (classic answer 1, generative answer 2, agent action 5, tenant graph grounding 10, agent flow actions 13 per 100 actions); Foundry token or PTU billing; premium reasoning tokens |
| Build | Design, development, integration, testing, data preparation | Connectors, MCP servers, dual-write maps, knowledge clean-up |
| Hosting and platform | APIs, apps, databases, remote MCP servers | Custom engine agents incur hosting (Foundry, App Service, Bot Service); declarative agents are hosted by Microsoft 365 Copilot |
| Security and operations | Identity, secrets, networking, storage, monitoring, audit | Private networking for Foundry standard setup, Application Insights, Purview |
| Model customisation | Training runs, grading, evaluation, hosting of fine-tuned models | Foundry training type and deployment type |
| People and change | Training, adoption, CoE time, process redesign | Role-based enablement, manager-first training |

| TCO category | Components | Microsoft-specific cost driver |
|---|---|---|
| Run and evolve | Support, incident response, evaluation, servicing, deprecation, retirement | Model retirements, prompt library upkeep, agent decommissioning |

**COMMON TRAP**

Employee-facing Copilot Studio usage is included at no Copilot Credit charge when the user holds a Microsoft 365 Copilot licence and the agent runs with that user's authenticated identity (subject to fair use and with exceptions such as computer-using agents). **Customer-facing** agents and autonomous agents triggered by events are not covered this way, so their Copilot Credits belong in the TCO. Also note that agent flows triggered other than by "When an agent calls the flow" consume credits at the standard rate.

Two more facts shape cost risk. Copilot Studio enforces overage when a tenant reaches **125%** of prepaid capacity by disabling custom agents (ongoing conversations finish), so the TCO model must include capacity headroom or a pay-as-you-go meter. And Copilot Studio rates exclude bring-your-own-model configurations such as Foundry models, which are billed separately in Azure. Forecast credits with the Copilot Studio agent usage estimator.

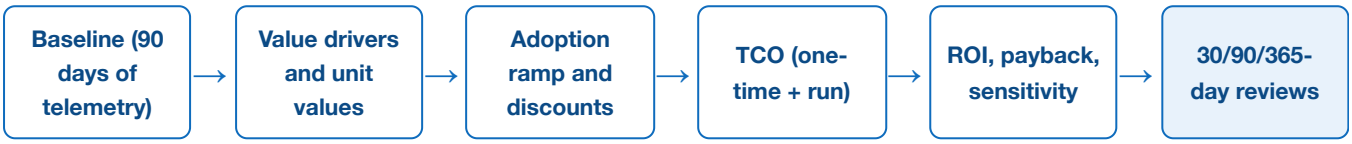
### 2.5.4 Where to read the numbers

| Surface | What it gives you |
|---|---|
| Copilot Studio analytics (Monitor page) | Engagement, resolution, escalation, abandon rates, CSAT, thumbs-up/down feedback |
| Copilot Studio Savings calculator | Time or cost saved **per run** (quick estimate, typical for autonomous agents) or **per tool** (detailed, for conversational agents), compared with another method; shown on the Monitor tab. Admins can turn off cost savings with the environment-level *Disable money saving rules* setting in the Power Platform admin center; time savings still work. |
| Copilot Studio agents report (Viva Insights) | Agent Assisted Hours and Agent Assisted Value using Microsoft's published formula |
| Copilot Dashboard (Viva Insights) | Microsoft 365 Copilot adoption, including routine adoption |
| Copilot business impact report (Viva Insights) | Links Copilot usage to business outcome metrics such as revenue indicators |
| Viva Glint Copilot Impact Survey | Employee sentiment (strategic driver) |
| Azure Cost Management | Actual Foundry and Azure spend, budgets, and alerts at multiple thresholds |
| Power Platform admin center | Copilot Credit consumption by environment and agent; per-agent monthly limits |

For conversational agents, the formula is: **Agent Assisted Hours = (knowledge references + weighted sessions without knowledge references) × time savings multiplier ÷ 60**, where each knowledge reference counts once, sessions without references are weighted 1.0 if resolved and 0.7 if escalated or abandoned, and the default multiplier is 6 minutes. **Agent Assisted Value = Agent Assisted Hours × hourly rate**. For autonomous agents, the variant sums knowledge references × retrieval savings, action time savings, and successful sessions without actions × generic savings, all divided by 60.

## 2.6 Creating an ROI analysis for a business process

### 2.6.1 Method




Use a simple, auditable definition: **ROI = (total benefits – total cost of ownership) ÷ total cost of ownership** over an agreed horizon (three years is common for platform investments). Report payback period alongside ROI, and show sensitivity to the two or three assumptions that matter most. CAF timeline guidance helps set expectations: Microsoft Copilots typically deliver ROI in days to weeks, custom Azure AI and machine learning workloads take several weeks to months to reach production, and you should add 20–30% contingency to initial estimates.

### 2.6.2 Worked example: Tier-1 customer inquiries

> **NOTE**
> All figures in this example are **illustrative assumptions** chosen to teach the method. They are not Microsoft prices or benchmarks. Always take current licence and consumption rates from the Copilot Studio licensing guide, billing rates page, and Azure pricing.

**Scenario.** Contoso handles 30,000 Tier-1 customer inquiries per month in Dynamics 365 Customer Service. Proposed solution: a customer-facing Copilot Studio agent on the website and messaging channels, grounded on returns and product knowledge, with a case-creation tool and handoff to representatives; representatives use Copilot case summaries for the cases that reach them. Customer Service licences are already owned.

| Baseline measure (90 days) | Value |
|---|---|
| Tier-1 inquiries per month | 30,000 |
| Median handle time (P90) | 8 minutes (20 minutes) |
| Fully loaded representative cost | $60 per hour (base × 1.4) |
| Misrouted cases | 6% of inquiries; $12 rework cost each |

| Benefit line (steady state) | Calculation | Annual value |
|---|---|---|
| A. Self-service resolution (efficiency) | 30,000 × 35% target resolution = 10,500 per month; × 8 min = 1,400 h per month; × 12 = 16,800 h; × $60 = $1,008,000; × 60% recapture (capacity actually redeployed or hiring avoided) | $604,800 |
| B. Assisted handling (efficiency) | 19,500 remaining cases × 1.5 min saved by summaries = 487.5 h per month; × 12 = 5,850 h; × $60 = $351,000; × 60% recapture | $210,600 |
| C. Fewer misroutes (quality) | Misroutes fall from 6% to 3%: 900 fewer per month × $12 × 12 = $129,600; × 75% attribution (intent-based routing also contributes) | $97,200 |
| **Total annual benefit at full adoption** | | **$912,600** |
| Year 1 at 50% adoption ramp | | $456,300 |

| TCO line | Assumption | Cost |
|---|---|---|
| **One-time** | | |
| Design, build, integration, testing | Partner plus internal team | $180,000 |
| Knowledge preparation | Clean, de-duplicate, label returns and product content | $40,000 |
| Change management and training | Representatives, supervisors, managers first | $35,000 |
| Responsible AI impact assessment and security review | Including prompt-injection testing | $15,000 |
| **One-time subtotal** | | **$270,000** |
| **Recurring (per year)** | | |
| Copilot Credits | 30,000 sessions per month × (3 generative answers × 2 + 1 agent action × 5 = 11 credits) = 330,000 per month = 3,960,000 per year × assumed $0.01 per credit | $39,600 |
| Foundry model usage | Summarisation calls through a model router deployment | $18,000 |
| Agent operations | 0.5 FTE AgentOps (monitoring, tuning, incidents) | $60,000 |
| Knowledge curation | 0.25 FTE content owner | $25,000 |
| Evaluation and prompt library upkeep | Test sets, rubric refresh | $12,000 |
| Monitoring and logging | Application Insights, Log Analytics | $6,000 |
| **Recurring subtotal** | | **$160,600** |

| Three-year view | Year 1 | Year 2 | Year 3 | Total |
|---|---|---|---|---|
| Benefits | $456,300 | $912,600 | $912,600 | $2,281,500 |
| TCO | $430,600 | $160,600 | $160,600 | $751,800 |

| Three-year view | Year 1 | Year 2 | Year 3 | Total |
|---|---|---|---|---|
| Net benefit | $25,700 | $752,000 | $752,000 | $1,529,700 |
| Cumulative net | $25,700 | $777,700 | $1,529,700 | |

**Results.** Three-year ROI = $1,529,700 ÷ $751,800 ≈ **203%**. Payback: in year 1 the average monthly benefit is $38,025 and the monthly run cost is about $13,383, so the net monthly gain of about $24,642 recovers the $270,000 one-time cost in roughly **11 months**.

### 2.6.3 Sensitivity and a cross-check with Agent Assisted Hours

| Change one assumption | Three-year net benefit | ROI |
|---|---|---|
| Base case | $1,529,700 | 203% |
| Self-service resolution 20% instead of 35% | $881,700 | 117% |
| Recapture 40% instead of 60% | $850,200 | 113% |
| Credit consumption doubles (more generative turns per session) | $1,410,900 | 162% |

The model is most sensitive to resolution rate and recapture, so those become the leading indicators in the 30-day and 90-day reviews, and the decommission criterion (for example "resolution below 15% after two quarters") is written into the agent record at build time.

Now cross-check with Microsoft's telemetry-based formula for one month at target adoption. Suppose 12,000 of the 30,000 sessions cite knowledge (1.5 references each, 18,000 references), and the other 18,000 split into 10,000 resolved and 8,000 escalated or abandoned. Weighted sessions = 10,000 × 1.0 + 8,000 × 0.7 = 15,600. Agent Assisted Hours = (18,000 + 15,600) × 6 ÷ 60 = **3,360 hours**; at the $72 default rate, Agent Assisted Value is $241,920 per month.

> **COMMON TRAP**
> The Agent Assisted Hours figure is far higher than the bottom-up business case because it applies a standard 6-minute multiplier per knowledge reference, a default hourly rate, and no recapture or attribution discount. It is a consistent telemetry indicator for comparing agents and trends; it is not a substitute for a finance-validated baseline. Customise the multiplier and rate in the report calculator, and never add the two figures together.

### 2.6.4 Presenting and sustaining the case

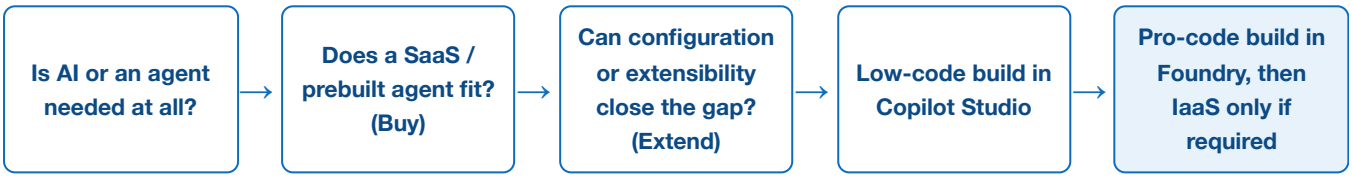
- **Executive scorecard:** one line per value driver with baseline, target, actual, leading and lagging indicator.
- **Governance before build:** named accountable executive, Responsible AI impact assessment, data residency and sensitivity labels, visible escalation path, scheduled 30/90/365-day value reviews, and a decommission criterion.
- **Adoption levers:** role-based training with managers first, anchoring to a named high-volume workflow, executive role modelling, shared prompt libraries refreshed monthly, visible guardrails, and a quarterly expansion rhythm.

## 2.7 Build, buy, or extend AI components

### 2.7.1 The decision sequence

The CAF AI agent decision tree asks one primary question first: **does a SaaS agent meet your functional requirements?** If yes, adopt it. If not, build, choosing between Microsoft Foundry (PaaS), Copilot Studio (SaaS low-code), or GPUs and containers (IaaS). Before that, the business plan rules out agents altogether for structured, predictable tasks (use code or non-generative models) and for static knowledge retrieval (use classic RAG). In AB-100 language:

- **Buy** means adopt a ready-made capability: Microsoft 365 Copilot and its agents (for example Researcher), Dynamics 365 prebuilt agents, in-product Copilots, AI Builder prebuilt models.
- **Extend** means configure or augment something you already own: add knowledge, connectors, MCP tools, or actions; build declarative agents for Microsoft 365 Copilot; extend Dynamics 365 agents and Copilot experiences with Copilot Studio; reuse prompts.
- **Build** means own the component: custom agents in Copilot Studio or Foundry, custom engine agents, custom or fine-tuned models, self-hosted models.




### 2.7.2 Decision table

| Criterion | Buy (SaaS / prebuilt) | Extend | Build |
|---|---|---|---|
| **Time to value** | Fastest (days to weeks) | Fast | Weeks to months |
| **Differentiation** | Same as competitors | Your data and processes on a common base | Unique capability or process |
| **Fit to process** | Standard functions | Standard plus specific knowledge or actions | Non-standard, multistep, cross-system |
| **Control over models, orchestration, hosting** | Lowest | Moderate (choose knowledge, tools, sometimes models) | Highest (pick models, memory, networking) |
| **Operational responsibility** | Microsoft | Shared | Mostly yours |
| **Skills** | Admin and adoption | Makers, functional consultants | Developers, AI engineers, AgentOps |
| **Cost profile** | Licence-dominated | Licence plus consumption | Consumption, hosting, people |
| **Compliance and isolation** | Inherits product boundaries | Inherits plus your data policies | Full control (for example Foundry standard setup with private networking, or IaaS) |

CAF also recommends validating the choice with **time-boxed experiments of one to two weeks per option**, comparing low-code Copilot Studio with pro-code Foundry on development speed, functional coverage, and integration complexity, and proceeding with low-code if it meets functional and security requirements. Copilot Studio can also use Foundry models, a hybrid that gives business teams low-code delivery with access to advanced models.

**EXAM TIP**

"Buy" is not always cheaper in TCO, and "build" is not always more flexible in practice. The exam rewards the option that meets *all* stated requirements with the least build and run effort. If a prebuilt Dynamics 365 agent covers 80% and the remaining 20% is knowledge or a single action, the answer is usually **extend** with Copilot Studio, not rebuild in Foundry.

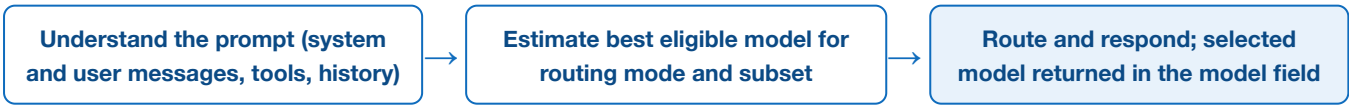
### 2.7.3 Cost implications of extending Microsoft 365 Copilot

| Aspect | Declarative agent (extend) | Custom engine agent (build) |
|---|---|---|
| Licence | Microsoft 365 Copilot add-on, or Copilot Chat access through an eligible Microsoft 365 licence | No additional licence required to access |
| Hosting | Hosted by Microsoft 365 Copilot; no extra hosting cost | Hosted externally with your orchestrator and models (for example Foundry, App Service, Bot Service) |
| Usage for unlicensed users | Free for instructions-only or public-website grounding; Copilot Credits for shared tenant data (SharePoint, Copilot connectors) | Copilot Credits may apply when shared tenant data is used |
| Usage for Microsoft 365 Copilot licensed users | No extra charge | No extra charge for the Copilot side; hosting still applies |

## 2.8 Implementing a model router

### 2.8.1 What model router is and how it routes

**Model router for Microsoft Foundry** is a purpose-built, trained routing model (not itself an LLM) that you deploy like any other Foundry model. For each request it analyses the prompt in real time and forwards it to the most suitable underlying LLM, so a single deployment delivers comparable quality at lower cost and latency. Smaller, cheaper models handle prompts they can answer well; frontier and reasoning models are used when the prompt needs them.




- It considers complexity, reasoning needs, task type, and other attributes; the routing overhead is a negligible fraction of inference time.
- It does not store prompts, routes only to models you are eligible for given access and deployment type, and honours data zone boundaries.

- Each request is routed independently by default, so turns of a stateless Chat Completions conversation can land on different models.
- The current version `2025-11-18` is actively updated in place with new underlying models; older versions (`2025-08-07`, `2025-05-19`) are frozen. With auto-update, the underlying model set can change, which can affect performance and cost.
- The routing pool spans OpenAI, Anthropic, xAI, DeepSeek, Meta, and other providers. Supported LLMs do not need separate deployments, **except Claude models**, which you must deploy from the catalogue first so the router can invoke them.
- It supports Global Standard and Data Zone Standard deployments, and agentic scenarios with tools in Foundry Agent Service.

### 2.8.2 Routing modes and model subsets

| Routing mode | Selection logic | Typical workloads |
|---|---|---|
| **Balanced (default)** | Considers models within a small quality band (for example 1% to 2%) of the best model for the prompt and picks the most cost-effective | General-purpose; start here |
| **Cost** | Considers a wider quality band (for example 5% to 6%) and picks the most cost-effective; most traffic goes to smaller models | High-volume, budget-sensitive: classification, simple Q&A, batch pipelines |
| **Quality** | Picks the highest-quality model for the prompt, ignoring cost | Critical outputs: legal review, medical summaries, complex reasoning |

A **model subset** restricts which underlying models the router may use. It is your control for cost, compliance, and capability: new base models are *not* added to a subset until you explicitly include them, and the subset doubles as the failover set so prompts are never processed by unapproved models. Built-in **automatic failover** redirects a request to the next most appropriate model when one has transient issues; it is on by default, and you need at least two models in a subset to benefit from it. Routing mode and subset changes can take up to five minutes to apply.

> **COMMON TRAP**
> The model router's effective context window is that of the **smallest** underlying model. A long prompt succeeds only if it happens to be routed to a model with a larger window. If prompts routinely exceed the smallest window, use a model subset containing only models that support the required context length (or summarise, truncate, or retrieve less context). Also note that image inputs are accepted but the routing decision is based on text only, and audio input is not processed.

### 2.8.3 Deploying and consuming model router

1. In the Foundry portal model catalogue, select `model-router` and choose **Default settings** (Balanced, all supported models) or **Custom settings**.
2. In custom settings, choose the routing mode and optionally **Route to a subset of models**.
3. Apply one content filter and one tokens-per-minute limit to the router deployment; these apply to all traffic through it, so do not configure them per underlying model and do not deploy the underlying chat models separately (Claude excepted).

4. If Azure Policy restricts model deployment, the built-in Foundry model deployment policy also governs which models can be in a router subset; allowed publishers must include Microsoft (the router's publisher) and each routed model's publisher.
5. Call the deployment through Foundry Responses or Chat Completions, or select it as the model of a Foundry agent; the agent keeps its tools and instructions while the router picks the model per request.

**Listing 2.2: Request body for a custom model router deployment (Azure management REST API PUT on the deployment resource)**

```
{
  "sku": { "name": "GlobalStandard", "capacity": 10 },
  "properties": {
    "model": { "format": "OpenAI", "name": "model-router", "version": "2025-11-18" },
    "routing": {
      "mode": "balanced",
      "models": [
        { "format": "OpenAI", "name": "gpt-4.1", "version": "2025-04-14" },
        { "format": "OpenAI", "name": "gpt-4.1-mini", "version": "2025-04-14" },
        { "format": "Meta", "name": "Llama-4-Maverick-17B-128E-Instruct-FP8", "version": "1" }
      ]
    }
  }
}
```

Omit the `routing` block to accept Balanced mode across the full supported set. If you include Claude models in `routing.models`, deploy them first to the same Foundry account with a matching SKU, or the request fails.

**Session affinity** PREVIEW lets an application pass an opaque session ID so that related Chat Completions turns try the same eligible model first. It can improve behavioural continuity and prompt-cache reuse, but it does not guarantee cache hits, does not override policy, safety, capability, quota, or failover, and the association expires after 30 minutes without a successful request. Prompt caching works automatically when the selected model supports it.

### 2.8.4 Cost and quality trade-offs, evaluation, and monitoring

The how-to guide describes model router cost as the sum of the costs incurred by the underlying models that serve requests, and the overview notes that router usage is charged for input prompts at the rate on the pricing page. To see where tokens and money go:

- **Performance:** Azure Monitor, Monitoring > Metrics on the resource, filter by the router deployment, split by underlying model.
- **Cost:** Cost analysis, filter by the Deployment tag with the router deployment name.
- **Per request:** log the `model` field in every response; it is the primary observability signal.

Microsoft frames optimisation as **hill climbing**: establish a baseline, measure the outcomes that matter, change one lever, and keep the change only if it moves the workload toward its quality, cost, latency, and policy requirements. To evaluate model router:

1. Define the decision (replace a direct deployment, choose a mode, or choose a subset) and workload acceptance criteria for **quality**, **cost**, **latency** (median and tail such as P90/P95), and **policy**.
2. Compare against a meaningful baseline using representative prompts grouped by category, with secrets and personal data removed; keep application configuration constant.

3. Use automated judging plus qualified human review for high-impact or regulated categories (Microsoft publishes a Model Router Auto Evaluation toolkit).
4. Read trade-offs together: lower cost does not justify a regression in an important category, and a good average latency can hide slow tails.
5. Validate under production-like traffic, then treat the chosen configuration as the new baseline and re-evaluate whenever traffic mix, mode, subset, supported models, or pricing changes.

| Observation | Next lever to try |
|---|---|
| Quality below requirement | Quality mode, or constrain the subset |
| Quality fine, savings limited | Cost mode; recheck high-impact categories |
| One category behaves differently | Evaluate that category separately; consider a direct deployment for it |
| Result depends on a disallowed or unavailable model | Choose a subset that is permitted and available in region |
| Tail latency too high | Test under production-like concurrency; consider Cost mode (smaller models respond faster) |

### 2.8.5 Router vs direct deployment vs custom routing

| Option | Choose when | Trade-off |
|---|---|---|
| **Model router (default path)** | Diverse prompt mix; cost matters; high-traffic user-facing chat; agents with mixed sub-tasks; you want one endpoint, one rate limit, failover, and transparent model lifecycle | Different requests can use different models; context limited by smallest model unless subset |
| **Direct deployment of one model** | Same model required on every request; parameter-sensitive or compliance-mandated workloads; fine-tuned model | You manage model selection, upgrades, and failover yourself |
| **Hybrid (recommended pattern)** | Router for general traffic; direct deployments for specialised paths | Two configurations to govern |
| **Hand-built routing logic in the app** | Rarely; only when routing depends on business rules the router cannot see | Ongoing engineering and evaluation burden that the router is designed to remove |

Before or alongside the router, the Foundry model leaderboards PREVIEW let you compare catalogue models on quality, safety, estimated cost, and throughput, which helps you choose a sensible subset or a direct model for specialised paths.

**EXAM TIP**

Best practices to remember: start in Balanced and observe the distribution for a few weeks; change one lever at a time; use the subset as your compliance gate; select at least two models; never use a single-model subset (it removes optimisation, savings, and failover and turns the router into an expensive passthrough); log the `model` field.

## 2.9 Key terms

| | |
|---|---|
| **Retrieval augmented generation (RAG)** | Adding retrieved organisational data to the prompt at run time so answers are grounded, current, and citable. |
| **Fine-tuning** | Additional training of a pretrained model on task-specific data that changes its weights; methods in Foundry include SFT, DPO, and RFT. |
| **Supervised fine-tuning (SFT)** | Training on prompts paired with desired responses. |
| **Direct preference optimisation (DPO)** | Training on preferred and rejected responses to the same input, for quality judged by preference such as tone. |
| **Reinforcement fine-tuning (RFT)** | Training with a grader that scores generated responses, where many valid solutions exist. |
| **Distillation** | Using outputs of a larger teacher model as training data for a smaller student model. |
| **Small language model (SLM)** | Generally a model with fewer than 10 billion parameters, such as Phi-4-mini (3.8 billion). |
| **Foundry Local** | On-device runtime and SDK for running optimised models locally, offline, without per-token cost or an Azure subscription. |
| **Organizational prompts** | Admin-published Copilot prompts (Microsoft 365 admin center) surfaced as suggestions, in the prompt lab, and in autosuggest. |
| **AI Center of Excellence** | Multidisciplinary team that sets AI strategy, standards, intake, reusable assets, skilling, and outcome reporting; evolves from centralised to advisory. |
| **Dual-write** | Near-real-time, bidirectional, tightly coupled sync between finance and operations apps and Dataverse. |
| **Virtual entities** | Dataverse representations of finance and operations data that do not copy the data. |
| **Total cost of ownership (TCO)** | All one-time and recurring costs to build, run, govern, and retire a solution over its life. |
| **Agent Assisted Hours / Value** | Microsoft's telemetry formula converting agent knowledge references and session outcomes into hours (default 6-minute multiplier) and value (default $72 per hour). |
| **Recapture rate** | The share of theoretical time saved that is actually redeployed to valuable work or avoided cost; protects against the time-savings trap. |
| **Copilot Credit** | Unit of Copilot Studio agent consumption; different features consume different numbers of credits. |
| **Model router** | Foundry deployment that selects an underlying LLM per request according to routing mode and model subset. |
| **Routing mode** | Balanced (default), Cost, or Quality; sets the cost-quality trade-off of model router. |
| **Model subset** | The approved list of underlying models model router may use, also used as the failover set. |

## 2.10 Additional practice questions

**Practice 2.1** MULTIPLE CHOICE

An insurer is building a Copilot Studio agent for brokers. Policy wording documents are revised every week, and brokers must see which clause supports each answer. A developer proposes fine-tuning a model on the current policy documents. You need to recommend the approach that meets the requirements with the least effort. What should you recommend?

A. Fine-tune a model weekly on the revised documents by using supervised fine-tuning.
B. Add the policy documents as a knowledge source and instruct the agent to cite clause references.
C. Train an AI Builder custom category classification model on the clauses.
D. Use reinforcement fine-tuning with a grader that checks clause numbers.

**Practice 2.2** MULTIPLE CHOICE

A distributor uses a frontier model to classify 2 million supplier emails per month into 15 categories. Accuracy is acceptable, but cost and latency are too high. The team has 8,000 emails labelled by the frontier model and verified by staff. You need to reduce cost and latency while keeping accuracy. What should you do first?

A. Move the workload to Quality mode in a model router deployment.
B. Fine-tune a smaller model on the verified examples (distillation) and compare it with the current model on a held-out test set.
C. Add all 8,000 examples to the system message as few-shot examples.
D. Deploy the frontier model with Provisioned Throughput.

**Practice 2.3** MULTIPLE CHOICE

A bank in France plans to fine-tune a model in Microsoft Foundry by using customer complaint transcripts. Regulations require that training data stays in the Foundry resource's region. The finance team asks for the cheapest training option. Which training type should you recommend?

A. Global
B. Developer
C. Data zone
D. Standard

**Practice 2.4** MULTIPLE SELECT (CHOOSE TWO)

The AI CoE has approved 40 prompts for Microsoft 365 Copilot users. Four of them must appear as suggested prompts when users open Copilot Chat, and all 40 must be searchable by department. Which two actions should you take?

A. Import the 40 prompts into organizational prompts in the Microsoft 365 admin center by using the CSV template, setting the Department field.
B. Share the prompts to a Teams team from the Prompt Gallery.
C. Pin the four priority prompts in the organizational prompts list.
D. Share the prompts as Co-owner from the Prompts page in Copilot Studio.
E. Store the prompts in a SharePoint list and add it as an agent knowledge source.

**Practice 2.5** MULTIPLE CHOICE

You are reviewing prompt engineering guidelines for a Foundry agent that will use a reasoning model. Which guideline should you remove?

A. Define the assistant's role, scope, and a fallback when information is missing.
B. Specify a small, stable JSON output contract.
C. Instruct the model to think step by step and print its full reasoning before the answer.
D. Ground answers in retrieved content and require inline citations.

**Practice 2.6** MULTIPLE CHOICE

A retailer with an established Cloud Center of Excellence is starting AI adoption. Different departments are experimenting with AI tools independently, and leadership wants consistent standards quickly. According to the Cloud Adoption Framework, what should you recommend?

A. Create a separate AI organisation reporting to the CTO with its own platform team.
B. Integrate AI expertise into the existing Cloud Center of Excellence and start with a centralised AI CoE operating model.
C. Start with an advisory model so departments keep full autonomy.
D. Delegate AI governance to each department's IT lead.

**Practice 2.7** MULTIPLE CHOICE

Service representatives use a Copilot Studio agent in Dynamics 365 Customer Service. The agent must answer questions about on-hand inventory held in Dynamics 365 Supply Chain Management. Inventory data must not be copied into Dataverse. What should you use as the agent's knowledge source?

A. Dual-write table maps for inventory
B. A nightly export of inventory to SharePoint
C. Virtual entities for finance and operations apps
D. An AI Builder prediction model

**Practice 2.8** MULTIPLE CHOICE

A proposed agent has one-time costs of $200,000 and recurring costs of $100,000 per year. Expected benefits are $300,000 in year 1, and $450,000 in each of years 2 and 3. What is the three-year ROI?

A. 140%
B. 200%
C. 240%
D. 67%

**Practice 2.9** MULTIPLE CHOICE

You are building the TCO for three Copilot Studio agents. Which usage must you budget for in Copilot Credits?

A. An HR agent used by employees who hold Microsoft 365 Copilot licences, running under their authenticated identity.
B. A customer-facing returns agent on the public website.
C. Test runs of an agent flow from the agent's test chat.
D. Power Automate cloud flows called by a canvas app.

**Practice 2.10** MULTIPLE CHOICE

A company uses Dynamics 365 Sales. The prebuilt Sales Qualification Agent meets most requirements, but sellers also need it to consider an internal pricing policy document and to create a follow-up task in a third-party system that has a REST API. You need to minimise build and run effort. What should you recommend?

A. Build a new qualification agent in Microsoft Foundry with a hosted agent.
B. Extend the solution with Copilot Studio: add the policy as knowledge and a connector or tool for the third-party API.
C. Host an open-source model on AKS and build custom orchestration.
D. Fine-tune a model on past qualified leads.

**Practice 2.11** MULTIPLE CHOICE

The security team has approved exactly three models for a customer support assistant. The assistant must keep working if one model has a transient outage, and newly released models must never be used without approval. You plan to use Foundry model router. What should you configure?

A. Default deployment with auto-update enabled.
B. Custom deployment in Quality mode with a model subset of one approved model.
C. Custom deployment with a model subset containing the three approved models.
D. Three direct deployments and a round-robin load balancer.

**Practice 2.12** MULTIPLE CHOICE

After moving a contract-analysis workload to a model router deployment in Balanced mode, some requests with very long contracts fail with context-length errors while others succeed. You need to stop the failures without changing the application's prompts. What should you do?

A. Switch the routing mode to Cost.

B. Enable session affinity.

C. Configure a model subset that includes only models whose context window supports the contracts.

D. Increase the tokens-per-minute limit on the deployment.

## Answers and explanations

**2.1: B.** Frequently changing knowledge with citation requirements is a grounding problem. Foundry guidance states that fine-tuning does not replace retrieval for current information; weekly retraining (A, D) adds cost and still cannot cite sources. C classifies text but does not answer questions.

**2.2: B.** Specialising a smaller model with teacher-generated examples is a documented fine-tuning use case that lowers cost and latency; evaluating against a held-out baseline is required because benefits are not guaranteed. A increases cost. C bloats every prompt. D addresses throughput predictability, not per-request cost or model size.

**2.3: D.** Standard training runs in the Foundry resource's region and guarantees data residency. Global and Developer are cheaper but copy data or provide no residency guarantee, and Data zone currently supports only the US data zone.

**2.4: A and C.** Organizational prompts support bulk CSV import with a Department field that powers filters in the prompt lab, and pinning (up to four) makes prompts appear as suggestions in Copilot Chat. B shares only with one team; D is for maker prompts in Power Platform; E does not create prompt suggestions.

**2.5: C.** Foundry guidance says classic prompt techniques such as chain-of-thought are not recommended for reasoning models, and extracting reasoning other than through the reasoning summary parameter is unsupported. A, B, and D are sound system-message and grounding practices.

**2.6: B.** The CAF recommends building on an existing CCoE rather than creating a standalone team unless current teams cannot support AI or critical risks exist, and recommends a centralised model early in the journey, moving to advisory later.

**2.7: C.** Virtual entities expose finance and operations data in Dataverse without copying it and can be added as agent knowledge. Dual-write (A) copies data bidirectionally. B copies data and is stale. D is unrelated.

**2.8: A.** TCO = $200,000 + 3 × $100,000 = $500,000. Benefits = $300,000 + $450,000 + $450,000 = $1,200,000. ROI = ($1,200,000 − $500,000) ÷ $500,000 = 140%.

**2.9: B.** Customer-facing usage is not covered by Microsoft 365 Copilot licences and consumes Copilot Credits. Employee use by Microsoft 365 Copilot licensed users under their identity is included (A), agent flow test runs do not consume capacity (C), and Power Automate cloud flows use Power Automate licensing, not Copilot Credits (D).

**2.10: B.** When a prebuilt agent covers most needs, extending with knowledge and a connector or tool meets the gap with the least effort. A and C rebuild what already exists and add operational responsibility; D does not add the policy document or the action.

**2.11: C.** A model subset is the compliance gate; new models are excluded until explicitly added, and with more than one model in the subset the built-in failover stays within approved models. A could introduce unapproved models. B is a single-model subset, which removes failover. D rebuilds routing and failover by hand.

**2.12: C.** The router's effective context window is that of the smallest model in the pool; restricting the subset to models with sufficient context resolves the failures. Cost mode (A) favours smaller models and may worsen the problem. Session affinity (B) does not check capability. TPM (D) is a rate limit, not a context limit.

## Key takeaways

- Climb the customisation ladder only with evidence: prompt engineering, then grounding for knowledge, then fine-tuning for behaviour and efficiency, then custom or self-hosted models when control demands it.
- Fine-tuning needs a baseline, quality data, a budget for training and hosting, and a retirement plan; it never replaces retrieval or application-level safety.
- SLMs (generally under 10 billion parameters) suit narrow, high-volume, precise, offline, or regulated tasks; distillation and Foundry Local are the key patterns.
- Give each prompt audience its own guidance: goal, context, source, expectations for users; grounded, short, tool-aware instructions for makers; system message checklists and model-specific rules for developers.
- Host prompt libraries where the audience works: organizational prompts for Copilot users, shared prompts in Dataverse for makers; govern with owners, review dates, analytics, and ALM.
- An AI CoE needs a sponsor, a leader, a multidisciplinary team, defined responsibilities, and an operating model that moves from centralised to advisory; integrate with an existing CCoE and Power Platform governance.
- Cross-app Dynamics 365 designs rely on Dataverse, dual-write for shared masters, virtual entities for read access without copies, ERP connectors or MCP for actions, and agent feed for supervision.
- ROI criteria map to four value drivers (efficiency, quality, revenue, strategic); build on a 90-day telemetry baseline, apply recapture and attribution, and include every TCO category, especially Copilot Credits for customer-facing and autonomous agents.
- Buy, then extend, then build: the least operational responsibility that meets all requirements wins.
- Model router selects a model per request; tune with routing mode and model subset, keep at least two models, watch the smallest context window, log the model field, and evaluate against a baseline one lever at a time.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Exam AB-100: Agentic AI Business Solutions Architect
- Getting started with customizing a large language model (LLM)
- Fine-tuning in Microsoft Foundry
- Concepts: Small and large language models
- What is Foundry Local?
- Model leaderboards in Microsoft Foundry portal (preview)
- AI Builder: AI models and business scenarios
- Prompt engineering techniques
- System message design
- Write an effective prompt (training unit)
- Discover, share, and save prompts with Prompt Lab (training unit)
- Organizational prompts overview
- Understand Prompt Gallery in Copilot

- Copilot Studio: Overview of prompts
- Copilot Studio: Share your prompt
- Copilot Studio: Bring your own model for your prompts
- Configure high-quality instructions for generative orchestration
- CAF: AI adoption
- CAF: Plan for AI adoption
- CAF: Guidance to set up your organization's AI management process
- CAF: Establish an AI Center of Excellence
- AI Center of Excellence (learning path)
- Determining organizational roles and responsibilities (training unit)
- Power Platform Center of Excellence (CoE) Starter Kit overview
- Copilot Agent Kit overview
- Agents, Copilot, and AI capabilities in Dynamics 365 apps
- Unified customer engagement solution with Dynamics 365 (reference architecture)
- Dual-write overview
- Dual-write FAQ
- Chat with finance and operations data (preview)
- Measure the ROI and business value of AI agents
- Define value before you build your agent
- Measure the impact of your agents
- Monitor time and cost savings for agents
- Copilot Studio billing rates and management
- Plan licensing and cost (Microsoft 365 Copilot extensibility)
- CAF: Plan your Azure environment for cost estimation
- CAF: Business plan for AI agents
- CAF: Technology plan for AI agents
- Model router for Microsoft Foundry
- How model router works in Microsoft Foundry
- Use model router for Microsoft Foundry
- Evaluate model router for your workload

DOMAIN 2 · PART A

# Design AI and agents for business solutions

**Part of Domain 2: Design AI-powered business solutions (25–30%)**

This chapter covers the first and largest sub-skill group of Domain 2. It is where the exam moves from strategy into concrete design decisions: which agent type fits a process, how Copilot in Dynamics 365 learns your vocabulary, how a Copilot Studio agent reaches customers through Dynamics 365 Contact Center, which language understanding option to pick, when a topic beats knowledge or a tool, how grounding data is processed, and which Foundry Tool answers a specific requirement.

Expect scenario questions that describe a business requirement plus one or two constraints (licensing, determinism, latency, security trimming, data residency, maker skills) and ask you to choose the design that satisfies all of them. The fastest way to study is to learn the *decision boundaries* in each decision table, then read the "Common trap" callouts, because most wrong options are real features used in the wrong place.

| Skills outline bullet (Design AI and agents for business solutions) | Section |
|---|---|
| Design business terms for Copilot in Dynamics 365 apps for customer experience and service | 3.1.1 |
| Design customizations of Copilot in Dynamics 365 apps for customer experience and service | 3.1.2 |
| Design connectors for Copilot in Dynamics 365 Sales | 3.1.3 |
| Design agents for integration with Dynamics 365 Contact Center channels | 3.1.4 |
| Design task agents | 3.2.3 |
| Design autonomous agents | 3.2.4 |
| Design prompt and response agents | 3.2.2 |
| Propose Microsoft Foundry Tools for a given requirement | 3.4.2 |
| Propose code-first generative pages and the use of an agent feed for apps | 3.4.3 |
| Design topics for Copilot Studio, including fallback | 3.3.2 |
| Design data processing for AI models and grounding | 3.4.1 |
| Design a business process to include AI components in a Power Apps canvas app | 3.4.4 |
| Apply the Microsoft Power Platform Well-Architected Framework to intelligent application workloads | 3.4.5 |
| Determine when to use standard natural language processing, conversational language understanding, or generative AI orchestration in Copilot Studio | 3.3.1 |
| Design agents and agent flows with Copilot Studio | 3.3.3 |
| Design prompt actions in Copilot Studio | 3.3.4 |

## 3.1 Copilot in Dynamics 365 customer experience and service apps

Copilot in Dynamics 365 Sales and Copilot in Dynamics 365 Customer Service are *prebuilt* experiences. As an architect you rarely rebuild them; you shape them. The levers are vocabulary (glossary and synonyms), instructions and prompts, knowledge sources, filters, and extensibility through a managed Copilot Studio agent that sits behind each experience. Knowing *where* each lever lives (Copilot Studio, Copilot Service admin center, or Power Platform admin center) is a recurring exam theme.

### 3.1.1 Design business terms for Copilot in Dynamics 365 apps

Copilot answers questions about CRM data by interpreting table and column metadata. Real organisations rarely talk in schema names: sellers say "deal value" rather than `cr123_customrevenue`, and "overdue task" is a business rule, not a column. Microsoft Learn describes two complementary mechanisms that close this gap.

| | |
|---|---|
| **Synonym** | An alternative name for a specific column in a specific table, for example "opportunity revenue" for the **Custom Revenue** column of the **Opportunity** table. Synonyms help Copilot recognise different ways users refer to the same field. |
| **Glossary term** | An organisation-specific term with a natural-language description that maps it to tables, columns, values, or rules. Microsoft's examples include acronyms ("VP" refers to the Vice President value in **JobTitle** on **Contact**), custom ownership ("activity owner" is **PartyId** on **ActivityParty**), and complex filters ("overdue task" is an open task whose scheduled end date is earlier than today). |

**Where business terms live for Sales.** Glossary and synonyms for Copilot in Dynamics 365 Sales are authored in Copilot Studio: open the Dynamics 365 Sales environment, select **Agents** > **Copilot in Dynamics 365 Sales** (it might still appear as **Sales Copilot Power Virtual Agents Bot**), open the **SalesSpecificQnA** knowledge, then use the **Glossary** and **Synonyms** sections and publish. The author needs the **Copilot Studio Author** role. The same glossary and synonyms are shared with the Sales agent in Microsoft 365 Copilot, so one set of definitions serves both surfaces. Glossary updates can take up to 15 minutes to take effect.

**Business terms for Dataverse knowledge in custom agents.** When a custom Copilot Studio agent uses Dataverse tables as knowledge, the same pattern applies at the knowledge-source level: you can add up to 15 tables per Dataverse knowledge source, then add column synonyms and glossary terms. Microsoft notes that numeric or cryptically named columns need an explicit description so the model can interpret them.

**Business terms for Customer Service.** Copilot in Customer Service grounds mostly on knowledge articles rather than ad-hoc CRM queries, so vocabulary is shaped differently. In the Copilot Service admin center, the **Customer Support** agent's **Overview** tab accepts custom **Instructions** (up to 5,000 characters) that can include tone, escalation guidance, compliance reminders, and organisation-specific terminology. On the **Prompts** tab, each organisational prompt has an **Alias** that lets Copilot recognise different phrasings for the same prompt behaviour. Custom instructions apply only when the Dynamics 365 knowledge base is the selected knowledge source.

| Requirement | Design choice | Where it is configured |
|---|---|---|
| **Sellers use a different word for an existing column** | Synonym on that column | Copilot Studio, Copilot in Dynamics 365 Sales, SalesSpecificQnA |
| **A term implies a filter or rule across columns** | Glossary term with a precise description | Same location (Sales), or Dataverse knowledge source glossary (custom agents) |

| Requirement | Design choice | Where it is configured |
|---|---|---|
| **Same vocabulary must work in Outlook and Teams for sellers** | Author once in Copilot Studio; shared by Sales agent in Microsoft 365 Copilot | Copilot Studio (Dynamics 365 CRM). Salesforce glossary is configured separately; synonyms aren't supported for Salesforce |
| **Service representatives' answers must use house terminology and escalation wording** | Customer Support agent instructions | Copilot Service admin center, Agents within Copilot, Customer Support, Overview |
| **Different phrasings should run the same organisational prompt** | Prompt alias | Customer Support agent, Prompts tab |

**EXAM TIP**

If an option says "rename the column display name" or "create a topic with trigger phrases for each business term", it is almost always a distractor. The documented, low-maintenance design is synonyms plus glossary, and for Sales they are authored once in Copilot Studio and reused by the Sales agent in Microsoft 365 Copilot.

**PRODUCTION NOTE**

Treat glossary descriptions like code: version them, test them with real seller phrasing, and expect a propagation delay of up to 15 minutes. If you are unsure how to word an entry, Microsoft suggests asking the Sales agent itself to draft a glossary entry from a description of your custom entity or relationship, then reviewing it before use.

### 3.1.2 Design customisations of Copilot in Dynamics 365 customer experience and service apps

Customisation splits along product lines, and the exam expects you to know the boundaries.

**Copilot in Dynamics 365 Sales.** Documented customisations are: add or edit topics in the **Copilot in Dynamics 365 Sales** agent in Copilot Studio (to bring in organisation data, knowledge sources, and actions), customise the prompt guide by editing the **SalesSparks** topic, and add glossary terms. Customising the Sales Copilot through Copilot Studio is marked PREVIEW, and Microsoft recommends managing these customisations in a solution so they can move across environments. Custom topics consume Copilot Studio credits, and image and video message types aren't supported from agent to user (Adaptive Cards are the workaround).

**Copilot chat in other model-driven apps.** For standalone model-driven apps you use **Configure in Copilot Studio** from the app designer; every such app gets its own dedicated agent where you add knowledge sources or topics. This is generally available in Dynamics 365 apps but PREVIEW in Power Apps, and Microsoft recommends Microsoft Copilot in model-driven apps for environments not enabled for Dynamics 365 apps because Copilot chat is being deprecated there from January 2026.

**Copilot in Customer Service.** Configuration is spread over two admin centres:

- **Power Platform admin center**: turn on AI-powered Copilot features for the environment, allow data movement across regions where Copilot isn't available in region (also used to relieve capacity throttling), and decide on data sharing.

- **Copilot Service admin center**: opt in, choose **Copilot immersive**, **Copilot side pane**, or both, use experience profiles to control which representatives get which features, and configure the **Customer Support** agent (Overview, Immersive settings, Email settings, Prompts, Extend agent tabs).

Key Customer Support agent settings that appear in design questions:

- **Knowledge filters** for **Ask a question** and **Draft a response**: preset filters (hidden, always applied), representative filters (visible, selectable), and automated filters (applied from the current record's field values, optionally visible). Up to five filters per category.
- **Deterministic response**: returns consistent, verbatim responses for the same input, useful for validation and testing scenarios.
- **Knowledge sources**: the organisation's knowledge base (only the content attribute of published, latest-version articles in the user's UI language is used), external trusted webpages for email drafting (up to five domains; note the documented Bing Search API retirement caveat), and Copilot Studio knowledge sources for Ask a question.
- **Prompts tab**: organisational prompts with Prompt Text, In Guide, Alias, Filter, Entity, Scope, and optional instructions for Copilot (up to 10,000 characters).

**Extending Copilot in Customer Service with Copilot Studio** PREVIEW. Select **Extend agent with Copilot Studio** on the Extend agent tab. This links Ask a question to a managed agent called **Customer Service Copilot Bot**, where you add knowledge sources, tools (connectors, custom connectors, MCP servers), other agents, and topics. You can also define **custom context** (entity, data field, description) that Copilot passes to the Copilot Studio agent so a tool can use, for example, the customer's name without the representative typing it. Default context already includes the current record, user role information, and session metadata. Limitations: you can't change the agent's model (connected agents can use a different model), topics whose names start with `msdyn_` can't be modified, and the feature isn't available in GCC-M or GCC-H.




**COMMON TRAP**

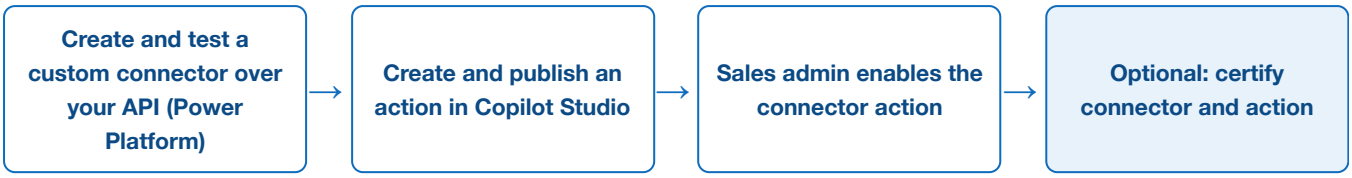
Custom context is configured in the **Copilot Service admin center** (Extend agent tab), not inside the Copilot Studio tool. Microsoft explicitly says that if you want to pass Dynamics context to a tool, do it in the admin center. Also, if you need to change anything beyond a plug-in's connection or tool description, create a new tool and remove the old one.

### 3.1.3 Design connectors for Copilot in Dynamics 365 Sales

"Connectors" in a Sales context covers three different design patterns. Pick the one that matches the direction of data flow and the surface the seller uses.

**Pattern 1: Copilot Studio extensions of Copilot in Dynamics 365 Sales.** Because Copilot in Dynamics 365 Sales is backed by a Copilot Studio agent, you can add topics that call connector actions or agent flows and add knowledge sources. This keeps the experience inside the model-driven Sales app and is the right choice when the requirement is "sellers working in Dynamics 365 Sales ask Copilot about data held in system X".

**Pattern 2: Connector actions that extend the Sales agent in Microsoft 365 Copilot** PREVIEW. For sellers working in Outlook and Teams, Microsoft documents an extension path for partner or in-house applications:




The extension points are specific Sales capabilities: enriching email summaries, email drafts (file links), opportunity insights, CRM record details, and CRM record summaries. After an administrator enables an action it can take up to seven days to be ingested (users can sign out and in to speed it up). Microsoft warns that enabling a connector action for Sales may let users in Microsoft 365 and Teams send and receive data from external sources even if the same connector action is disallowed for direct use with Microsoft 365, and that third-party terms might apply. That is a governance review point, not a footnote.

**Pattern 3: Knowledge connectors.** When the requirement is read-only grounding rather than an action, use knowledge:

- **Copilot connectors** (formerly Microsoft Graph connectors) index external content into Microsoft 365 so Copilot and custom agents can reason over it with security trimming. Copilot Chat is read-only against external systems unless extended with action connectors or plugins.
- **Real-time connectors** in Copilot Studio retrieve live structured data from systems such as Salesforce, ServiceNow, Zendesk, and Azure SQL. The signed-in user must create connections to the target systems.
- **SharePoint** content recommendations are a built-in Copilot in Dynamics 365 Sales capability (information assistance), so don't build a connector for that requirement.

| Requirement | Pattern | Why |
|---|---|---|
| **Sellers in the Sales Hub ask about invoices held in an ERP API** | Topic or tool in the Copilot in Dynamics 365 Sales agent using a connector or agent flow | Keeps the experience in the Dynamics 365 Sales Copilot pane |
| **Opportunity insights in Outlook should include data from a partner app** | Custom connector plus Copilot Studio action enabled for the Sales agent | Documented extension point for opportunity insights |
| **Answer questions from a third-party knowledge base, with permissions honoured** | Copilot connector or real-time connector knowledge | Read-only grounding with user-level access |
| **Write back to an external system from chat** | Action (connector, agent flow, MCP tool) | Knowledge connectors are read-only |

> **NOTE**
> Customisations other than glossary and synonyms don't automatically carry over to the Sales agent in Microsoft 365 Copilot. Summary field configuration must be redone in the Sales agent settings, and custom topics and skills (Q&A and actions) must be rebuilt as declarative agents or custom tools. Welcome card and prompt suggestion customisation isn't supported there.

### 3.1.4 Design agents for integration with Dynamics 365 Contact Center channels

Dynamics 365 Contact Center is a Copilot-first contact centre that can run embedded in Dynamics 365 Customer Service or standalone alongside another CRM. Self-service in every channel is delivered by Copilot Studio agents (Microsoft now calls them Copilot agents or AI agents; human agents are customer service representatives).

**Channels.** Live chat, SMS, voice (PSTN with live transcription, sentiment, and in-call consults), Microsoft Teams for internal support, social channels (for example Facebook, LINE, and WhatsApp), and custom messaging channels through Direct Line. Licensing differs: telephony is available with Customer Service Enterprise and the Chat and Digital Messaging add-ons; web chat needs the Chat or Digital Messaging add-on; social and custom channels need the Digital Messaging add-on. Agent usage across voice and digital channels consumes Copilot credits.

**Integration design.** One agent can serve all channels without channel-specific code. The connection steps matter on the exam:

1. Create the agent in Copilot Studio (or start it from the Copilot Service admin center). You need the CCI Admin and Omnichannel administrator roles.
2. In Copilot Studio, **Channels** > **Dynamics 365 Customer Service** > **Enable**. The agent is modelled as an *application user*, so its Microsoft Entra application ID must be unique per agent (reuse causes error 1004, DuplicateBotAppId). Agent and Customer Service must be in the same environment for analytics.
3. Attach the agent to a workstream and queue in the Copilot Service admin center.
4. Configure escalation: the **Escalate** system topic hands off to a representative with full conversation history and variables. Use the representative availability APIs to check queue availability before escalating.
5. For voice: **Settings** > **Voice** > **Enable voice**, choose **Basic voice** or **Real-time voice**, add IVR features (DTMF, speech grammars, call transfer, termination), then configure the voice channel.

**Real-time voice agents** support barge-in, low latency, automatic multilingual switching, DTMF fallback, and full context on human escalation. They don't support every topic trigger: **On Unknown Intent** (the Fallback trigger), multiple-topics-matched disambiguation, **A message is received**, and **A plan completes** are listed as not supported, while On Error, Silence detection, Unknown DTMF key press, and redirects are supported.

**Orchestration choice for contact centre agents.** Classic orchestration matches utterances to topics by trigger phrases. Generative orchestration chooses tools, knowledge, and topics dynamically. NLU+ is available when voice or chat channels are managed with a Dynamics 365 Contact Center licence; with NLU+ the runtime processing happens in Dynamics 365 Contact Center rather than in Copilot Studio, and its training data also tunes speech recognition.

> **COMMON TRAP**
> When testing on your own website, use the **embed code from the chat widget** configured in Dynamics 365. If you use the embed code from Copilot Studio, handoff to a representative doesn't work. A second trap: if an agent connected to a voice workstream has voice disabled in Copilot Studio, calls start to fail, so disconnect it from workstreams first.

> **EXAM TIP**
> Dynamics 365 Contact Center also ships prebuilt agents (for example Customer Assist Agent for high-volume voice and digital requests, and Customer Intent Agent). If the scenario asks for the least custom build, consider whether a prebuilt agent plus Copilot Studio extension already meets it.

## 3.2 Agent types: prompt and response, task, and autonomous

Microsoft describes agents as ranging "from simple prompt-and-response agents to more advanced, fully autonomous agents." The Cloud Adoption Framework (CAF) frames the same spectrum as three categories with rising complexity and governance needs: retrieval (productivity) agents, task (action) agents, and autonomous (automation) agents.

### 3.2.1 Selecting the agent type

| Dimension | Prompt and response (retrieval) | Task (action) | Autonomous (automation) |
|---|---|---|---|
| **Starts when** | A user asks | A user asks for something to be done | An event, schedule, or another agent triggers it, with no user prompt |
| **Core building blocks** | Instructions plus knowledge | Knowledge plus tools (connectors, agent flows, MCP, prompts) | Knowledge, tools, and triggers that decide when to run, stop, or escalate |
| **Typical example** | Policy or product Q&A, case knowledge lookup | Create a service ticket, update a CRM record, book a visit | Triage inbound email, process invoices, react to Dataverse row changes |
| **Key risk** | Ungrounded or stale answers | Wrong action or wrong parameters | Unattended actions at scale, credential misuse, cost runaway |
| **Primary controls** | Knowledge scoping, citations, moderation | Confirmation, input validation, least-privilege connections | Scoped permissions, human-in-the-loop, audit, throughput planning |
| **Copilot Studio requirement** | Generative or classic orchestration | Generative orchestration recommended | Event triggers require generative orchestration |

### 3.2.2 Design prompt and response agents

A prompt and response agent answers questions from trusted content. Its quality depends almost entirely on knowledge design, instructions, and the boundaries you set.

- **Platform choice.** If the audience lives in Microsoft 365, a declarative agent that extends Microsoft 365 Copilot (built in Copilot Studio or Microsoft 365 Agents Toolkit) is often enough. Copilot Studio now describes a **Copilot chat harness** for extending Microsoft Copilot Chat with enterprise knowledge, published to internal teams. For customers on a website or in a contact centre, use a Copilot Studio custom agent.
- **Knowledge.** Add sources at agent level (they are automatically added to the **Conversational boosting** system topic) or at topic level in a generative answers node. Give every source a unique name and a detailed description, because generative orchestration uses descriptions to choose sources.
- **General knowledge.** Decide whether the agent may use the model's general knowledge. Child and connected agents respect the main agent's setting.
- **Boundaries.** Microsoft's RAG guidance states that retrieval-augmented answers work best for factual Q&A and summarising policies, not for full document comparison, compliance evaluation, or complex

reasoning over long documents. Route those to prompts with code interpreter, deep reasoning models, or a Foundry solution.

**EXAM TIP**

"Answers must only come from approved content and always show a source" points to a retrieval agent with generative answers (citations are generated), general knowledge turned off, and content moderation set appropriately. Adding tools to such an agent changes its risk profile and turns it into a task agent.

### 3.2.3 Design task agents

A task agent performs specific actions within a defined workflow, combining knowledge tools with action tools. In Copilot Studio, tools include prebuilt and custom connectors, agent flows, prompts, MCP servers, REST APIs, computer use, and other agents.

- **Describe tools precisely.** Under generative orchestration the planner selects tools by name and description, generates questions for missing inputs, and composes one answer from several tool outputs.
- **Deterministic where it matters.** The agent design framework recommends topics or flows for multistep data collection, guided troubleshooting, compliance-driven processes, and high-impact or irreversible actions. Use orchestration for breadth and topics or agent flows for precision.
- **Authentication.** Decide between end-user credentials (actions run as the user; supports security trimming) and maker-provided credentials (simpler UX, but every user acts with the maker's rights). Customer Service extensibility recommends maker authentication for user experience when possible, which is a deliberate trade-off you must govern.
- **Confirmation and error handling.** Ask for confirmation before writes, design an error-handling pattern for flows, and remember a flow's response to the agent has a size limit.
- **Scale of choice.** As a rule of thumb, selection precision can degrade when a main agent has more than 30 to 40 choices (tools, topics, other agents), or fewer with similar descriptions. Split into child agents or connected agents (see 3.3.3).

### 3.2.4 Design autonomous agents

Autonomous agents act without waiting for a user prompt. They perceive events through **event triggers**, decide using instructions and guardrails, and execute through tools. Examples of triggers: an item is created in SharePoint, a file is created in OneDrive, a Planner task is completed, a Dataverse row changes, a new email arrives, or a **Recurrence** schedule. Finance and operations business events can also trigger Copilot Studio agents.




**Design rules from Microsoft Learn:**

- Event triggers are available only for agents with generative orchestration turned on.
- Event triggers authenticate only with the agent author's credentials, and all triggers and actions that need authentication must use maker credentials for the agent to run autonomously. If you publish an agent with authenticated triggers, users might access information or prompt actions using the author's credentials. Copilot Studio warns makers at publish time.

- Trigger payloads can contain sensitive data that actions may write elsewhere. Narrow trigger parameters, instruct the agent how to handle payload data, and test what reaches outputs.
- Administrators can block event triggers in an environment with data policies, and available triggers depend on those policies.
- Frequent triggers (especially short recurrences) consume capacity and can hit quotas and throttling. Triggers affect billing.
- Best practices: narrow scope and explicit authority, quality data and instructions, gradual rollout, human approval for sensitive actions, least privilege, input validation (for example sender checks on email triggers), audit logging, and alerting on unusual behaviour.

**Monitoring.** The Copilot Studio **Monitor** page for agents with event triggers focuses on run outcomes, trigger use, tool use, and knowledge source use, with up to 360 days of history and a Savings area for time and cost savings.

**Harness note.** Copilot Studio now runs everything on a *harness*: the **standard harness** for rule-based agents and agent flows, the **GitHub Copilot harness** for reasoning-heavy, multistep work (with files, skills, and memory, billed in Copilot Credits), and the **Copilot chat harness** for extending Microsoft Copilot Chat. Triggers for agents on the GitHub Copilot harness are documented as PREVIEW. Most exam-relevant topic, NLU, and agent flow guidance in this chapter applies to the standard harness.

> **COMMON TRAP**
> Don't design an autonomous agent that runs as "the signed-in user". There is no signed-in user when a trigger fires. Plan a dedicated maker or service identity with least privilege, and consider whether the agent should also be published to interactive channels at all.

## 3.3 Copilot Studio conversation and automation design

### 3.3.1 Determine when to use standard NLU, CLU, or generative orchestration

The skills outline uses the phrase "standard natural language processing". In Copilot Studio terms that means the built-in **classic** language understanding options. The setting lives in **Settings** > **Generative AI** > **Orchestration** (Yes for generative, No for classic), then **Language understanding** for the classic model.

| | |
|---|---|
| **Generative orchestration** | The default. A language model plans which topics, tools, knowledge, and agents to use, recognises multiple intents in one utterance, generates questions for missing inputs, allows corrections mid-run, and composes a unified answer. Relies on high-quality names and descriptions. Limits: five messages per topic or action chain and 128 topics and actions considered for triggering. |
| **Built-in NLU** | Classic, deterministic routing by trigger phrases (about 5 to 20 short phrases per topic), prebuilt entities, and custom closed list or regex entities. Single intent per query; can't be extended. Now described as the fallback approach rather than the default. |
| **NLU+** | Classic, higher-accuracy model for large enterprise agents with many topics, entities, and training samples. Supports entity annotations, is precompiled and must be trained before production, and its training data improves speech recognition for voice agents. Requires a Dynamics 365 Contact Center licence for the voice or chat channels; runtime processing occurs in Dynamics 365 Contact Center. |

| | |
|---|---|
| **Azure Conversational Language Understanding (CLU)** | Bring your own Azure AI Language (Azure Language in Foundry Tools) CLU model. Adds native support for more languages, a customisable intent model for industry vocabularies, and advanced entity extraction (for example multiple entities of the same type). Requires an Azure subscription, extra cost, Azure service limits, and keeping CLU intents synchronised with topics. |

| Scenario signal | Choose | Reasoning |
|---|---|---|
| **Broad questions spanning several business areas; minimal authoring; multi-intent utterances** | Generative orchestration | Plans across topics, tools, and knowledge; generates follow-up questions automatically |
| **Need event triggers (autonomous agent)** | Generative orchestration | Event triggers are only available with generative orchestration |
| **Small, predictable FAQ-style agent; full repeatable control; simple entities** | Classic with built-in NLU | Deterministic; easy to program; low setup |
| **Large contact centre agent, hundreds of topics, heavy entity use, voice accuracy matters, Contact Center licensed** | Classic with NLU+ | Enterprise-scale accuracy; precompiled model; improves speech recognition |
| **Existing Azure CLU investment, non-English or industry-specific language, same-type multi-entity extraction** | Classic with Azure CLU | Custom model in Azure; native multilingual models |
| **More than 128 topics and actions must be routable** | Split into child or connected agents, or classic routing | Generative orchestration considers up to 128 for triggering |

**What happens when you switch to CLU.** You must first set orchestration to classic. Choosing **Utilize prebuilt Azure NLU** prompts you to save a snapshot (`botContent.zip` containing `botContent.yml`) and then **deletes existing trigger phrases**. After you enter the Azure AI Language project and deployment, a new system topic named **Analyze Text** appears to recognise intents and entities. Switching back to Copilot Studio NLU removes Analyze Text and you must re-add example phrases for topics mapped to external intents.

**Hybrid orchestration.** A documented case study (Holland America Line) kept classic orchestration but removed trigger phrases from custom topics, then modified the **Conversational boosting** topic to call an AI prompt that classifies intent from the current question and conversation context, and redirected with a condition to the matching topic, falling back to generative answers and human escalation. This is a pattern to recognise when a scenario needs precise intent control plus LLM-quality understanding.

**COMMON TRAP**

"Use generative orchestration and NLU+ together" can be correct only in the voice channel context described in Contact Center docs (selecting an NLU model under generative orchestration). In Copilot Studio settings, NLU+ is configured as a classic option. Read the scenario carefully: if it stresses full repeatable control and trigger-phrase routing, it is classic.

### 3.3.2 Design topics for Copilot Studio, including fallback

Under generative orchestration, topics have shifted from rigid intent paths to modular, deterministic building blocks that the planner can call. Topics are still the primary mechanism for deterministic logic. Design them as either **entry point topics** (triggered by the user's utterance) or **reusable subtopics** (triggered by redirect, with input and output variables), and use events to trigger topics where appropriate.

**Trigger types. The agent chooses** (generative orchestration; matches the topic's name and description), **User says a phrase** (classic), **A message is received**, **A custom client event occurs**, **An activity occurs**, **The conversation changes**, **It's invoked**, **It's redirected to**, and inactivity. Execution order when several fire: activity first, then message, event, conversation update, or invoke, then agent-chooses or phrase triggers. A **Priority** property and Power Fx **conditions** refine this. The inactivity duration is in seconds.

**System topics** are created automatically and can be turned off but not deleted:

| System topic | Fires when | Design use |
|---|---|---|
| Conversation Start | Agent first engages (in Teams only once per user) | Greeting, capability disclosure |
| Conversational boosting | No topic match; before Fallback | Generative answers over agent-level knowledge |
| Fallback | Agent can't match the message to a topic with enough confidence | Clarify, offer options, call external Q&A, escalate after repeated misses |
| Multiple Topics Matched | Message closely matches several topics | "Did you mean" disambiguation |
| Escalate | "Talk to agent" or Escalate system event | Handoff to Dynamics 365 Contact Center or other engagement hub |
| End of Conversation | Redirect from a custom topic | Confirm resolution, survey |
| On Error | A user error occurs (not system errors) | Show error code, conversation ID, and timestamp |

**Fallback design.** Conversational boosting triggers before Fallback, so unmatched queries first try knowledge. Fallback then becomes the safety net. Documented patterns:

- **Offload single-turn Q&A.** Copilot Studio limits an agent to 1,000 topics. Large volumes of question and answer pairs can live in an external system (for example question answering in Azure Language) reached from an action node in Fallback through an agent flow.
- **Recover gracefully.** Power Platform Well-Architected XO:10 recommends clarifying questions instead of "I didn't understand", alternative suggestions that explain capabilities, and escalation to a human or help content after repeated misunderstandings.
- **Topic enrichment.** Analyse utterances that hit Fallback or Conversational boosting offline. Missed existing intents get more trigger phrases; clusters of new intents become new topics.

| Need | Topic | Knowledge | Tool (action) |
|---|---|---|---|
| **Exact wording, regulated script, or fixed sequence** | Yes | No | Optional inside the topic |
| **Answer factual questions from documents or sites** | Only to wrap a generative answers node | Yes | No |

| Need | Topic | Knowledge | Tool (action) |
|---|---|---|---|
| **Read or write live system data** | To control collection and confirmation | Real-time connector for read-only lookups | Yes |
| **Slot filling of several entities with validation** | Yes (question nodes, entities) | No | After validation |
| **Open-ended, multi-part request** | As callable building blocks | Yes | Yes, chosen by orchestrator |

Topics can be edited in the code editor as YAML. The following abbreviated example from Microsoft Learn shows a classic entry point topic with trigger phrases, a question bound to a prebuilt entity, and a condition expressed in Power Fx.

**Listing 3.1: Topic YAML with trigger phrases, prebuilt entity, and condition (abbreviated from Microsoft Learn)**

```
kind: AdaptiveDialog
beginDialog:
  kind: OnRecognizedIntent
  id: main
  intent:
    triggerQueries:
      - Buy items
      - Order product
  actions:
    - kind: Question
      id: eRH3BJ
      variable: init:Topic.State
      prompt: To what state will you be shipping?
      entity: StatePrebuiltEntity
    - kind: ConditionGroup
      id: sEzulE
      conditions:
        - id: pbR5LO
          condition: =Topic.State = "California" || Topic.State = "Washington"
      elseActions:
        - kind: SendMessage
          id: X7BFUC
          message: There will be an additional shipping charge.
```

**EXAM TIP**

Real-time voice agents don't support the On Unknown Intent (Fallback) trigger or Multiple Topics Matched. For voice, design recovery through On Error, silence detection, unknown DTMF key press handling, and escalation instead.

### 3.3.3 Design agents and agent flows with Copilot Studio

**Agent anatomy.** A Copilot Studio agent combines instructions, knowledge, tools, topics, triggers, and optionally other agents, and publishes to channels such as Teams, Microsoft 365 Copilot, websites, Dynamics 365 Contact Center, and messaging platforms.

**Multi-agent design.** Add other agents as **child agents** (lightweight, inside the parent, shared settings and ALM), **connected Copilot Studio agents** in the same environment, or external agents over the A2A protocol, Microsoft Foundry agents, Fabric data agents, and Microsoft 365 Agents SDK agents (the last three are

PREVIEW).

| Choose child agents when | Choose connected agents when |
|---|---|
| **One team owns the whole solution** | Different teams own and release parts independently |
| **You want to group tools, instructions, and knowledge logically** | Each agent needs its own settings, for example a different model or authentication |
| **No separate publishing or reuse is needed** | The agent must be published on its own channels and reused by several agents |
| **Shared ALM is acceptable** | Independent ALM per agent is required |

Trade-offs: each hop adds orchestration latency and increases testing and governance surface. An agent that already has connected agents can't itself be connected to a second main agent.

**Agent flows** are deterministic, low-code automations built natively in Copilot Studio with an authoring experience similar to Power Automate. They can run standalone or be attached to an agent as a tool, and they can run prompts, call agents, and include human review steps. Compared with Power Automate cloud flows:

- Billed in Copilot Studio by consumption (Copilot Credits per action) without individual Power Automate licences; they are premium flows that can use premium connectors.
- Support high throughput and low latency, human-in-the-loop actions (request for information, multistage and AI approvals), Key Vault-based secret handling, large data volumes, and parallelism.
- To be added to an agent, a flow must be in a solution and use the **When an agent calls the flow** trigger and the **Respond to the agent** action. Logic can continue after responding, and the response has a size limit.
- They can't be copied, shared, co-owned, or given run-only permissions in Copilot Studio, and can't call desktop flows.
- A cloud flow in a solution can be converted to an agent flow by changing its plan to Copilot Studio. Conversion is one-way.

Copilot Studio also describes a newer **workflows** format with a drag-and-drop designer where steps can reason and act. Agent flows remain the established format on the standard harness, and Power Automate flows can be converted to agent flows but not to the new workflow format.




> **NOTE**
> Topics and agent flows are both deterministic. Microsoft's distinction: topics are optimised for managing conversation; agent flows are optimised for business processes and nonconversational automation.

### 3.3.4 Design prompt actions in Copilot Studio

A **prompt** (built with prompt builder; Power Apps and Power Automate surface the same capability as AI Builder prompts in AI hub) is a reusable natural-language instruction with input variables and optional knowledge, run against a chosen model. Prompts are stored in a prompt library (as Dataverse AI Model

records), support ALM, role-based access, and sharing, and can be used in Copilot Studio (as an agent tool or within a topic), agent flows (**Run a prompt**), Power Automate, and Power Apps.

**Prompt or orchestrator?** The orchestrator uses a fixed system prompt you can't edit; you influence it only through agent instructions and descriptions. A prompt gives full control over model choice, instructions, examples, constraints, and output format. Choose a prompt for structured extraction, classification, precise stylistic output, or when you need features the orchestrator doesn't expose: a broader model range including Microsoft Foundry models, **Dataverse grounding**, **file inputs**, **JSON output**, and **code interpreter**.

**Design parameters:**

- **Inputs**: text, image, or document inputs, plus Dataverse knowledge with selected fields and filters.
- **Output**: text, or JSON when downstream logic needs individual elements (for example invoice fields or multiple categories). With code interpreter, files (including Excel) can be outputs.
- **Model**: mini, general, or deep categories (the documented default is a mini model at basic rate; reasoning models are premium). Choose mini for cost-effective summarisation and extraction, general for complex multimodal tasks, deep for reasoning and analysis. Experimental and preview models aren't for production.
- **Settings**: temperature 0 to 1 (default 0; not available for reasoning models), record retrieval count, include links, code interpreter, and content moderation level (Low, Moderate default, High; only for managed models).
- **Billing**: prompts in Copilot Studio consume Copilot Credits; in Power Apps and Power Automate they consume AI Builder credits first if available.
- **Human oversight**: Microsoft stresses human review of generated content before it reaches customers or informs decisions.

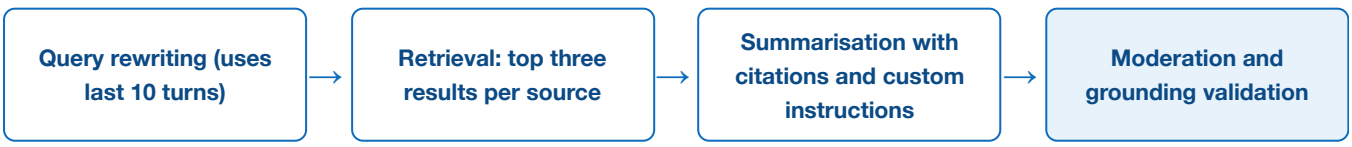
> **COMMON TRAP**
> To search over Excel content, Microsoft recommends code interpreter rather than semantic search, because semantic search has cell-indexing limitations. Similarly, a long multi-document comparison is a poor fit for RAG answers and a better fit for a prompt with a reasoning model or code interpreter.

## 3.4 Data, Foundry Tools, and intelligent apps

### 3.4.1 Design data processing for AI models and grounding

Grounding quality is decided long before the model is called. Microsoft's Azure Well-Architected guidance for grounding data calls data design an iterative, experiment-driven process covering preprocessing, embedding selection, and chunking, and the agent architecture guidance puts it bluntly: truncated or chunked content missing required metadata degrades output regardless of how good the transform and output stages are.

**How Copilot Studio retrieves.** Its RAG pipeline has four steps:




**Dataverse search is the foundation.** It has three index types: relevance search (global search and SearchQuery API), a structured index, and an unstructured index; the last two power Copilot and generative AI experiences, including Copilot Studio agents, Copilot in Dynamics 365, declarative agents, app Copilot, and

Dataverse MCP. File and Dataverse table knowledge in Copilot Studio require Dataverse search to be turned on, and indexes consume Dataverse capacity.

**What happens to uploaded files.** Files uploaded from a device, OneDrive, or SharePoint are stored in Dataverse, **chunked**, and vector-indexed; at query time the most relevant chunks are retrieved. Unstructured connectors also ingest knowledge base articles from Salesforce, ServiceNow, Confluence, and Zendesk into this index.

| Knowledge source | Processing and storage | Security | Choose when |
|---|---|---|---|
| **Upload files** | Copied to Dataverse, chunked, vector-indexed (up to 512 MB per file, 500 files per agent) | Anyone with access to the agent can get answers from them | Static reference documents for all agent users |
| **Upload files > SharePoint (or OneDrive)** | Copied to Dataverse, synchronised about every four to six hours; page-level citations for PDFs | User permission validated against SharePoint; sensitivity labels not supported | High-quality semantic search over a defined set of files or folders; lists supported |
| **SharePoint (connector)** | Stays in SharePoint; queried live through SharePoint search | Security trimmed; sensitivity labels supported; filters by title, author, modified date | Frequently updated content, large libraries, avoid Dataverse storage |
| **Dataverse tables** | Natural language turned into analytical queries; up to 15 tables per source; synonyms and glossary | Requires Authenticate with Microsoft | Structured business records |
| **Azure AI Search** | Your own vectorised index, chunking, and enrichment pipeline | Not delegated: no security trimming | Custom chunking, metadata filtering, very large corpora |
| **Custom data (OnKnowledgeRequested)** | Your flow, connector, or HTTP call returns `Content`, `ContentLocation`, `Title` | Whatever your API enforces | Proprietary search APIs or bespoke retrieval |

**Multiline text and file columns.** In PREVIEW, Dataverse knowledge can apply unstructured reasoning to Multiline Text and File columns once they are marked searchable and added to the table's Quick Find view. Adding the knowledge source before configuring the columns can mean up to two days of backfill.

**Custom pipelines.** A documented case study moved from Dataverse knowledge to an Azure AI Search index: Power Automate collected and converted source documents, a second flow extracted content and metadata into one index with a city attribute, an agent flow queried the index with the user's selected city as a filter and returned the top five chunks as JSON, and an AI prompt generated the answer. This is the pattern to propose when topic-per-partition designs become unmanageable and you need metadata filtering.

**Listing 3.2: Custom knowledge source topic that maps API results to the generative answers schema (from Microsoft Learn)**

```
kind: AdaptiveDialog
beginDialog:
  kind: OnKnowledgeRequested
  id: main
  intent: {}
```

```
  actions:
    - kind: HttpRequestAction
      id: searchRequest
      url: ="https://search-api.contoso.com/search?q=" & System.KeywordSearchQuery
      response: Topic.searchResults
    - kind: SetVariable
      id: setSearchResults
      variable: System.SearchResults
      value: |-
        =ForAll(Topic.searchResults.results,
        { Content: snippet, ContentLocation: url, Title: title })
```

**Preparing difficult content.** For scanned, multimodal, or layout-heavy documents, Azure Content Understanding in Foundry Tools supports RAG ingestion: figure descriptions make charts searchable, layout analysis preserves tables and hierarchy, and annotation detection captures handwriting and strike-throughs. It can output RAG-ready Markdown.

> **COMMON TRAP**
> Azure AI Search as a Copilot Studio knowledge source is not security trimmed. If the requirement says "users must only see content they're permitted to see", either use a delegated source (SharePoint connector, Dataverse, Copilot connectors) or enforce filtering yourself in a custom retrieval flow.

### 3.4.2 Propose Microsoft Foundry Tools for a given requirement

**Foundry Tools** (formerly Azure AI services) are prebuilt and customisable APIs and models, available through REST and SDKs and created as part of a Microsoft Foundry resource. Many can run in containers for on-premises or disconnected scenarios. Choose by requirement:

| Requirement | Foundry Tool | Notes |
|---|---|---|
| **Speech to text, text to speech, speech translation, speaker recognition** | Azure Speech | Custom Speech supports CI/CD for custom models |
| **Translate text or whole documents; custom terminology** | Azure Translator | Real-time, batch document, and custom translation; more than 100 languages |
| **NER, PII detection, sentiment, language detection, summarisation, question answering, custom intents (CLU)** | Azure Language | CLU can drive Copilot Studio classic orchestration; question answering can back a Fallback topic |
| **Extract fields from forms with a prebuilt model (invoices, receipts, IDs) or train a custom extractor** | Azure Document Intelligence | High-accuracy, deterministic extraction from structured documents |
| **No prebuilt model fits; need confidence scores and grounding; multimodal content (documents, images, audio, video); RAG-ready Markdown; post-call analytics** | Azure Content Understanding | Generative AI analyzers with schema-defined fields, classification, and industry prebuilt analyzers |
| **Analyse image or video content** | Azure Vision (Custom Vision for trained image classifiers; Video Indexer for video) | Azure Face is a separate Foundry Tool for face analysis |

| Requirement | Foundry Tool | Notes |
|---|---|---|
| **Enterprise retrieval, vector and hybrid search for RAG** | Azure AI Search | Connectable to Copilot Studio as knowledge |
| **Detect harmful content in user or AI-generated text and images** | Content Safety | Complements platform moderation |
| **Reading assistance and comprehension** | Immersive Reader | Accessibility scenarios |
| **Generative tasks, custom agents, model choice** | Foundry Models and Foundry Agent Service in Microsoft Foundry | Covered further in the extensibility chapter |

**EXAM TIP**

Document Intelligence versus Content Understanding is a favourite distinction. Microsoft's guidance: use Content Understanding when the document type has no prebuilt Document Intelligence model, when you need confidence scores and grounding for automated workflows, or when you need RAG-ready Markdown. Use Document Intelligence for deterministic extraction from structured documents with prebuilt or custom models.

**NOTE**

Before proposing a Foundry Tool from a Power Platform scenario, check whether a low-code wrapper already exists: AI Builder prebuilt models (invoice processing, receipts, text recognition, sentiment, entity extraction) and prompt builder often meet the requirement inside Power Apps, Power Automate, or Copilot Studio without Azure provisioning.

### 3.4.3 Propose code-first generative pages and an agent feed for apps

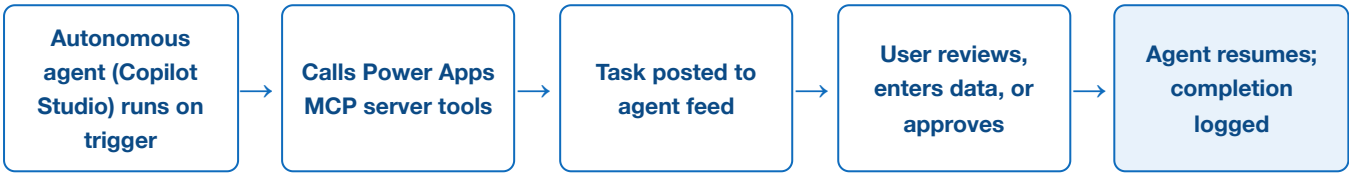
**Generative pages** let makers describe a page in natural language for a model-driven app; an app agent generates React code for UI and business logic, bound to Dataverse tables (up to six in the designer), Power Platform connectors PREVIEW, or Dataverse custom APIs PREVIEW. Pages are solution-aware (exported as UX Agent Project rows with the app and sitemap), can accept `recordId`, `entityName`, and `data` input parameters, can be embedded in a form section, and can send telemetry to Application Insights PREVIEW.

**Code-first path.** Microsoft recommends building generative pages with AI code generation tools such as GitHub Copilot CLI or Claude Code. You install the Power Platform skills plugin (`model-apps`), use the `/genpage` skill, and a planner agent proposes pages, tables, data sources, target app, and solution; specialised agents build TypeScript and React code and deploy with PAC CLI, optionally running generated Playwright tests. Prerequisites include Node.js 18 or later, the latest PAC CLI with an authenticated session, and an existing model-driven app. Pages built this way behave like native generative pages in the designer (restore, undo, code compare).

| Choose | When |
|---|---|
| **Generative page (designer)** | Maker wants a custom page quickly in Power Apps without local tooling |
| **Code-first generative page** | Developers want the newest frontier models, local IDE and source control, several pages plus supporting tables in one run, or connector and custom API bindings |

| Choose | When |
|---|---|
| **PCF control or custom page** | A reusable component or hand-written canvas-based page is required |

**Agent feed** PREVIEW is a task panel at the top of a model-driven app's site map that keeps humans in control of autonomous agent work. Tasks appear in **Needs attention** (logged with the `request_assistance` or `invoke_data_entry` MCP tools) or **Completed** (including agent work logged for review with `request_review`), with an Insights panel for trends. Since 1 May 2026 the feed supports only agents that use the **Power Apps MCP server**; agents without it don't appear. Model-driven apps can host *autonomous agents* (through the MCP server) and *app assistant agents* (custom topics and knowledge in the app).




**COMMON TRAP**

Everyone with access to the Agent Task table can see agent feed items. Microsoft advises not configuring agents to log tasks targeted at specific users. If tasks contain sensitive data, control access through table security before enabling the feed.

### 3.4.4 Design a business process that includes AI components in a Power Apps canvas app

Canvas apps can embed AI at three levels: AI Builder components on screens, AI models and prompts called from Power Fx, and conversational Copilot experiences.

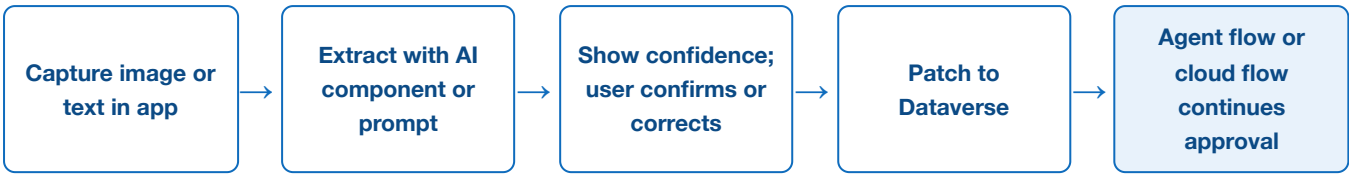
- **AI Builder components** (Insert tab): prebuilt **Business card reader**, **Receipt processor**, and **Text recognizer**; custom-model **Form processor** (document processing) and **Object detector**. Outputs include fields, tables, confidence scores, and bounding boxes, which you should use to drive validation.
- **Power Fx**: call models with `.Predict` (sentiment, entity extraction, key phrase, language detection, category classification), and call prompts the same way after adding the prompt as data. When you move an app to another environment, AI models must be re-added to the app.
- **Conversational**: **Microsoft Copilot in canvas apps** PREVIEW is the recommended natural-language experience; it requires a Microsoft 365 Copilot licence per user, Dataverse search on, and the tenant setting that makes Dataverse data available in Microsoft Copilot, and it's read-only (changes need agent customisation). The older Copilot control can't be added to new apps since 2 February 2026. A custom Copilot Studio agent can also be connected through the app's Copilot settings PREVIEW, and the Copilot answer control PREVIEW shows a predefined question's answer from Dataverse.

**Listing 3.3: Power Fx calling an AI Builder model and a custom prompt (patterns from Microsoft Learn)**

```
// Label.Text: detect language of user input
'Language detection'.Predict(TextInput1.Text).Language

// Button.OnSelect: run a custom prompt and keep the result
Set(result, 'Task identifier'.Predict(TextInput1.Text));
// Label.Text
result.text
```

**Designing the process, not just the screen.** A robust pattern for an expense or field inspection app:




- Put the human confirmation step before persistence whenever confidence is below a threshold or the output feeds a financial or customer-facing decision.
- Keep heavy or long-running AI work (multi-page documents, batch classification) in flows rather than in the app's OnSelect.
- Plan credits: prompts in Power Apps consume AI Builder credits first when available.

### 3.4.5 Apply the Power Platform Well-Architected Framework to intelligent application workloads

Power Platform Well-Architected is based on the Azure Well-Architected Framework and uses five pillars: **Reliability**, **Security**, **Operational Excellence**, **Performance Efficiency**, and **Experience Optimization**. In September 2024 Microsoft added the **intelligent application** workload, defined as a workload that applies Well-Architected best practices to integrate generative AI into business processes, and updated all pillars with Copilot Studio guidance. On Learn, that workload guidance is now surfaced through the Copilot Studio guidance hub (planning, architecture, governance, ALM, testing, and monitoring). For Azure-hosted AI components, align with Azure Well-Architected AI workload guidance.

| Pillar | Intelligent application design decisions |
|---|---|
| Reliability | Fallback and escalation paths; error handling in agent flows; plan throughput and rate limits across Power Automate, Dataverse, connectors, and downstream APIs; default model used as fallback when a selected model is unavailable; avoid experimental models in production |
| Security | Least-privilege connections; end-user versus maker credentials; data policies (including blocking event triggers and knowledge connectors); authentication for Dataverse and SharePoint knowledge; security-trimmed sources; Customer Lockbox; cross-region data movement settings |
| Operational Excellence | Solutions and pipelines for agents, prompts, flows, and generative pages; evaluation and test sets; analytics, transcripts, and Application Insights telemetry; topic enrichment from Fallback data; managed model lifecycle |
| Performance Efficiency | Right-size models (mini, general, deep); limit multi-agent hops; agent flows for high-throughput work; index only needed columns; monitor Copilot Credit consumption |
| Experience Optimization | XO:10 conversation design: explain capabilities, understand intent, clarify instead of failing, offer alternatives, escalate after repeated misunderstandings; citations for trust; human-in-the-loop surfaces such as agent feed |

**PRODUCTION NOTE**

Well-Architected is about trade-offs. Azure's AI design guidance summarises the central one: orchestration gives predictability and control but limits adaptability, whereas agent collaboration enables dynamic problem solving but adds variability. Hybrid designs (deterministic orchestration that delegates complex steps to agents, or escalates to agent reasoning when rules run out) are usually the defensible architecture answer.

## 3.5 Additional practice questions

**Practice 3.1** MULTIPLE CHOICE

Fabrikam uses Dynamics 365 Sales. The Opportunity table has a custom column named **Custom Revenue**. Sellers ask Copilot questions such as "show my open deals with deal value over 50,000" and get poor results. The same sellers also use the Sales agent in Microsoft 365 Copilot in Outlook. You need to improve results in both experiences with the least administrative effort. What should you do?

A. Rename the column display name to Deal Value in the Opportunity table.
B. In Copilot Studio, open the Copilot in Dynamics 365 Sales agent, select SalesSpecificQnA, and add "deal value" as a synonym for the Custom Revenue column.
C. Create a custom topic with trigger phrases that include "deal value" in the Copilot in Dynamics 365 Sales agent.
D. Add an organisational prompt with the alias "deal value" in the Copilot Service admin center.

**Practice 3.2** MULTIPLE CHOICE

Contoso Service uses Copilot in Customer Service. Representatives want the Ask a question experience to return order status from an external order API. The API needs the customer's account number, which is stored on the case's related account. Representatives must not have to type the account number. Copilot Studio extensibility is enabled. You need to design the solution. What should you do?

A. Add a custom connector tool to the Customer Service Copilot Bot agent and define custom context for the account number field on the Extend agent tab in the Copilot Service admin center.
B. Add the account number field as a representative filter for Ask a question.
C. Add a custom connector tool and configure a global variable in the tool to read the account number from the form.
D. Create an organisational prompt that asks the representative to paste the account number.

**Practice 3.3** ORDERING

A partner wants opportunity insights in the Sales agent in Microsoft 365 Copilot to include data from its proposal management application. You need to recommend the sequence of actions. Place the actions in order.

A. Sales administrator enables the connector action for users.
B. Create and test a custom connector for the partner API in Power Platform.
C. Optionally submit the connector and action for certification.
D. Create and publish an action in Copilot Studio based on the connector.

**Practice 3.4** MULTIPLE CHOICE

You connect a Copilot Studio agent to Dynamics 365 Contact Center and attach it to a chat workstream. During testing on the company's staging website, the agent answers questions, but when a customer asks for a person the conversation isn't transferred to a representative. The Escalate topic is configured correctly. What is the most likely cause?

A. The agent uses generative orchestration.
B. The website uses the embed code from Copilot Studio instead of the chat widget embed code from Dynamics 365.
C. The agent's application ID is unique.
D. The representative availability API isn't called.

**Practice 3.5** MULTIPLE CHOICE

An insurer is building a voice IVR agent for Dynamics 365 Contact Center. It has more than 300 topics with many custom entities and a large set of real call transcripts for training. Business owners require full, repeatable control over routing and the best possible speech recognition. The voice channel is licensed through Dynamics 365 Contact Center. Which language understanding option should you recommend?

A. Generative orchestration with no topics.
B. Classic orchestration with built-in NLU.
C. Classic orchestration with NLU+.
D. Classic orchestration with question answering in Azure Language.

**Practice 3.6** MULTIPLE CHOICE

A bank has a mature Azure CLU model for Arabic and French that extracts multiple account entities of the same type from one utterance. You plan to use it to drive an existing Copilot Studio agent that has 40 topics with trigger phrases. What should you expect when you configure the agent to use the CLU model?

A. Trigger phrases are kept and merged with CLU intents automatically.
B. You are prompted to save a snapshot, existing trigger phrases are deleted, and an Analyze Text system topic is added.
C. Generative orchestration must remain turned on.
D. CLU intents are synchronised to topics automatically with no maintenance.

**Practice 3.7** MULTIPLE CHOICE

A Copilot Studio agent using generative orchestration has grown to 45 tools and topics owned by three teams (HR, IT, and Facilities). Testers report that the agent increasingly selects the wrong tool. Each team wants to release changes on its own schedule, and IT wants its part to use a different model. What should you recommend?

A. Convert all tools into child agents of the main agent.
B. Split the solution into separate Copilot Studio agents per team and connect them to a main agent.
C. Switch the agent to classic orchestration with built-in NLU.
D. Increase the number of trigger phrases on each topic.

**Practice 3.8** MULTIPLE CHOICE

You design an autonomous agent that triggers when an email arrives in a shared mailbox, reads attachments, and creates Dataverse rows. The security team is concerned about identity and data exposure. Which statement is correct and should shape your design?

A. Event triggers run with the credentials of whichever user last chatted with the agent.
B. Event triggers use the agent author's credentials, so if the agent is also published to interactive channels, users might act with those credentials; narrow trigger parameters and validate senders.
C. Event triggers are available only with classic orchestration.
D. Administrators can't prevent makers from using event triggers.

**Practice 3.9** MULTIPLE CHOICE

You need a deterministic automation that an agent calls to look up warranty data, pauses for a manager to supply missing information when needed, and returns the result to the agent. Users don't have Power Automate licences. Which design meets the requirements?

A. A Power Automate cloud flow outside a solution with a manual trigger.
B. An agent flow in a solution that uses the When an agent calls the flow trigger, a request for information action, and Respond to the agent.
C. A desktop flow called from the agent flow.
D. A topic with a generative answers node.

**Practice 3.10** MULTIPLE CHOICE

An agent must extract supplier name, invoice number, total, and due date from email bodies and pass them to an agent flow that writes to Dataverse. Results must be consistent run to run. You need to design the extraction step. What should you use?

A. Agent instructions telling the orchestrator to list the fields.
B. A prompt tool with JSON output and temperature 0.
C. A generative answers node over uploaded sample invoices.
D. Conversational boosting with custom instructions.

**Practice 3.11** MULTIPLE CHOICE

An internal agent must answer questions from a SharePoint policy site that is edited several times a day. Some pages carry sensitivity labels that must be honoured, and the organisation wants to avoid Dataverse storage growth. Which knowledge configuration should you use?

A. Upload files > SharePoint.
B. The SharePoint knowledge source that uses the SharePoint connector.
C. Azure AI Search index over the site.
D. Upload a weekly PDF export of the site.

**Practice 3.12** MULTIPLE CHOICE

A logistics company receives customs declarations in many layouts with no matching prebuilt extraction model. Extracted values must include confidence scores so low-confidence items go to human review, and the content must also feed a RAG index as Markdown. Which Foundry Tool should you propose?

A. Azure Document Intelligence prebuilt invoice model.
B. Azure Content Understanding.
C. Azure Translator document translation.
D. Azure Vision image analysis.

**Practice 3.13** MULTIPLE CHOICE

An autonomous Copilot Studio agent reconciles supplier records. When it can't match a record, a user of a model-driven app must enter the missing value, after which the agent continues. You need the request to appear inside the app. What should you implement?

A. A generative page that polls a Dataverse table.
B. Connect the agent to the Power Apps MCP server and use the invoke_data_entry or request_assistance tool so the task appears in the agent feed.
C. A Teams adaptive card sent by a cloud flow.
D. The Copilot answer control on a form.

**Practice 3.14** MULTIPLE CHOICE

You are designing a new canvas app over Dataverse for account managers who all have Microsoft 365 Copilot licences. They want to ask natural-language questions about account data from inside the app. No write-back from chat is required. Which design should you propose?

A. Add the Copilot control to the app.
B. Enable Microsoft Copilot in canvas apps for the environment and app, with Dataverse search turned on.
C. Add an AI Builder text recognizer component.
D. Call a sentiment analysis model with Predict on each screen.

## Answers and explanations

**3.1: B.** Synonyms map alternative names to columns and are authored in Copilot Studio on the SalesSpecificQnA knowledge of Copilot in Dynamics 365 Sales. They are shared with the Sales agent in Microsoft 365 Copilot, so one change covers both. Renaming columns affects all users and apps, topics don't fix metadata interpretation, and organisational prompts belong to Customer Service.

**3.2: A.** Extend agent with Copilot Studio links Ask a question to the Customer Service Copilot Bot, where you add the tool. Context such as the account number is passed through custom context (entity, field, description) defined in the Copilot Service admin center. Filters refine knowledge results; they don't feed tool inputs.

**3.3: B, D, A, C.** Create the custom connector, create and publish the action in Copilot Studio, have the Sales administrator enable it (ingestion can take up to seven days), then optionally certify.

**3.4: B.** Handoff works only with the chat widget embed code configured in Dynamics 365. The Copilot Studio embed code bypasses the omnichannel connection. A unique application ID is required, not a fault, and availability APIs are optional.

**3.5: C.** NLU+ targets large enterprise agents with many topics, entities, and training samples, gives classic deterministic control, uses training data to optimise speech recognition, and is available with a Dynamics 365 Contact Center licence.

**3.6: B.** CLU requires classic orchestration. Copilot Studio prompts you to save a botContent.zip snapshot, deletes trigger phrases, and adds the Analyze Text system topic. CLU intents and topics must be kept in sync manually.

**3.7: B.** Selection precision tends to degrade beyond 30 to 40 choices. Independent release schedules, separate models, and separate ownership are documented reasons to use connected agents rather than child agents, which share settings and ALM.

**3.8: B.** Event triggers authenticate with the author's credentials and require generative orchestration. Microsoft warns that publishing such an agent can expose those credentials to users, and recommends narrowing trigger scope and validating inputs. Administrators can block event triggers with data policies.

**3.9: B.** Agent flows are billed through Copilot Studio without Power Automate licences, support human-in-the-loop request for information, and must be solution-aware with the agent trigger and Respond to the agent action. Agent flows can't call desktop flows.

**3.10: B.** Prompts give full control over instructions and output format; JSON output exposes each field to downstream logic, and temperature 0 favours predictable results. The orchestrator's system prompt can't be edited and isn't designed for structured extraction.

**3.11: B.** The SharePoint connector option queries content in place in real time, supports sensitivity labels and security trimming, and avoids Dataverse storage. Upload files > SharePoint copies content to Dataverse, refreshes every four to six hours, and doesn't support sensitivity labels.

**3.12: B.** Content Understanding is recommended when no prebuilt Document Intelligence model fits, when confidence scores and grounding are needed for automation, and when RAG-ready Markdown output is required.

**3.13: B.** Agent feed tasks come from agents that use the Power Apps MCP server. invoke_data_entry and request_assistance create Needs attention items that users complete so the agent can continue.

**3.14: B.** Microsoft Copilot in canvas apps is the recommended experience, needs Microsoft 365 Copilot licences and Dataverse search, and is read-only. The Copilot control can't be added to new canvas apps since 2 February 2026.

### Key takeaways

- Business terms for Sales are synonyms and glossary entries on SalesSpecificQnA in Copilot Studio, shared with the Sales agent in Microsoft 365 Copilot; Customer Service vocabulary is shaped by Customer Support agent instructions and prompt aliases.
- Customer Service Copilot is configured across Power Platform admin center (enablement, data movement) and Copilot Service admin center (experiences, Customer Support agent, filters, prompts, Extend agent, custom context).
- Sales "connectors" means three patterns: extend the Sales Copilot agent in Copilot Studio, extend the Sales agent in Microsoft 365 Copilot with a custom connector action, or ground with Copilot or real-time connectors.
- Contact Center integration: unique app ID per agent, same environment, workstream attachment, Escalate topic for handoff, chat widget embed code, and real-time voice trigger limitations.
- Agent types scale from prompt and response (knowledge) to task (tools) to autonomous (triggers), and controls must scale with them.
- Generative orchestration is the default; classic NLU, NLU+, and CLU exist for deterministic control, Contact Center scale and voice accuracy, or custom multilingual models.
- Conversational boosting runs before Fallback; design Fallback to clarify, offer alternatives, offload Q&A, and escalate; enrich topics from fallback data.
- Agent flows are deterministic, Copilot Credit-billed, solution-aware tools with human-in-the-loop actions; prompts give full control over model, inputs, JSON output, and code interpreter.
- Grounding choices trade freshness, security trimming, storage, and retrieval quality; Azure AI Search isn't security trimmed.
- Use Content Understanding for unknown layouts and RAG-ready output, Document Intelligence for deterministic structured extraction.
- Generative pages are best built code-first; agent feed needs the Power Apps MCP server; canvas apps use AI Builder components, Predict, and Microsoft Copilot in canvas apps.
- Apply all five Power Platform Well-Architected pillars to intelligent apps and explicitly document the predictability versus adaptability trade-off.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Customize Copilot in Dynamics 365 Sales
- Set up Sales agent in Microsoft 365 Copilot
- FAQ about Copilot in Dynamics 365 Sales
- Extend Sales agent with partner applications (preview)
- Create a custom connector and connector action (preview)
- Manage extensions for Sales in Microsoft Copilot (preview)
- Copilot connectors overview
- Customize Copilot chat using Copilot Studio (preview)
- Manage Copilot features in Customer Service
- Configure Copilot features in the help pane
- Extend Copilot with Copilot Studio (preview)
- Integrate a Copilot agent
- Configure handoff to Dynamics 365 Customer Service
- Integrate a voice-enabled agent with Dynamics 365 Contact Center

- Configure real-time agents
- Overview of channels
- Extend Dynamics 365 Contact Center
- Dynamics 365 Contact Center overview
- AI agent adoption (Cloud Adoption Framework)
- Using agents in Microsoft Copilot Chat
- Harnesses in Copilot Studio
- Design autonomous agent capabilities
- Event trigger overview
- Monitor autonomous agents
- Natural language understanding (NLU) overview
- Design effective language understanding
- Configure NLU+
- Get started with conversational language understanding integration
- Holland America Line transforms customer experience with Copilot Studio
- Set topic triggers
- Use system topics
- Use the Fallback topic
- Topic enrichment analysis
- Follow topic authoring best practices
- Create and edit topics
- Add other agents overview
- Agent flows overview
- Agent flows in Microsoft Copilot Studio FAQ
- Plan and design integration strategies
- Use the agent design framework
- Use agent tools to extend, automate, and enhance your agents
- Overview of prompts
- Change the model version and settings
- JSON output
- Use code interpreter in a prompt
- Enhance AI responses by using Retrieval Augmented Generation
- Unstructured data as a knowledge source
- Upload files as a knowledge source
- Add Dataverse tables as a knowledge source
- Add knowledge to an agent
- Custom knowledge sources
- What is Dataverse search?
- Dunaway streamlines city code research with Copilot Studio
- Grounding data design for AI workloads on Azure
- Map agent flows to requirements
- What are Foundry Tools?
- What is Azure Content Understanding in Foundry Tools?
- What is Azure Document Intelligence in Foundry Tools?
- Choose an Azure AI targeted language processing technology
- Generate a page using natural language
- Create and edit generative pages with AI code generation tools
- Add agents to your model-driven app (preview)
- Supervise agents in model-driven apps with agent feed (preview)
- AI Builder in Power Apps overview
- Use AI Builder models in Power Apps
- Use your prompt in Power Apps

- Add Microsoft Copilot for app users in canvas apps (preview)
- Add a Copilot control to a canvas app (preview)
- Add a custom Copilot to a canvas app (preview)
- What is Power Platform Well-Architected?
- What's new in Power Platform Well-Architected
- Recommendations for designing conversational user experiences (XO:10)
- Define your solution architecture (Copilot Studio guidance)
- Application design for AI workloads on Azure

DOMAIN 2 · PART B

# Extensibility and prebuilt agents and apps

**Part of Domain 2: Design AI-powered business solutions (25–30%)**

The second half of the design domain asks two architect-level questions. First, when an out-of-the-box agent or Copilot is not enough, **which extensibility mechanism** should carry the requirement: a custom or fine-tuned model in Microsoft Foundry, a declarative or custom engine agent in Microsoft 365 Copilot, a Copilot Studio tool (connector, custom connector, agent flow, REST API, Model Context Protocol server, computer use), a connected agent over the Agent2Agent (A2A) protocol, or a behaviour change such as deep reasoning or voice? Second, when Microsoft already ships the capability, **how do you orchestrate the configuration** of prebuilt agents and apps across Dynamics 365 finance and supply chain, Dynamics 365 customer experience and service, Microsoft 365 Copilot, and Power Platform, including grounding the finance and operations chat experiences in extra knowledge?

Expect scenario questions where several options would technically work and only one meets the stated constraint (least custom code, data residency, governance by data policies, licensing, latency, or reuse across many agents). Study the decision tables in this chapter, learn the prerequisites that break deployments (Power Platform Integration, cross-region data movement, Activity protocol endpoints, Allowed MCP Clients, Teams setup policies), and keep a mental list of what is still in preview.

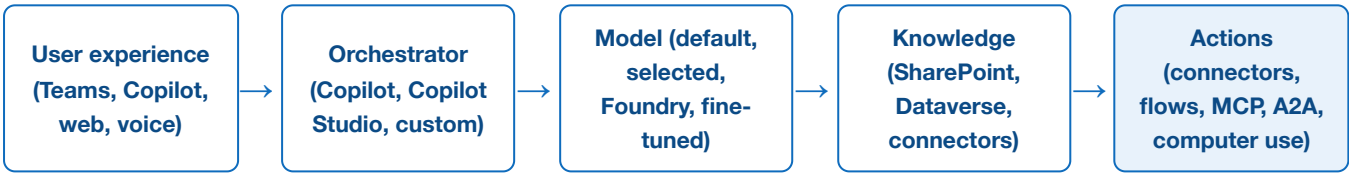
| Official skill (Design AI-powered business solutions) | Section |
|---|---|
| **Design extensibility of AI solutions** | |
| Design AI solutions by using custom models in Microsoft Foundry | 4.1.2 |
| Design agents in Microsoft 365 Copilot | 4.1.3 |
| Design agent extensibility in Copilot Studio | 4.1.1, 4.1.4 |
| Design agent extensibility with Model Context Protocol in Copilot Studio | 4.1.5 |
| Design agents to automate tasks in apps and websites by using Computer Use in Copilot Studio | 4.1.6 |
| Design agent behaviors in Copilot Studio, including reasoning and voice mode | 4.1.7 |
| Optimize solution design by using agents in Microsoft 365, including Teams and SharePoint | 4.1.8 |
| **Orchestrate configuration for prebuilt agents and apps** | |
| Orchestrate AI features in Dynamics 365 apps for finance and supply chain | 4.2.1 |
| Orchestrate AI features in Dynamics 365 apps for customer experience and service | 4.2.2 |
| Propose Microsoft 365 agents for business scenarios | 4.2.3 |
| Orchestrate the configuration of Microsoft 365 Copilot for Sales and Microsoft 365 Copilot for Service | 4.2.4 |
| Propose Microsoft Power Platform AI features, including AI hub | 4.2.5 |
| Design interoperability of the finance and operations agent chats to use additional knowledge sources | 4.2.6 |

| Official skill (Design AI-powered business solutions) | Section |
|---|---|
| Related: process of adding knowledge sources to in-app help and guidance for Dynamics 365 Finance or Supply Chain Management | 4.2.7 |
| Practice, takeaways, and references | 4.3 |

# 4.1 Design extensibility of AI solutions

## 4.1.1 The extensibility landscape: surfaces, harnesses, and layers

Before choosing a mechanism, place the requirement on a layer. Microsoft's agent stack separates **knowledge** (what the agent can read), **actions** (what it can do), the **orchestrator** (how it plans and selects tools), the **foundation model** (reasoning and generation), and the **user experience layer** (where people meet the agent). Most extensibility decisions change exactly one layer. Adding a SharePoint library changes knowledge; adding an MCP server changes actions; replacing the orchestrator and model moves you from a declarative agent to a custom engine agent; fine-tuning in Foundry changes the model.




Copilot Studio now describes the runtime that sits between your design and the model as a **harness**. The harness decides when to call the model, what to send, how to interpret the response, and which tools to call. The harness you build on affects billing, features, and capabilities, so it is an architecture decision, not a maker preference.

| Harness | Best for | Behaviour | Publishing |
|---|---|---|---|
| **GitHub Copilot harness** | Complex, multi-step business processes (for example accounts payable: read invoices, match to purchase orders, route exceptions) | Takes a goal, breaks it into steps, calls connectors, knowledge, MCP, and connected agents, retries and finds alternative paths, creates and edits Word, Excel, PowerPoint, and PDF files, supports skills and memory, runs each task in a sandbox | Internal teams or external customers |
| **Standard harness** | Rule-based agents, structured and repeatable conversations and agent flows | Follows the topics, prompts, and paths you define; predictable | Internal teams or external customers |
| **Copilot chat harness** | Extending Microsoft 365 Copilot Chat with enterprise knowledge | Connects enterprise knowledge to Copilot Chat | Internal teams |

**NOTE**

The GitHub Copilot harness is a Copilot Studio authoring and orchestration framework, not the GitHub Copilot service. Learn states that customer data isn't sent to or processed by the GitHub Copilot service, and existing Copilot Studio privacy, security, compliance, and data residency commitments continue to apply. Many feature articles (MCP onboarding wizard, computer use, deep reasoning, voice) currently state that they describe the **standard harness**; equivalent articles for the GitHub Copilot harness are labelled preview.

**EXAM TIP**

Read scenario wording for clues: "predictable", "consistent answers", "help desk" point to the standard harness; "reason through a goal", "recover when a step fails", "produce Excel or PowerPoint output" point to the GitHub Copilot harness; "employees get grounded answers without leaving Copilot Chat" points to the Copilot chat harness.

### 4.1.2 Custom models in Microsoft Foundry

Microsoft Foundry (formerly Azure AI Foundry) provides the model catalogue, deployments, evaluation, and model customisation used when a business solution needs more than the models built into Copilot Studio or Microsoft 365 Copilot. An architect must decide **whether** to customise a model, **how** (which method and training type), **where** to serve it, and **how** Copilot Studio or Microsoft 365 will consume it.

#### When fine-tuning is the right lever

Fine-tuning adapts a pretrained model through additional training on task-specific data, changing the weights rather than adding examples to each prompt. Learn lists the outcomes it can help with when you have high-quality, task-specific data: improved accuracy and relevance, reduced prompt overhead (fewer examples, lower token cost and latency), consistent style and structure, better tool selection and argument accuracy, better use of retrieved context, and specialising a smaller model with teacher-generated examples (distillation). Equally important is what it does *not* do: fine-tuning adds training and hosting costs and **doesn't replace retrieval for current information or application-level safety controls**.

| Requirement | Preferred approach | Why |
|---|---|---|
| **Answers must reflect documents that change weekly** | Retrieval (knowledge sources, Azure AI Search, Dataverse) with a base model | Fine-tuning doesn't keep a model current; retrieval does |
| **Consistent JSON schema or house style on every response, long few-shot prompts are expensive** | Supervised fine-tuning (SFT) | Teaches the format; removes prompt examples, reducing tokens and latency |
| **Quality depends on preferences (tone, completeness) rather than one correct answer** | Preference fine-tuning with direct preference optimisation (DPO) | Trains on preferred versus rejected responses to the same input |
| **Many valid solutions, but quality can be scored reliably by a grader** | Reinforcement fine-tuning (RFT) | Uses prompts and a grader that scores generated responses |
| **High-volume narrow task where a large model is accurate but costly** | Distillation into a smaller model (SFT on teacher-generated data) | Lower cost and latency at acceptable quality |

| Requirement | Preferred approach | Why |
|---|---|---|
| **Behaviour can be fixed with clearer instructions or a structured prompt** | Prompt engineering, Copilot Studio prompt tool | No training cost or model lifecycle to manage |

Learn also recommends stacking techniques: SFT first to specialise, then DPO to align responses to preferences.

### Training approaches, training types, and deployment

Foundry offers **managed fine-tuning** (submit data and settings, Foundry runs the job; supports model-specific SFT, DPO, and RFT, and distillation) and **interactive training** PREVIEW (write a Python loop controlling losses, rewards, rollouts, and checkpoints; requires access approval). Neither requires you to provision training GPUs. Start with managed fine-tuning unless the team has deep ML expertise.

| Training type | Where training runs | Architectural consequence |
|---|---|---|
| Standard | Region of the Foundry resource | Guarantees data residency; choose when data must stay in a region |
| Data zone | Within a supported data zone (currently only the US data zone) | Zone-level residency |
| Global | Capacity beyond your region; data and weights are copied to the training region | Cheaper and faster queues; no data residency |
| Developer | Idle capacity, preemptible | Largest savings; no latency or SLA guarantee and no data residency; for experimentation |

Training types are separate from deployment: a fine-tuned model can be served **serverless** (Foundry hosts it), on **managed compute** PREVIEW (dedicated GPUs running a base model with LoRA adapters), or through **Fireworks on Foundry** PREVIEW, with deployment types from Standard and Global Standard to Developer (evaluation only, 24 hours, no SLA) and Provisioned Throughput. Check that model, method, training type, and deployment option are compatible; an adapter isn't interchangeable across similarly named base models.

> **PRODUCTION NOTE**
> Budget for iteration. SFT and DPO are billed on training tokens multiplied by epochs; RFT is billed on training time plus grading. You aren't charged for queue time, failed jobs, or jobs cancelled before training starts. Plan held-out evaluation (separate training, validation, and test data), test RFT graders against adversarial responses, and schedule retraining when the base model approaches retirement.

### Consuming Foundry models and agents from business applications

- **Prompts in Copilot Studio, Power Apps, and Power Automate.** The prompt tool can use models built into the platform or a model deployed through Foundry. In the prompt's model list you select **+** to connect a Foundry model, entering the **model deployment name** and **base model name** exactly as they appear in Foundry. The integration currently supports chat-completion models. Foundry models are reached through a connector, so admins govern them with data policies in the Power Platform admin center (the connector is listed as "Azure AI Foundry"). Learn recommends applying responsible AI policies in Foundry for these models.

- **Foundry agents connected to Copilot Studio** PREVIEW. A Copilot Studio agent can call a Foundry agent as a connected agent. The Foundry agent must have the **Activity protocol** endpoint enabled (new agents expose only Responses and A2A by default; enable Activity with the REST API or Python SDK, not the portal), and only agents created in the new Foundry portal are supported (older ones fail with "404 - Version not found").
- **Foundry agents in Microsoft 365 Copilot and Teams.** Publish directly from the Foundry portal (provisions Azure Bot Service and Entra ID automatically, minimal setup) or build a proxy app with the Microsoft 365 Agents Toolkit when you need custom logic, SSO, or multi-environment deployment. Either way, the Foundry agent becomes a custom engine agent from Microsoft 365's point of view.

**COMMON TRAP**

A first test call from Copilot Studio to a Foundry agent that returns HTTP 400 "endpoint does not support activity" is not a permissions problem: the Activity protocol endpoint isn't enabled on the Foundry agent. The Foundry portal's Endpoints section still shows only Responses and A2A after you enable it; that is expected.

### 4.1.3 Agents in Microsoft 365 Copilot

Microsoft 365 Copilot can be extended with agents built in two ways. A **declarative agent** uses your instructions, knowledge, and actions together with Copilot's own orchestrator and foundation models. A **custom engine agent** brings its own orchestrator and models. The choice determines hosting, compliance responsibility, channels, and whether the agent can act proactively.

| Dimension | Declarative agent | Custom engine agent |
|---|---|---|
| **Orchestrator and model** | Copilot's orchestrator and models | Your choice of orchestration and models (foundation, small, fine-tuned, industry-specific) |
| **Hosting** | Hosted in Microsoft 365; no extra hosting | Hosted in Microsoft 365 with Copilot Studio, or externally (for example Azure) at additional cost |
| **Compliance and RAI** | Inherits Microsoft 365 security, compliance, and responsible AI standards | You must ensure your own compliance, RAI practices, and security |
| **Proactive interactions** | Not supported; user-initiated | Supported: can trigger actions without direct user input |
| **Collaboration** | Designed for individual use | Individual and group collaboration (Teams channels and meetings); agent-to-agent delegation |
| **Channels** | Microsoft 365 Copilot and Microsoft 365 apps (Teams, Word, Excel, Outlook) | Microsoft 365 and external apps (web, SMS, email, other messaging channels) |
| **Tooling** | Agent Builder in Microsoft 365 Copilot, Copilot Studio, Microsoft 365 Agents Toolkit (Visual Studio, VS Code), SharePoint | Copilot Studio (low-code) or Agents Toolkit with Microsoft 365 Agents SDK in .NET, Python, or JavaScript; Foundry agents |

Learn's guidance: build declarative when you want Copilot's orchestration for consistency with security and compliance, want faster or low-code delivery, and the workflow lives inside Microsoft 365 (an IT help desk agent responding to @mentions in Teams, a document summarisation agent in SharePoint). Build a custom engine agent when you need custom orchestration and precise business rules (a loan approval agent calling

multiple credit-check systems), your own or domain-specific models, group productivity in Teams channels or meetings, an existing assistant you want to bring into Microsoft 365, availability outside Microsoft 365, or proactive messaging.

### Anatomy of a declarative agent manifest

A declarative agent is defined by a manifest referenced from the Microsoft 365 app package. The latest schema at the time of writing is **v1.8**, which added the `EmailActions` and `MeetingActions` capabilities. Required properties are `version`, `name` (up to 100 characters), `description` (up to 1,000), and `instructions` (up to 8,000). Optional properties include `capabilities` (at most one of each type), `conversation_starters` (up to 12), `actions` (1 to 10 API plugins), `behavior_overrides`, `disclaimer`, `sensitivity_label`, `editorial_answers`, `worker_agents` PREVIEW, and `user_overrides`.

**Listing 4.1: Minimal declarative agent manifest scoped to a SharePoint library, Dataverse tables, and one API plugin**

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/copilot/declarative-
agent/v1.8/schema.json",
  "version": "v1.8",
  "name": "Account planning agent",
  "description": "Helps account managers prepare account plans from approved content.",
  "instructions": "Answer only from the configured SharePoint library and Dataverse tables. Cite
sources. Use the createTask action only when the user asks to create a follow-up task.",
  "capabilities": [
    {
      "name": "OneDriveAndSharePoint",
      "items_by_url": [
        { "url": "https://contoso.sharepoint.com/sites/sales/Shared%20Documents/AccountPlans" }
      ]
    },
    {
      "name": "Dataverse",
      "knowledge_sources": [
        {
          "host_name": "contoso.crm.dynamics.com",
          "tables": [ { "table_name": "account" }, { "table_name": "opportunity" } ]
        }
      ]
    }
  ],
  "conversation_starters": [
    { "title": "Plan", "text": "Draft an account plan for Fabrikam" }
  ],
  "actions": [
    { "id": "tasksPlugin", "file": "tasks-api-plugin.json" }
  ],
  "behavior_overrides": {
    "special_instructions": { "discourage_model_knowledge": true }
  }
}
```

**NOTE**

Capability objects available in v1.8 include Web search, OneDrive and SharePoint, Copilot connectors ( `GraphConnectors` ), Graphic art, Code interpreter, Dataverse, Teams messages, Email, Email actions, People, Scenario models, Meetings, Meeting actions, and Embedded knowledge. Users can access declarative agents that use any capability other than Web search only if the tenant allows metered usage or the user has a Microsoft 365 Copilot licence.

#### Building agents for Microsoft 365 from Copilot Studio

In Copilot Studio you can produce two artefacts for Microsoft 365 Copilot. An **agent for Microsoft Copilot** (called a declarative agent in Microsoft 365) is created from the Microsoft Copilot page in Copilot Studio, uses instructions, knowledge (SharePoint, Teams chats and meetings, emails, websites, Copilot connectors), capabilities (code and documents, images), and tools, and is published to the organisation's catalogue; it isn't automatically deployed when published. A **custom agent** (custom engine agent) has its own Copilot Studio orchestration and can be published to the **Teams and Microsoft 365 Copilot** channel as well as other channels. Microsoft 365 Copilot caches answers within a session, so use a new test session when testing changes, and note that Copilot (declarative) agents don't collect data for the Copilot Studio **Monitor** page.

#### Distribution and governance




- Agents published to the organisation through the Teams and Microsoft Copilot channel appear as **requested agents** in Copilot controls in the Microsoft 365 admin center; admins approve or reject, assign to users or groups, and can show them in the Teams app store.
- Admins can **upload a custom agent** as a .zip package (downloadable from Copilot Studio under Channels > Teams and Microsoft Copilot > Availability options > Download .zip) instead of requiring the maker to publish.
- Declarative agents built with SharePoint can't be published to an organisation catalogue.
- Agents Toolkit can publish to the commercial marketplace, the organisation's catalogue, or for personal use.

**COMMON TRAP**

"Use Microsoft 365 Agents SDK" is not the answer when the requirement is "fastest delivery, inherits Microsoft 365 compliance, no extra hosting". That is a declarative agent. Choose the SDK (custom engine) when the stem mentions proactive messages, group chat or channel participation, your own model, or channels outside Microsoft 365.

### 4.1.4 Agent extensibility in Copilot Studio

Copilot Studio extends an agent with **tools**, the building blocks that let it interact with external systems in response to users or autonomous triggers. With generative orchestration (on by default) the agent selects tools, topics, and knowledge dynamically; in classic mode it can only use topics, although topics can still call tools explicitly.

| Tool mechanism | What it is | Design notes |
|---|---|---|
| Prebuilt connector | One of the Power Platform connectors to Microsoft and non-Microsoft services | Fastest path when a connector exists; governed by data policies |
| Custom connector | Your own Power Platform connector over a custom API (OpenAPI definition) | Reusable across Power Apps, Power Automate, and agents; needs view and share permissions for the organisation for agent use; also the infrastructure behind A2A and manually defined MCP connections |
| Agent flow | Deterministic automation built in Copilot Studio | Optimised for business processes; billed through Copilot Studio consumption, not Power Automate licences; can't be copied, shared, co-owned, or given run-only permissions |
| Prompt | Single-turn model-based prompt with inputs, knowledge, and optional code interpreter | Choose any built-in model or a Foundry model; stored in a prompt library with ALM and RBAC |
| REST API PREVIEW | Endpoints from an OpenAPI specification added as tools | Requires OpenAPI v2 (v3 is translated automatically); auth None, API key, or OAuth 2.0; stored in a solution |
| Model Context Protocol | Connection to an MCP server that publishes tools and resources | Descriptions and inputs come from the server; see 4.1.5 |
| Computer use | Vision-based automation of websites and desktop apps | For systems without APIs; see 4.1.6 |
| Bot Framework skills, client tools | Tool-like behaviour through Azure Bot Service skills or an event sent to the client | Specialist scenarios; client tools let the hosting app perform an action |

Every tool has a **name and description** that the orchestrator uses to decide when to call it, an option to **let the agent decide dynamically** (or only call it from topics), an option to **ask the end user before running**, and an **authentication** mode: end user credentials (users only reach data they are authorised to see) or maker-provided credentials (shared resources, or when users shouldn't need individual access). Inputs default to *Dynamically fill with AI*, so the orchestrator generates questions to collect missing values; you can override with a custom value or Power Fx. Completion settings control whether the agent writes a generative response, sends a specific response, or sends an adaptive card.

**PRODUCTION NOTE**

Generative orchestration supports up to 128 tools per agent, but Learn recommends 25 to 30 for best results. Each child agent has its own orchestration and its own budget of up to 128 tools. Data policies apply: a tool whose connector is blocked appears disabled in the Add tool pane with an information icon explaining the restriction.

#### Multi-agent extensibility: child, connected, and external agents

An agent can also be extended with other agents, listed on its **Agents** page:

- **Child agents**: lightweight agents inside the main agent that group tools, instructions, and knowledge.
- **Connected Copilot Studio agents** in the same environment.

- **External agents**: agents over the **A2A protocol** (generally available since April 2026), **Microsoft Foundry agents** PREVIEW, **Fabric data agents** PREVIEW, and **agents built with the Microsoft 365 Agents SDK** PREVIEW.

Use child agents when a single team owns the solution, you don't need separate settings, authentication, deployment, or ALM, and you won't reuse the subagent elsewhere. Split into connected agents when tool selection precision degrades (rule of thumb: more than 30 to 40 choices of tools, topics, and agents, or fewer choices with overlapping descriptions), when different teams own parts, when parts must be published independently or reused by several agents, or when they need their own model or ALM. Expect higher latency (each connected agent runs its own orchestration) and a larger testing and governance surface.

> **COMMON TRAP**
> An agent that itself has connected agents can't also be used as a connected agent by a second main agent. An agent without connected agents can be connected to many main agents. Child and connected agents also respect the main agent's **Use general knowledge** setting, and citations might not always survive the hand-back to the calling agent.

A2A is designed for agent workflows: it supports multiturn interactions and rich contextual metadata (Copilot Studio sends a `contextId`, locale, and the full chat history in message metadata) and is interoperable across frameworks, unlike a plain HTTP connector. Copilot Studio reads the agent card from `/.well-known/agent.json` to prefill the name and description, supports None, API key, or OAuth 2.0 authentication, and because A2A connections use the custom connector infrastructure they can reach agents on-premises or inside a virtual network.

The reverse direction exists too: agents on the GitHub Copilot harness can be exposed to external clients through **MCP client** and **A2A client** channels PREVIEW, secured with an Entra app registration and OAuth.

### Decision table: choosing the extensibility option

| Need | Choose | Rather than |
|---|---|---|
| **Call a SaaS service that already has a certified Power Platform connector** | Prebuilt connector | Custom connector or REST API (unnecessary build) |
| **Call an internal API that several apps, flows, and agents will reuse, governed by data policies** | Custom connector | REST API tool defined per agent |
| **Quickly add a few endpoints from an OpenAPI file to one agent** | REST API tool PREVIEW | Building an MCP server (setup overhead) |
| **Multistep deterministic business process with approvals, loops, and connectors** | Agent flow (or an existing Power Automate cloud flow) | Long topic chains or prompt logic |
| **Expose a catalogue of tools to many agents and clients; API changes often; descriptions maintained centrally** | MCP server | Connectors whose action descriptions must be updated per agent |
| **Delegate a task to an agent built on another framework with its own reasoning, multiturn** | A2A connected agent | HTTP call that loses conversational context |
| **Reuse a Foundry agent with custom models and tools** | Connected Foundry agent PREVIEW | Re-implementing its logic in Copilot Studio |

| Need | Choose | Rather than |
|---|---|---|
| **Operate a website or desktop app that has no API** | Computer use | Screen-scraping custom code |
| **Stable UI, high volume, strict GA-only policy, existing RPA team** | Power Automate desktop flow (RPA) | Computer use |
| **Precise, structured generation (strict format, complex extraction)** | Prompt tool, optionally with a Foundry model | Relying on the orchestrator's fixed system prompt |

### 4.1.5 Extensibility with Model Context Protocol in Copilot Studio

Model Context Protocol (MCP) is an open, universal interface for AI models to use external tools and data. An MCP server can publish **resources** (file-like data such as API responses or file contents), **tools** (functions the model can call), and **prompts** (templates). Copilot Studio currently supports **tools and resources**. Each tool or resource the server publishes is automatically available to the agent with the name, description, inputs, and outputs the server provides, and Copilot Studio reflects updates and removals dynamically, so agents always use the latest definitions without republishing.

#### Design rules to memorise

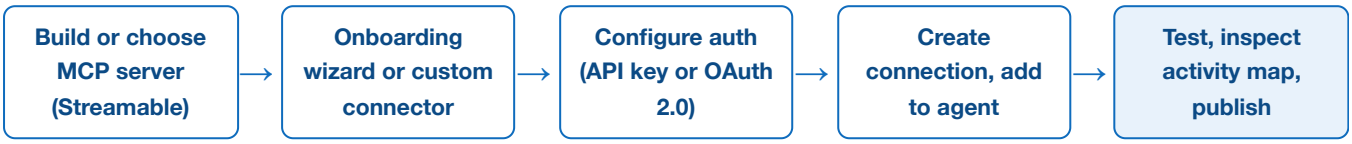
- **Generative orchestration must be turned on** to use MCP.
- Copilot Studio supports the **Streamable HTTP** transport. SSE is deprecated in the MCP specification and is no longer supported by Copilot Studio after August 2025.
- Two ways to connect an existing server: the **MCP onboarding wizard** on the agent's Tools page (recommended) or a **custom connector** created in Power Apps from an OpenAPI schema. You can also register a server through **Bring your own (BYO) MCP server** with the Agents 365 CLI and Microsoft 365 admin center, so it becomes available in Copilot Studio under admin approval and visibility controls.
- Authentication options: **None**, **API key** (header or query), or **OAuth 2.0** with *Dynamic discovery* (dynamic client registration with discovery, the simplest), *Dynamic* (DCR without discovery; you supply authorisation and token URLs), or *Manual* (client ID, secret, authorisation, token, and refresh URLs, scopes). OAuth produces a callback URL to register with the identity provider.
- **Data policies apply**: MCP access relies on Power Platform connectors, so a data policy that regulates connectors also regulates the MCP server and its tools.
- Limitations: you can't enrich the server's tool descriptions with extra "when to invoke" context in Copilot Studio, and **topics can't call MCP servers directly**.
- Built-in MCP servers exist for services such as Outlook, Dataverse (which enforces Dataverse RBAC), GitHub, Salesforce, and Jira; partner servers can carry **MCP certification**.

**Listing 4.2: OpenAPI (Swagger 2.0) schema for a custom MCP connector using the streamable transport (fictional host)**

```
swagger: '2.0'
info:
  title: Contoso Inventory MCP
  description: Streamable MCP server exposing inventory tools for Copilot Studio
  version: 1.0.0
host: mcp.contoso.com
basePath: /
schemes:
  - https
paths:
  /mcp:
    post:
```

```
    summary: Contoso Inventory MCP server
    x-ms-agentic-protocol: mcp-streamable-1.0
    operationId: InvokeMCP
    responses:
      '200':
        description: Success
```

The `x-ms-agentic-protocol: mcp-streamable-1.0` extension is what tells Power Platform that the operation is an MCP endpoint rather than an ordinary REST operation. After the connector exists, add it to the agent from the Tools page; the configuration page then shows **Tools** and **Resources** sections instead of Inputs and Completion.




### MCP or something simpler?

Learn's guidance is explicit. Use MCP when you need a **standardised, centrally managed** way to expose tools and resources to **multiple agents** without per-client configuration, especially when upstream APIs change often: change the definition once on the server and every consuming agent picks it up. If no MCP server exists, or you are rapidly prototyping, calling APIs directly is faster and avoids the MCP lifecycle overhead. For Dynamics 365 finance and operations apps, Microsoft ships the **Dynamics 365 ERP MCP server** (section 4.2.1), which makes MCP the default answer for agent access to ERP data and business logic.

**EXAM TIP**

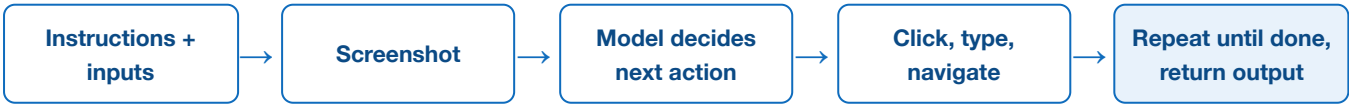
If a stem says the MCP server uses SSE, the fix is to move to the Streamable transport. If it says makers can see the server in the Add tool pane but it is greyed out, think data policy. If it says a topic must call an MCP tool directly, that isn't supported; let generative orchestration call it or use a connector or flow.

**PRODUCTION NOTE**

When you connect to a non-Microsoft or external MCP server, you are responsible for the tools and resources the agent accesses. Keep the number of servers per agent small (each counts toward the agent's tool limit, and concurrently running server instances per turn are capped on the GitHub Copilot harness), review every tool description before production, and use the activity trace to confirm which tool was called with which arguments.

## 4.1.6 Automating apps and websites with computer use

**Computer use** is a Copilot Studio tool that lets an agent operate a Windows computer, selecting buttons, choosing menus, and typing into fields in websites and desktop apps, using a virtual mouse and keyboard. It is powered by Computer-Using Agents (CUA), models that combine vision with reasoning. At each step the agent takes a screenshot, decides the next action, executes it, and repeats until the task is complete; screenshots and reasoning steps are kept in the run history. Because it reasons over what it sees, it adapts when buttons or screens change. It requires generative orchestration.



### Configuration decisions

| Setting | Options and guidance |
|---|---|
| Model | OpenAI Computer-Using Agent (standard, generally available), Anthropic Claude Sonnet 4.5 (standard, generally available), Claude Sonnet 4.6 (standard, experimental), Claude Opus 4.6 (premium, experimental). Anthropic models require the admin to allow external models for the environment. |
| Machine | **Hosted**: shared pool of Windows 365 Cloud PCs, not Entra-joined to your tenant, one per user at a time, can be throttled; prototyping only. **Bring your own**: provisioned in your virtual network, Entra ID support, Intune-enrolled, web and desktop; registered and managed in Power Automate; use for production. |
| Credentials to use | **Maker-provided** (default, suits autonomous agents; anyone using a shared agent acts with the maker's access on the machine) or **End user** (each user needs machine credentials). |
| Stored credentials | Sign-in credentials per website or desktop app, stored in Power Platform internal storage or in your Azure Key Vault (subscription needs the PowerPlatform resource provider registered). |
| Access control | Allow list of websites (wildcards supported) and desktop apps. It blocks actions on unlisted targets but doesn't stop the model opening them. |
| Enforce HTTPS | Only interact with sites whose URL starts with https:// |
| Human supervision | Email (Outlook) a reviewer when potentially harmful instructions are detected; the run stops if the request expires without a response. |

**Licensing**: computer use bills as the Agent action feature. Each step (which may contain several clicks or keystrokes) consumes 5 Copilot Credits, or 15 with a premium model. A long, poorly specified task therefore costs more; precise instructions (full URLs, exact app names, explicit submit actions, step lists) cut both errors and cost.

| Aspect | RPA (desktop flows) | Computer use (CUA) |
|---|---|---|
| **Automation type** | Rule based script | LLM driven |
| **Interaction** | UI tree and selectors | Vision |
| **Authoring** | Script, complex | Natural language instructions |
| **Error handling** | Static | Self-correcting from visual feedback |
| **Choose when** | Stable UI, clear rules, speed and volume matter, only GA features allowed, an RPA team owns it | UIs vary or change often, delivery is urgent, the task depends on what's visible (charts, colours, dynamic layouts), decisions are fuzzy |

**PRODUCTION NOTE**

Use dedicated, isolated machines; give the computer-use account least privilege; restrict web access to vetted domains (for example with Microsoft Edge policies via Intune); and restrict installed apps with application control. Computer use works best for autonomous agents running in the background. In conversational use, it shares reasoning messages and screenshots in the chat, and with user authentication each user needs valid credentials for the machine.

### 4.1.7 Agent behaviours: reasoning and voice

#### Model selection and deep reasoning

Copilot Studio lets you choose the primary model for an agent's orchestration. Models carry tags that describe their trade-off: **Deep** (deliberate, multistep, tool-rich reasoning; highest latency and cost; for complex analytics, policy and contract analysis, long-document synthesis), **Auto** (routes per turn across mixed workloads; variable latency and cost), and **General** (lowest latency and cost for drafting, summarising, FAQ-style grounded answers). Each model also has a release type: **experimental** and **preview** (not for production, subject to preview terms, may vary in latency or availability, still billed if published), **generally available**, **default** (also the fallback if a selected model is unavailable), and **retired** (usable for up to one month after retirement).

**Deep reasoning** PREVIEW is a separate capability for agents on the standard harness. When enabled, the agent can apply an advanced reasoning model (Learn names Azure OpenAI o3) to the tasks that benefit, or you force it for specific steps by including the keyword **reason** in those steps of the agent instructions.

1. Turn on generative orchestration for the agent.
2. In the agent's **Settings**, turn on **Deep reasoning (preview)**.
3. Add *reason* only to the instruction steps that need it (for example "use reason to provide one top supplier recommendation with context").
4. Test and expand the deep reasoning node in the activity map to inspect the reasoning steps, inputs, and output.

> **COMMON TRAP**
> Deep reasoning is slower and consumes billed Copilot Credits. Every step that should use it must contain the keyword, and putting it on many steps substantially slows responses. In preview it is supported only in the United States and the EU (excluding the United Kingdom) and makes no data residency commitments. A UK customer with strict residency requirements shouldn't be offered it as the solution.

#### Voice-enabled agents

Copilot Studio agents can serve telephony and interactive voice response (IVR) scenarios, typically integrated with Dynamics 365 Contact Center. There are two voice designs.

| Aspect | Basic voice | Real-time voice |
|---|---|---|
| **Orchestration** | Topics and classic orchestration | Generative orchestration |
| **Interaction style** | Structured: speech or DTMF input, menus, SSML-formatted output, barge-in, silence detection | Natural, context-aware conversation without traditional IVR menus; DTMF, barge-in, voice activity detection; multilingual |
| **Models** | Platform speech recognition and synthesis | GPT-Realtime, GPT-Realtime-Mini PREVIEW, or a text LLM voice model using GPT-5-Chat PREVIEW with Neural TTS (custom voices) |
| **Status** | Generally available | Listed as preview in What's new (April 2026); explicitly preview for digital messaging channels |
| **Choose when** | Compliance-heavy, deterministic menus and routing | Conversational self-service where callers describe needs in their own words |

To enable voice: **Settings > Voice > Enable voice**, then select Basic or Real-time. A voice-enabled agent uses the **Speech & DTMF** modality, which is mutually exclusive with the Text modality. Basic voice adds the system topics *Silence detection*, *Speech unrecognized*, and *Unknown dial pad press* and turns on *Increase accuracy with agent data*. **Enable voice changes authoring only, not the channel**: you must also turn on the **Telephony** channel and connect the agent to Dynamics 365 Contact Center (which needs an Azure Communication Services resource connected to the instance). Hold and resume lets a caller pause mid-conversation and continue later.

**COMMON TRAP**

Real-time voice models have regional constraints. GPT-Realtime is hosted in North America and GPT-Realtime-Mini in Australia, both requiring cross-geo processing elsewhere. EU Data Boundary and UK customers can use GPT-5-Chat with in-region processing but can't use the realtime models without cross-geo processing. Also note that data policies aren't supported in the enhanced voice experience in Customer Service, and real-time digital messaging agents in preview don't support OAuth or Microsoft authentication.

### 4.1.8 Optimising solution design with agents in Microsoft 365, Teams, and SharePoint

Many requirements are best met where users already work. An architect should exhaust the Microsoft 365 surfaces before building a standalone portal.

#### SharePoint agents

- Every site has a ready-made agent scoped to its content; users with access to SharePoint can create agents scoped to specific sites, folders, or files. Agents are stored as **.agent files** and don't automatically appear in any list; users open them like Word or Excel files.
- Site owners can choose one **main agent** for the site (opened from the Agent icon in the global header) and mark agents as **Approved** so they appear in a dedicated section of the agent picker for all site users.
- Licensing: users need a Microsoft 365 Copilot licence, or the tenant must have pay-as-you-go enabled for SharePoint.
- Sharing follows existing permissions. SharePoint agents can be used in SharePoint and Teams chats, can only be shared in Teams, and can't be published to the organisation catalogue.
- Governance: **restricted content discovery** on a site removes the Agent icon, blocks the ready-made agent and new agents on that site, and prevents its content from being added to other agents, while leaving site access unchanged. Admins can also block agents for Copilot Chat in the Agents section of the Microsoft 365 admin center (currently this affects Copilot Chat only, not OneDrive, SharePoint, or Teams).

#### Teams agents

- **Facilitator** is a collaboration agent in meetings that everyone can interact with like a team member (also available for in-person meetings through Teams Rooms).
- **Channel Agent** helps manage channel conversations: flags deadlines buried in threads, summarises progress, assigns tasks and due dates, and answers natural-language questions.
- Copilot Studio agents on the **Teams and Microsoft 365 Copilot** channel can be shown to shared users in **Built with Power Platform** or, after admin approval, in **Built for your org**; admins can auto-install and pin them with Teams app setup policies (Power Platform apps must be allowed in Teams).
- Group productivity (an agent participating in a channel or meeting with several users) is a custom engine agent scenario, not a declarative agent one.

| Requirement | Optimal Microsoft 365 design |
|---|---|
| **Project team wants Q&A over one site's documents, no IT build** | SharePoint ready-made or site agent, marked Approved by the site owner |
| **Business user wants a personal or team agent over SharePoint and web sources** | Agent Builder in Microsoft 365 Copilot (declarative) |
| **Department-wide agent with actions on line-of-business systems, admin-governed rollout** | Copilot Studio agent published to Teams and Microsoft 365 Copilot, approved in the admin center |
| **Meeting notes, action items, and follow-ups captured collaboratively** | Facilitator |
| **Status reports and deadline tracking from a busy channel** | Channel Agent |
| **Agent must post proactively into a Teams channel when an ERP event occurs** | Custom engine agent (Copilot Studio or Agents SDK) with autonomous trigger |
| **Sensitive site must never feed any agent but keep its permissions** | Restricted content discovery on the site |

## 4.2 Orchestrate configuration for prebuilt agents and apps

Prebuilt does not mean zero design. Each prebuilt agent or Copilot feature has prerequisites (environment integration, regions, data movement consent, licences, roles), admin switches in more than one admin centre, and extension points. The exam tests whether you know which switch lives where and in which order to configure them.

### 4.2.1 AI features in Dynamics 365 apps for finance and supply chain

Copilot in finance and operations apps (Finance, Supply Chain Management, Commerce, Project Operations, Human Resources) appears in three forms:

- **Sidecar**: the chat pane alongside the application; the primary Copilot interface. Generative help and guidance is delivered here.
- **Embedded**: AI built into pages, such as AI summaries in Supply Chain Management, the Confirmed purchase orders with changes workspace, customer page summary, and collections coordinator summary in Finance.
- **Outside**: external agents that orchestrate across apps, such as asking Microsoft 365 Copilot about finance and operations data.

| App | Copilot features | Agents |
|---|---|---|
| All finance and operations apps | Generative help and guidance, workflow history summary, enhanced user feedback, chat with finance and operations data PREVIEW, Immersive Home PREVIEW | Agent management PREVIEW; Dynamics 365 ERP MCP server for building agents |
| Finance | Collections coordinator summary, customer page summary | Account Reconciliation Agent PREVIEW (subledger to general ledger differences, proposes and applies resolutions) |

| App | Copilot features | Agents |
| --- | --- | --- |
| Supply Chain Management | AI summaries, analyse demand plans with Copilot, workload insights in the Warehouse Management mobile app, review changes to confirmed purchase orders | Procurement Agent with supplier communications and impact analysis features PREVIEW |
| Project Operations | Copilot in time entry, Copilot for project (WBS generation), Microsoft 365 Copilot chat PREVIEW, agent feed PREVIEW | Time Entry, Expense, and Approvals agents PREVIEW |

Learn labels several of these agents "production ready preview": preview terms apply, but Microsoft positions them for production use.

### Enablement prerequisites




- **Power Platform Integration** must be enabled in Lifecycle Services; dual-write isn't required unless you use it. Copilot and AI agents aren't supported in developer environments deployed through Lifecycle Services.
- Required Dynamics 365 apps in the linked Power Platform environment: **Copilot for finance and operations apps** (generative help and the sidecar bot), **Copilot in Dynamics 365 Supply Chain Management** (SCM and Commerce summaries), **Copilot in Dynamics 365 Finance** (Finance summaries). Install missing ones from the environment's Dynamics 365 apps page in the Power Platform admin center.
- The sidecar uses Copilot Studio generative answers, which require **Bing Search**; if the AI services aren't available in your Dataverse region, **cross-region data movement** must be allowed. If either is disabled, users can't open the sidecar or get no answers.
- The tenant setting **Publish bots with AI features** must stay enabled.

### Architecture and extension points

The sidecar is the same AI Copilot control used in canvas and model-driven apps, hosted in the **SysCopilotChatPanel** form. It sends prompts to Copilot Studio, which provides the orchestration. Copilot in finance and operations apps is bound to a single agent named **Copilot for finance and operations apps**, deployed in the **msdyn_FnoCopilot** solution. Generative orchestration is supported for this agent but **not enabled by default**; turn it on when your extensions need it. You extend it with tools:

- **AI tools**: headless operations that don't need the app UI (for example customer balance or on-hand inventory) and can be reused across Copilot interfaces, including Microsoft 365 Copilot. Business logic is exposed as X++ classes through the AI tool framework.
- **Client plugins**: skills that only make sense in the current in-app context or require a client action (for example alternate production plans on the production schedule page).

### Dynamics 365 ERP MCP server

The Dynamics 365 ERP MCP server lets any compatible agent platform (Copilot Studio, Microsoft Foundry, or non-Microsoft clients) work with finance and operations data and business logic without custom connectors or APIs. It is **dynamic**: the context returned at each call reflects the agent identity's security roles, configuration, extensions, and personalisation.

| Tool category | Purpose | Examples |
|---|---|---|
| Data tools | CRUD through OData data entities; most efficient for standard operations | `data_find_entity_type`, `data_get_entity_metadata`, `data_create_entities`, `data_find_entities_sql` |
| Form tools | Work through server forms as a user would (buttons, lookups, grids) for non-CRUD business logic; uses server APIs, not the client UI | `form_open_menu_item`, `form_set_control_values`, `form_click_control`, `form_save_form` |
| Action tools | Find and invoke X++ classes exposed as AI tools (implementing `ICustomAPI`) | `api_find_actions`, `api_invoke_action` |

- **Prerequisites**: version 10.0.47, 10.0.46 PQU-2, or 10.0.45 PQU-7 or later; the feature enabled in Feature management (on by default); the agent platform added on the **Allowed MCP Clients** page; a **Tier 2 or higher** environment or a Unified Developer Environment (not cloud-hosted environments).
- The earlier **static** ERP MCP server (13 fixed tools) is retired on **1 October 2026**; design new agents on the dynamic server.
- **Licensing**: in Copilot Studio, tool calls bill as Agent actions (orchestration and execution in one fixed rate). In other clients, the server charges 0.1 Copilot Credits per tool call on top of the client's own token costs; Finance Premium and Supply Chain Management Premium users are exempt from that execution charge for non-Copilot Studio clients. Agent identities assigned the **System agent** security role don't need a finance and operations user licence; assign other roles for actual permissions.
- **Limitations** include US English metadata, excluded security and admin forms, and no availability during servicing. Adding the ERP MCP server to the **Copilot for finance and operations apps** sidecar agent isn't yet supported.

**EXAM TIP**

"Build an autonomous agent that creates purchase orders in Supply Chain Management from Copilot Studio with no custom code" points to the ERP MCP server with an agent identity holding the System agent role plus a purchasing role. "Add a capability to the in-app sidecar" points to tools (AI tools or client plugins) on the Copilot for finance and operations apps agent, not to MCP.

## 4.2.2 AI features in Dynamics 365 apps for customer experience and service

### Dynamics 365 Sales

Copilot capabilities include record summarisation (leads, opportunities, accounts), embedded summary widgets, recent changes to records (requires audit history), information assistance from SharePoint documents, meeting preparation, email assistance, and news updates. AI agents include the **Sales Qualification Agent** (researches leads, assesses fit, sends outreach), **Sales Opportunity Agent** (researches

opportunities, surfaces risks), **Sales Close Agent** (engages customers through to closure), **Sales Research Agent** (natural-language dialogue over sales data for complex business questions), **Recommended Actions Agent**, AI-powered data enrichment, and the **Sales agent in Microsoft 365 Copilot** (section 4.2.4).

1. Confirm the region supports Copilot in Dynamics 365 Sales; if there's no Azure OpenAI endpoint in your region, consent to cross-region data movement in the Power Platform admin center (Learn recommends enabling it as a fallback even where an endpoint exists).
2. Ensure data policies allow the connectors Copilot requires.
3. In the Power Platform admin center, go to **Copilot > Settings** and, under Dynamics 365 Sales, turn on **Copilot** and/or **AI Agents** for environments or environment groups (on by default; turning off disables the features).
4. In Sales Hub, **App Settings > General Settings > Copilot**: turn audit on for lead, opportunity, and account if needed, and set Copilot on for all sales apps or per app.
5. Configure summary fields and Entra groups for access control.

### Dynamics 365 Customer Service and Contact Center

Copilot features include Ask a question, auto prompts, proactive prompts, conversation and case summaries, email drafting, chat with case data in Microsoft 365 Copilot, resolution notes, knowledge drafts from cases, and a Copilot analytics report. AI agents include the **Case Management Agent** (create, update, resolve, and close cases, follow-ups), **Customer Knowledge Management Agent** (creates and maintains knowledge from closed cases), **Quality Evaluation Agent**, **Customer Intent Agent** (and a voice variant) with intent-driven routing, a **rollout manager** for phased adoption of autonomous agents, and an agent insights dashboard.

Copilot features are managed in two places: the **Power Platform admin center** (turn on AI features for the environment, data movement, and data sharing) and the **Copilot Service admin center** (older content says Customer Service admin center), where you opt in, choose the Copilot experiences representatives use, configure Copilot agents, extensibility, translation, and transcript settings. Summarise conversations works in the Copilot Service workspace only; using Copilot in Customer Service Hub or custom apps needs the Power Platform admin center step as well. Copilot in Customer Service can also be extended with Copilot Studio PREVIEW.

### Dynamics 365 Field Service

Copilot answers natural-language questions about Field Service data, updates work orders from text or speech in the mobile app PREVIEW, summarises work orders, converts PDFs or images into draft inspection templates, and provides form-fill assistance, row summaries, and view visualisation. The **Scheduling Operations Agent** PREVIEW optimises resource schedules interactively on the schedule board or in batches, with custom optimisation goals. Agent feed lets users supervise agent-generated tasks.

> **NOTE**
> Customer Insights - Journeys introduced a **Dynamics 365 AI Hub** PREVIEW (public preview from May 2026 in the release plan) as a single in-product place to enable agents and Copilot features and configure guardrails, aligned with tenant-level settings in the Power Platform admin center Copilot hub. Don't confuse it with the Power Platform **AI hub** in Power Apps and Power Automate (section 4.2.5).

### 4.2.3 Proposing Microsoft 365 agents for business scenarios

Before building, check whether a Microsoft-built or partner agent already fits. Learn groups deployment into Microsoft-installed agents (currently only **Researcher** and **Analyst** are preinstalled or pre-pinned for licensed users), admin-installed agents, and user-installed agents from the Agent Store.

| Business scenario | Propose | Why |
|---|---|---|
| **Multi-step research combining internal files, emails, meetings, and the web into a cited report (market entry, competitor analysis)** | Researcher | Deep-research reasoning model over work data and the web; spends longer to produce sourced analysis |
| **Analyse raw data across several spreadsheets, run code, produce insights and visualisations** | Analyst | Reasons like a data analyst and runs code |
| **Improve writing, brainstorm, or learn better prompting** | Writing Coach, Idea Coach, Prompt Coach (also Career Coach, Learning Coach, Visual Creator) | Included Microsoft agents for everyday productivity |
| **Capture meeting decisions and action items collaboratively** | Facilitator (Teams) | Collaborative meeting agent |
| **Track deadlines and status in a Teams channel** | Channel Agent (Teams) | Collaborative channel agent |
| **Sellers need CRM insights in Outlook, Teams, and Copilot Chat** | Sales agent | Connects Dynamics 365 Sales or Salesforce to Microsoft 365 (section 4.2.4) |
| **Service representatives need case data and summaries in Microsoft 365** | Service Agent / Service app | Integrates Microsoft 365 Copilot with Customer Service (section 4.2.4) |
| **Employees need HR and IT answers and actions (Workday, ServiceNow HRSD and ITSM)** | Employee Self-Service agent (HR and IT starters) | Prebuilt declarative agent package deployed through Copilot Studio, with knowledge from SharePoint and ServiceNow knowledge, preconfigured Workday and ServiceNow scenarios |
| **Team Q&A over one SharePoint site** | SharePoint agent | No build, permissions inherited |

Admins browse the catalogue in the Microsoft 365 admin center under **Agents > All agents**, review Overview, Users, Data & tools (capabilities, knowledge, actions), Security & compliance, Certification, and Activity, then deploy to **Just me**, **Entire organization**, or **Specific users/groups**, or block or remove. Deployed agents appear in the Agent Store across Teams, Outlook, Word, Excel, and PowerPoint.

**COMMON TRAP**

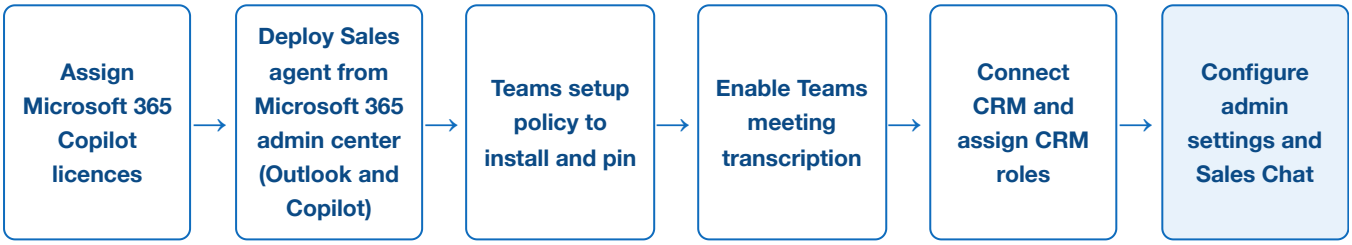
Researcher and Analyst are part of the core Copilot Chat experience and don't fall under agent-related settings in the admin center. Disabling Copilot extensibility affects Microsoft-deployed agents as well as others. If a question asks how to stop all users seeing a Microsoft-installed agent, the answer is to block it tenant-wide; per-user assignment controls are greyed out for these agents.

### 4.2.4 Configuring Microsoft 365 Copilot for Sales and Microsoft 365 Copilot for Service

> **NOTE**
> Naming has moved on since the outline was written. Microsoft 365 Copilot for Sales is now documented as the **Sales agent** (the Sales app in Outlook and Teams and "Sales in Microsoft 365 Copilot"). The service equivalent is documented as **Service Agent**, delivered through the **Service app** ("Service in Microsoft 365 Copilot"), while the Learn training module still uses "Microsoft 365 Copilot for Service". Some Learn articles also now say "Microsoft Copilot" where earlier content said "Microsoft 365 Copilot". Recognise all of these names.

#### Sales agent (formerly Microsoft 365 Copilot for Sales)

The Sales agent brings CRM context from **Dynamics 365 Sales** or **Salesforce** into Outlook, Teams, and Microsoft 365 Copilot: email summaries and drafts, opportunity summaries, meeting preparation cards, sales insights in Teams meeting recaps, saving emails and meetings to CRM, and conversational queries over CRM data in Copilot Chat. It isn't available for Dynamics 365 Customer Engagement (on-premises) or in GCC, and data sharing with a CRM starts only after admins and users consent to connect accounts.




- **Install** as an integrated app from the Microsoft 365 admin center (tenant administrator); the add-in can take up to 24 hours to appear. A Teams administrator creates a **setup policy** to install and pin it in Teams.
- **Roles** for Dynamics 365 Sales: admins need System Administrator or System Customizer to customise; users need Sales Manager or Salesperson (custom roles need extra privileges). For Salesforce, admins assign the security role in the **msdyn_viva** environment in the Power Platform admin center. Permission changes in CRM can take up to 15 minutes to reflect in Teams.
- **Admin settings** are opened from the Sales app in Outlook or Teams (Settings tab). Tenant-level: Copilot AI (who can use AI capabilities), collaboration spaces. Environment-level: Copilot AI, access settings (features on or off), Save to CRM fields, and forms and fields.
- **Sales Chat** (the agent in Microsoft 365 Copilot): requires Sales Chat turned on in Access settings and a CRM connection. You must **open the Sales Chat settings page at least once** to trigger initialisation and load supported entities. For Dynamics 365, the agent uses tables with synonyms configured in Copilot Studio for Copilot in Dynamics 365, otherwise a default list (lead, opportunity, account, contact, quote, sales order, invoice, and others); for Salesforce, the objects configured in Forms settings. Admins can add or remove CRM entities.
- **Access across apps**: tenant-level access settings control availability separately for Microsoft 365 Copilot, the Sales pane in Outlook, Teams, and Dynamics 365 (Global Administrator, Exchange Administrator, or Power Platform Administrator). These settings don't install the app, assign licences, or grant CRM data access.
- Sellers can also open the Sales agent inside Dynamics 365 Sales PREVIEW once Microsoft 365 Copilot is enabled in model-driven apps.

#### Copilot for Service and Service Agent

Copilot for Service brings generative answers over trusted knowledge and case context to representatives in Outlook, Teams, and an **embeddable widget** in non-Microsoft CRM desktops such as Salesforce and ServiceNow (the 2024 release plan also lists Genesys and Zendesk integrations). The configuration has a Microsoft 365 side and a CRM side.

1. **Licences and environment**: assign Microsoft 365 Copilot for Service licences; for non-Microsoft CRMs create a Power Platform environment with Dynamics 365 apps enabled and **Copilot for Service** selected for automatic deployment.
2. **Outlook**: deploy the Copilot for Service and Copilot for Service for Microsoft Outlook add-ins from the Microsoft 365 admin center (Fixed mode by default; can take up to six hours to appear).
3. **Teams**: create app **setup policies** in the Teams admin center to install and pin the app for representatives' groups, and turn on **transcription** in the Global (Org-wide default) meeting policy.
4. **Dynamics 365 Customer Service**: configure **server-side synchronisation** with Exchange Online so emails and appointments can be saved to the CRM.
5. **Salesforce**: Copilot for Service uses the Power Platform Salesforce connector; allow it in any data policy covering the **msdyn_viva** environment (created when the first user connects), and in Salesforce ensure the **Microsoft Power Platform** connected app is installed with permitted users, relaxed IP restrictions, refresh tokens valid until revoked, and no session timeout.
6. **Embedded widget**: in the admin center app, select **Set up Copilot embeddable widget** and embed it into the CRM agent desktop; assign the Customer Service representative role to users.

For Dynamics 365 Customer Service customers, the current model is **Service Agent in Microsoft Copilot**. A Microsoft 365 administrator manages installation and assignment of the **Service app** (installed automatically when a Customer Service licence is assigned in the tenant); separately, a Dynamics 365 administrator adds Microsoft Copilot to the Customer Service model-driven app for the in-app experience. Installing the app doesn't configure the environment or grant record access; tools available to representatives depend on Service Agent configuration and Dataverse privileges. Service Agent can be extended with Copilot Studio agents and MCP tools where supported.

> **COMMON TRAP**
> Installing the Sales or Service app as an integrated app activates it in Teams but doesn't install it for users. If representatives can't see it in Teams, the missing step is a Teams app setup policy. If meeting recaps lack sales or service insights, check that Teams transcription is on.

### 4.2.5 Power Platform AI features, including AI hub

**AI hub** is the entry point in Power Apps and Power Automate (left pane, **More > AI hub**) for discovering and managing AI capabilities: **AI models** (AI Builder), **prompts**, document automation, and **Monitor activity** (tables to monitor models and prompts, the data they process, and their consumption).

| Requirement | Propose | Notes |
|---|---|---|
| **Common task that is the same across businesses, such as scanning receipts for expense reports** | AI Builder **prebuilt** model | No training; straight to use in flows and apps |

| Requirement | Propose | Notes |
|---|---|---|
| **Extract fields from your own document layouts** | AI Builder **document processing** custom model, or a prompt with document input | Custom model trains on as few as five documents; prompt-based extraction with GPT models avoids labelling and training |
| **Predict an outcome from historical Dataverse data, detect your products in images** | AI Builder **custom** prediction or object detection model | Data unique to your business |
| **Summarise, classify, extract, translate, assess sentiment, draft replies** | **Prebuilt prompts** (also Power Fx `AIClassify`, `AIExtract`, `AIReply`, `AISentiment`, `AISummarize`, `AITranslate`) | Usable in Dataverse low-code plug-ins, Power Apps, and Power Automate |
| **Bespoke generation with your instructions, grounding, or a Foundry model** | **Custom prompt** | Shared prompt library across Copilot Studio, Power Apps, Power Automate |
| **End-to-end document intake with human review queue** | **Document automation toolkit** | Power Automate orchestrates, AI Builder extracts, Power Apps for review, Dataverse stores the queue |

**Governance and licensing**: admins enable or disable **AI prompts** per environment (Power Platform admin center, environment **Settings > Product > Features**) or through an **environment group rule**; once a group rule is published it locks the setting for environment admins. Copilot Credits are mandatory for AI tools and AI Builder capabilities used in Copilot Studio agents and agent flows; in Power Apps and Power Automate, Copilot Credits can be used alongside AI Builder credits or alone. Turning AI prompts off also removes prebuilt prompts.

**EXAM TIP**

"Monitor how many AI Builder and prompt runs each flow consumes" points to AI hub > Monitor activity. "Stop environment admins re-enabling AI prompts in a set of environments" points to an environment group rule, not an environment setting.

### 4.2.6 Interoperability of finance and operations agent chats with additional knowledge sources

The sidecar chat in finance and operations apps is a Copilot Studio agent, so its knowledge model is Copilot Studio's. Two distinct goals are often confused: **help and guidance** (how to use the product, covered in 4.2.7) and **chat with finance and operations data** PREVIEW (questions answered from structured ERP data the user can access).

For chat with data, add finance and operations structured data as a **knowledge source** in Copilot Studio:

- **Virtual entities** for finance and operations apps added as Dataverse knowledge, querying the ERP data in place.
- **Native Dataverse tables** populated from finance and operations apps with a synchronisation utility such as **dual-write**, added as Dataverse knowledge.

The same knowledge can be attached to several agent experiences: the **Copilot for finance and operations apps** agent (users ask about data in context in the sidecar), **custom agents** in Copilot Studio, and a **declarative agent** published to Microsoft 365 Copilot so users ask ERP questions from Copilot Chat. Answers respect what the user can access.

| Need | Pattern |
|---|---|
| **Sidecar users ask "what is the on-hand quantity of item X?" in context** | Add virtual entities or Dataverse tables as knowledge to the Copilot for finance and operations apps agent (or build an AI tool) |
| **Employees ask about vendor balances from Microsoft 365 Copilot Chat** | Declarative agent (Copilot Studio agent for Microsoft Copilot) with the ERP data as knowledge |
| **Data must be joined with CRM data in Dataverse and reused by many apps** | Dual-write to native tables, then Dataverse knowledge |
| **Agent must also change ERP data or run business logic** | Dynamics 365 ERP MCP server tools (outside the sidecar) or AI tools / client plugins (inside the sidecar) |
| **Answers must include policy documents stored in SharePoint** | Add SharePoint knowledge to a custom agent, or add your own topic to the Copilot for finance and operations apps agent |

**COMMON TRAP**

Learn's generative help article states that Dataverse virtual entities published from finance and operations apps aren't currently supported as knowledge sources for **generative help and guidance**. Virtual entities are a knowledge pattern for **chat with finance and operations data**, not for product help.

### 4.2.7 Adding knowledge sources to in-app help and guidance (Finance and Supply Chain Management)

**Generative help and guidance with Copilot** is the sidecar experience that answers how-to questions. By default its knowledge is limited to Microsoft's published product documentation for finance and operations apps, filtered by each user's language preference. It runs through the out-of-box **conversational boosting** topic, which uses Copilot Studio generative answers over the agent's knowledge sources. It is usually available by default when Power Platform Integration is enabled.

**Process: add organisation-specific knowledge**




1. Open Copilot Studio and select the Dataverse environment associated with the finance and operations apps environment.
2. Under **Agents**, open **Copilot for finance and operations apps**.
3. On the **Knowledge** tab, select **Add knowledge** and use **Upload files** (PDF, RTF, Word, and other supported formats). Content limits depend on the knowledge source type.
4. Wait until the source shows **Ready**, then select **Test** and ask representative questions.
5. Select **Publish**. Users should close open Copilot sessions and start a new one to pick up the knowledge.

- **Other knowledge types** (for example SharePoint or other data sources) aren't handled by the out-of-box topic: you must add **your own topic** to the Copilot for finance and operations apps agent.
- The conversational boosting topic assumes all knowledge is about product usage. Adding unrelated content can make Copilot imply that it relates to the application, so keep help knowledge on-topic.
- **AI general knowledge** (model knowledge, Bing web content) is disabled by default; enable it on the agent's Overview page under Knowledge ("Allow the AI to use its own general knowledge"). Copilot uses it only after exhausting custom sources. Learn warns it increases the risk of incorrect content, so test before publishing to production.

### Classic Help pane versus generative help

| Aspect | Copilot generative help and guidance | Custom Help in the Help pane |
|---|---|---|
| **Experience** | Conversational answers in the sidecar | Context-sensitive articles and task guides in the Help pane |
| **How you add content** | Upload files (or add a topic for other sources) to the Copilot for finance and operations apps agent in Copilot Studio, then publish | Create HTML content, publish to a website (no sign-in required), index it with a search service (ConvertHtmlToJson to Azure Search), extend the Help pane in Visual Studio, assign an index per language |
| **Context** | Generative answers grounded on knowledge | Metadata: `title`, `description`, `ms.search.form` (AOT form name), `ms.locale`, `ms.search.scope` (Core, Operations, Retail, Human Resources) |
| **Skills needed** | Copilot Studio maker | Finance and operations developer tools and content authoring |

**EXAM TIP**

If a stem says "users ask Copilot how to post a vendor invoice under our company-specific procedure, documented in a Word file", the answer is to upload the file to the Copilot for finance and operations apps agent and publish. If it says "the procedure is on a SharePoint site", uploading isn't enough: add a custom topic (or SharePoint knowledge through your own topic). If it says "show our articles in the Help pane for the vendor invoice form", that is custom Help with `ms.search.form` metadata.

## 4.3 Additional practice questions

These are original items written against the official skills outline and current Microsoft Learn content. Answers and explanations follow the last question.

**Practice 4.1** MULTIPLE CHOICE

Contoso Logistics has six Copilot Studio agents that need the same set of warehouse tools (check stock, reserve stock, create transfer). The warehouse API changes several times per quarter, and the integration team wants to update tool definitions once without republishing every agent. The tools must remain subject to the company's Power Platform data policies.

You need to recommend an extensibility approach. What should you recommend?

A. Create a REST API tool from the OpenAPI file in each agent.
B. Host an MCP server that publishes the tools by using the Streamable transport, and connect it to each agent.
C. Build an A2A agent that wraps the API and connect it to each agent.
D. Create one agent flow per operation and add the flows to each agent.

**Practice 4.2** MULTIPLE CHOICE

Fabrikam must enter supplier invoices into a legacy Windows desktop application that has no API. The vendor redesigns screens frequently. The solution will run unattended in production, and security requires that the automation machine is joined to Fabrikam's Microsoft Entra ID tenant and managed by Intune.

You need to design the automation. What should you use?

A. Computer use in Copilot Studio on a Microsoft-hosted machine
B. Computer use in Copilot Studio on a bring-your-own machine registered in Power Automate
C. A Power Automate desktop flow that uses UI element selectors
D. An MCP server that exposes the desktop application's screens as resources

**Practice 4.3** MULTIPLE CHOICE

A bank wants an agent in Microsoft Teams that participates in a loan committee channel, posts a summary to the channel automatically when a credit bureau result arrives, and uses a model the bank fine-tuned in Microsoft Foundry.

You need to choose the agent type. Which type should you choose?

A. A declarative agent created with Agent Builder
B. A declarative agent with an API plugin action defined in the manifest
C. A custom engine agent
D. A SharePoint agent shared in Teams

**Practice 4.4** MULTIPLE RESPONSE

A procurement agent in Copilot Studio (standard harness, generative orchestration on) retrieves supplier data from several tools and then must produce one well-justified supplier recommendation. Response time for the other steps must stay as fast as possible. The tenant is in the United States.

You need to apply deep reasoning only where it is needed. Which two actions should you perform? Each correct answer presents part of the solution.

A. Turn on **Deep reasoning (preview)** in the agent's settings.
B. Change the agent's primary model to one tagged **Deep**.
C. Add the keyword *reason* to the instruction step that produces the recommendation.
D. Add the keyword *reason* to every instruction step so the model is used consistently.

**Practice 4.5** MULTIPLE CHOICE

A utility company in the EU Data Boundary is building a Copilot Studio voice agent for Dynamics 365 Contact Center. Callers must describe problems in their own words, and the agent must use generative orchestration. All AI processing must remain in region.

You need to select the voice design. What should you select?

A. Basic voice with classic orchestration and DTMF menus
B. Real-time voice with the GPT-Realtime model
C. Real-time voice with the GPT-Realtime-Mini model
D. Real-time voice with the GPT-5-Chat text LLM voice model

**Practice 4.6** MULTIPLE CHOICE

Users of Dynamics 365 Supply Chain Management ask the Copilot sidecar how to follow the company's internal returns procedure. The procedure is maintained as pages in a SharePoint site. You want generative help and guidance to answer from that content.

What should you do?

A. Upload the SharePoint pages as files on the Knowledge tab of the Copilot for finance and operations apps agent, and rely on the conversational boosting topic.
B. Add your own topic that uses the SharePoint content to the Copilot for finance and operations apps agent in Copilot Studio, then publish.
C. Add the SharePoint site to the Help pane by extending it in Visual Studio.
D. Enable AI general knowledge on the Copilot for finance and operations apps agent.

**Practice 4.7** MULTIPLE CHOICE

An insurer wants a Foundry model that always returns claim triage output in a strict JSON schema and follows the insurer's terminology. It has 2,000 labelled examples of inputs and desired outputs. Regulations require that training data and model weights never leave the Azure region of the Foundry resource.

You need to choose a customisation method and training type. What should you choose?

A. Reinforcement fine-tuning with Global training
B. Supervised fine-tuning with Standard training
C. Direct preference optimisation with Developer training
D. Supervised fine-tuning with Global training

**Practice 4.8** MULTIPLE CHOICE

Finance staff want to ask Microsoft 365 Copilot Chat questions such as "Which vendors in USMF have balances over 50,000?" using data from Dynamics 365 Finance. Answers must respect each user's access, and you must minimise custom code.

What should you recommend?

A. Build a declarative agent in Copilot Studio that uses finance and operations virtual entities as Dataverse knowledge, and publish it to Microsoft 365 Copilot.
B. Upload exported vendor balance spreadsheets to the Copilot for finance and operations apps agent.
C. Create a custom Help website and connect it to the Help pane.
D. Train an AI Builder prediction model on vendor balances.

**Practice 4.9** MULTIPLE RESPONSE

You are building a Microsoft Foundry agent that uses the dynamic Dynamics 365 ERP MCP server to create and update purchase orders in a Supply Chain Management environment running version 10.0.47.

Which two prerequisites must be met? Each correct answer presents part of the solution.

A. Add the agent platform on the Allowed MCP Clients page.
B. Enable dual-write between finance and operations apps and Dataverse.
C. Use a Tier 2 or higher environment or a Unified Developer Environment.
D. Add the ERP MCP server as a tool on the Copilot for finance and operations apps agent.

**Practice 4.10** MULTIPLE CHOICE

You deploy the Sales agent for a Dynamics 365 Sales organisation, connect it to the CRM, and turn on Sales Chat in Access settings. Sellers report that when they ask the Sales agent questions in Microsoft 365 Copilot, they are told chat isn't set up to connect to their CRM.

What should you do first?

A. Assign the Sales Manager role to all sellers.
B. Open the Sales Chat settings page in the Sales agent administrator settings to trigger initialisation.
C. Create a Teams app setup policy that pins the Sales app.
D. Enable Teams meeting transcription.

**Practice 4.11** MULTIPLE CHOICE

A company uses Copilot for Service with Salesforce. Representatives can view case summaries in Outlook, but saving emails to Salesforce fails. The Power Platform admin center shows a data policy that applies to all environments, including msdyn_viva.

You need to resolve the issue. What should you do?

A. Configure server-side synchronisation with Exchange Online.
B. Add the Salesforce connector to the allowed connectors in the data policy for the msdyn_viva environment.
C. Deploy the Outlook add-in in Fixed mode.
D. Embed the Copilot for Service widget in the Salesforce Service console.

**Practice 4.12** MULTIPLE CHOICE

A Power Platform administrator must ensure that AI prompts are turned off in twelve sandbox environments and that environment administrators can't turn them back on.

What should the administrator do?

A. Turn off AI prompts on each environment's Features settings page.
B. Create a data policy that blocks the AI Builder connector.
C. Put the environments in an environment group, turn off the AI prompts rule, and publish the rules.
D. Remove AI Builder credits from the tenant.

**Practice 4.13** MULTIPLE CHOICE

A finance business partner needs to combine three large Excel workbooks, run calculations across them, and produce charts explaining a margin decline. The organisation has Microsoft 365 Copilot licences and wants to avoid building anything.

Which Microsoft 365 agent should you propose?

A. Researcher
B. Analyst
C. Facilitator
D. Writing Coach

**Practice 4.14** MULTIPLE CHOICE

The HR leadership SharePoint site contains confidential material. Site members must keep their current access, but no ready-made or user-created agent may use the site's content, and its content must not surface in Microsoft 365 Copilot or organisation-wide search.

What should the SharePoint administrator configure?

A. Block the site's agents in the Agents section of the Microsoft 365 admin center.
B. Restricted content discovery on the site.
C. Remove the site owner's permission to approve agents.
D. A sensitivity label in each declarative agent manifest.

## Answers and explanations

**4.1: B.** MCP servers publish tool names, descriptions, and inputs centrally; Copilot Studio reflects changes dynamically, so you update once for all agents. MCP connectivity relies on Power Platform connectors, so data policies still apply. REST API tools are defined per agent; A2A is for delegating to agents with their own reasoning, not exposing tools; flows don't solve central definition.

**4.2: B.** No API and frequently changing screens favour computer use (vision-based, self-correcting) over RPA selectors. Production with Entra ID join and Intune enrolment requires a bring-your-own machine registered in Power Automate; hosted machines aren't Entra-joined to your tenant and are for prototyping.

**4.3: C.** Group participation in a channel, proactive posting without user input, and a custom fine-tuned model all require a custom engine agent. Declarative agents use Copilot's model, are user-initiated, and are designed for individual use.

**4.4: A and C.** Deep reasoning must be turned on in settings (with generative orchestration), and the *reason* keyword forces it only on the steps that contain it. Using it on every step slows the agent; changing the primary model affects all orchestration.

**4.5: D.** Generative orchestration requires real-time voice. EU Data Boundary customers can use GPT-5-Chat with in-region processing, while GPT-Realtime and GPT-Realtime-Mini require cross-geo processing. Basic voice uses classic orchestration. (Real-time voice is currently preview.)

**4.6: B.** The out-of-box conversational boosting topic supports uploaded files; other knowledge types such as SharePoint require you to add your own topic to the Copilot for finance and operations apps agent. The Help pane is a different experience, and general knowledge would add model and web content, not your procedure.

**4.7: B.** Labelled input-output pairs and a fixed format are the SFT use case. Standard training runs in the resource's region and guarantees data residency; Global and Developer copy data elsewhere or give no residency guarantee. RFT needs a grader and suits problems with many valid solutions.

**4.8: A.** Chat with finance and operations data uses virtual entities (or Dataverse tables) as knowledge; adding them to a declarative agent published to Microsoft 365 Copilot extends Copilot Chat with ERP data, respecting user access, with no custom code. Spreadsheet exports go stale and bypass security.

**4.9: A and C.** The platform must be listed on Allowed MCP Clients, and the environment must be Tier 2 or above or a UDE (cloud-hosted environments aren't supported). Dual-write isn't required, and adding the ERP MCP server to the sidecar agent isn't yet supported.

**4.10: B.** Sales Chat initialises only after an admin opens the Sales Chat settings page at least once; that loads supported entities and sets up the CRM connection for chat. Roles, Teams pinning, and transcription don't initialise chat.

**4.11: B.** Copilot for Service uses the Power Platform Salesforce connector to save activities to Salesforce. If a data policy covers the msdyn_viva environment, the Salesforce connector must be allowed. Server-side sync applies to Dynamics 365 Customer Service, not Salesforce.

**4.12: C.** A published environment group rule for AI prompts locks the setting at environment level so environment admins can't override it. Per-environment toggles can be changed back by environment admins.

**4.13: B.** Analyst reasons over raw data, runs code across multiple spreadsheets, and produces insights and visualisations. Researcher focuses on multi-step research reports from work content and the web.

**4.14: B.** Restricted content discovery leaves site access unchanged, removes agent features on the site (ready-made agent, creating agents, adding its content to other agents), and stops content surfacing in Copilot and organisation-wide search. Blocking in the admin center affects only Copilot Chat availability of specific agents.

### Key takeaways

- Place every requirement on a layer (experience, orchestrator, model, knowledge, actions) and change only that layer. The harness (GitHub Copilot, standard, Copilot chat) is an architectural choice that affects capability and billing.
- Fine-tune in Foundry for format, style, tool-use accuracy, or cost through distillation, never to keep knowledge current. SFT for demonstrations, DPO for preferences, RFT for gradable multi-solution tasks; Standard training for residency.
- Declarative agents inherit Copilot's orchestrator, model, and compliance; custom engine agents bring their own and unlock proactive, group, and external-channel scenarios at the cost of hosting and compliance responsibility.
- Copilot Studio tools: connector, custom connector, agent flow, prompt, REST API (preview), MCP, computer use. Keep to 25 to 30 tools per agent; split into connected agents when selection precision degrades.
- MCP needs generative orchestration and the Streamable transport, supports tools and resources, is governed by data policies, and can't be called directly from topics. Use it for central, multi-agent tool catalogues; use direct APIs for quick prototypes.
- Computer use is vision-based, costs 5 Copilot Credits per step (15 premium), and belongs on bring-your-own, Entra-joined machines in production; prefer RPA for stable, high-volume, GA-only scenarios.
- Deep reasoning (preview) needs the settings toggle plus the *reason* keyword per step. Voice needs the Speech & DTMF modality and the Telephony channel; real-time voice uses generative orchestration with region-dependent models.
- Finance and operations Copilot needs Power Platform Integration, the required Dynamics 365 apps, Bing Search, and possibly cross-region data movement. The sidecar is the Copilot Studio agent "Copilot for finance and operations apps"; external agents use the dynamic ERP MCP server.
- Generative help knowledge: upload files to the Copilot for finance and operations apps agent and publish; other sources need your own topic; general knowledge is off by default. Chat with ERP data uses virtual entities or Dataverse tables as knowledge.
- Sales agent and Service Agent/Copilot for Service need Microsoft 365 admin deployment, Teams setup policies, transcription, CRM roles, and CRM-side setup (Sales Chat initialisation, msdyn_viva data policies, Salesforce connected app, server-side sync).
- AI hub in Power Apps and Power Automate surfaces AI models, prompts, document automation, and activity monitoring; govern AI prompts per environment or with locked environment group rules.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Course AB-100T00: Architecting agentic AI business solutions
- Harnesses in Copilot Studio
- Fine-tuning in Microsoft Foundry
- Manage fine-tuning costs in Microsoft Foundry
- Bring your own model for your prompts
- Connect to a Microsoft Foundry agent (preview)
- Compare declarative and custom engine agents
- Custom engine agents for Microsoft 365 overview
- Declarative agent schema 1.8 for Microsoft 365 Copilot

- Microsoft 365 Agents Toolkit
- Extend Microsoft Copilot with the Copilot chat harness
- Connect and configure an agent for Teams and Microsoft 365
- Agents admin guide for Microsoft 365
- Manage agents in the Microsoft 365 admin center
- Add tools to custom agents
- Extend your agent with tools from a REST API (preview)
- Agent flows in Microsoft Copilot Studio FAQ
- Add other agents overview
- Connect an agent available over the Agent2Agent (A2A) protocol
- Connect clients to your agent with the MCP and A2A channels (preview)
- Extend your agent with Model Context Protocol
- Connect your agent to an existing MCP server
- Use agent tools to extend, automate, and enhance your agents
- Automate web and desktop apps with computer use
- Use deep reasoning models for complex tasks (preview)
- FAQ for deep reasoning
- Select a primary AI model for your agent
- Configure basic voice agents
- Real-time agents overview
- Integrate a voice-enabled agent with Dynamics 365 Contact Center
- Migrate workstreams to support voice-enabled Copilots
- What's new in Copilot Studio
- Manage access to agents in SharePoint
- Overview of AI in Microsoft Teams for IT admins
- Overview of Copilot capabilities in finance and operations apps
- Enable Copilot capabilities in finance and operations apps
- Architecture of Copilot in finance and operations apps
- Use Model Context Protocol for finance and operations apps
- Agents, Copilot, and AI capabilities in Dynamics 365 apps
- Turn on and set up Copilot in Dynamics 365 Sales
- Track, manage, and scale Copilot adoption in the Power Platform
- Manage Copilot features in Customer Service
- Get started with Copilot Service admin center
- Agent installation in Microsoft Copilot
- Set up Agent Store in Microsoft Copilot
- Get started with Researcher agent in Microsoft Copilot
- Pattern 1: Employee AI enablement
- Explore prebuilt Microsoft 365 Copilot agents (training)
- Introduction to Employee Self-Service declarative agents
- Welcome to Sales agent
- Set up Sales agent in Microsoft 365 Copilot
- Administrator settings for Sales agent
- Manage access to Sales agent across Microsoft apps
- Sales agent deployment guide for Salesforce CRM customers
- Deploy Copilot for Service in Outlook and Teams (training)
- Set up Copilot for Service (training)
- Enable Service Agent in Microsoft Copilot
- AI models and business scenarios
- Get started with prebuilt prompts
- Administer AI Builder
- Monitor AI Builder models and prompts activity

- Document automation toolkit
- Licensing and Copilot Credits (AI Builder)
- Chat with finance and operations data (preview)
- Enable generative help and guidance with Copilot
- Add knowledge to generative help and guidance with Copilot
- Custom Help overview
- Connect a custom Help website to the Help pane

DOMAIN 3 · PART A

# Monitor, tune, and test AI solutions

**Part of Domain 3: Deploy AI-powered business solutions (40–45%)**

Domain 3 is the heaviest part of AB-100, and this chapter covers its first two functional groups: **Analyze, monitor, and tune AI-powered business solutions** and **Manage the testing of AI-powered business solutions**. The exam profile explicitly expects you to have "experience in monitoring agent performance and interpreting telemetry data to help ensure reliability, optimize behavior, and drive continuous improvement". In practice, that means you must be able to choose the right observability surface for each type of agent (Copilot Studio, Microsoft 365 Copilot agents, Microsoft Foundry agents, Copilot features in Dynamics 365), read the metrics those surfaces produce, turn user feedback into a prioritised improvement backlog, and design a test strategy that combines deterministic tests for apps and processes with probabilistic, evaluator-based tests for generative components.

Study this chapter as an architect, not as a maker. Questions typically describe an organisation, a symptom, or a governance constraint and ask which tool, metric, threshold, or process you should recommend. Pay particular attention to: who owns each telemetry scope; which features are in preview; what the documented limits are (sampling, retention, test set sizes); and which testing tools Microsoft has deprecated recently, because an expert architect is expected to steer customers away from tools with a fixed end-of-support date.

| Official skill (as of October 14, 2026) | Covered in |
|---|---|
| **Analyze, monitor, and tune AI-powered business solutions** | |
| Recommend the process and tools required for monitoring agents | 5.1.1, 5.1.2 |
| Analyze backlog and user feedback of AI and agent usage | 5.1.3 |
| Apply AI-based tools to analyze and identify issues and perform tuning | 5.1.4 |
| Monitor agent performance and metrics | 5.1.2, 5.1.5 |
| Interpret telemetry data for performance and model tuning | 5.1.6 |
| **Manage the testing of AI-powered business solutions** | |
| Recommend the process and metrics to test agents | 5.2.1 |
| Create validation criteria of custom AI models | 5.2.2 |
| Validate effective Copilot prompt best practices | 5.2.3 |
| Design end-to-end test scenarios of AI solutions that use multiple Dynamics 365 apps | 5.2.4 |
| Build the strategy for creating test cases by using Copilot | 5.2.5 |

## 5.1 Analyze, monitor, and tune AI-powered business solutions

Monitoring an agentic solution is different from monitoring a classic application. Uptime and latency still matter, but most failures that users notice are *quality* failures: an answer that is not grounded, a tool that was not called, an escalation that happened too late, or slow drift after a knowledge source changed. Microsoft guidance for agent Centres of Excellence states it directly: observability for agents needs to go beyond uptime and latency to include accuracy, output quality, and drift, and "a dashboard nobody watches won't catch drift. An alert does." Your design must therefore combine three things: product analytics that business owners can read, engineering telemetry that operations teams can query and alert on, and evaluation that catches regressions before users do.

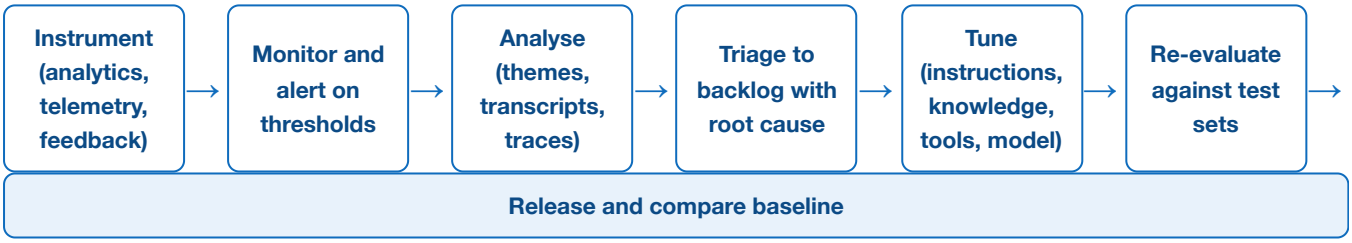
### 5.1.1 Recommend the process and tools for monitoring agents

Start every monitoring recommendation by asking two questions: **where is the agent built and run**, and **who needs the signal**. Microsoft explicitly notes that "an agent in Microsoft Copilot, an agent built in Copilot Studio, and a pro-code agent on Microsoft Foundry each expose different telemetry and dashboards". The table below maps the main monitoring surfaces to their audience and purpose.

| Monitoring need | Recommended tool | Primary audience | Notes for the architect |
|---|---|---|---|
| **Outcomes, satisfaction, answer quality, tool and knowledge use for one Copilot Studio agent** | Copilot Studio **Monitor** page (formerly the Analytics tab) | Agent owner, maker, business owner | Data for up to 360 days; session details and transcripts for the last 28 days; test panel activity is excluded. Share read-only with the *Analytics Viewer* sharing role. |
| **Message-level diagnostics, topic events, custom events for one agent** | **Agent-level telemetry** to Application Insights (`customEvents` table) | Maker or solution team | Configured per agent in Settings > Advanced; event based; not OpenTelemetry aligned. |
| **Centralised, cross-agent traces of agent invocations, tool execution, and outputs** | **Environment-level telemetry** to Application Insights **PREVIEW** (`dependencies` table) | Platform team, CoE, administrators | Configured once per managed environment from the Power Platform admin center; OpenTelemetry GenAI aligned spans. |
| **Operational health and adoption of agents across the tenant** | Power Platform admin center: **Monitor** > Copilot Studio, and the **Copilot** page | Power Platform administrators | Agent session success rate and recent agent sessions (both preview metrics); active agents in the last 28 days; message consumption. |
| **Adoption of agents and Copilot in Microsoft 365** | Microsoft 365 admin center usage reports (Microsoft Copilot usage, Agents usage); Copilot Dashboard and Copilot Analytics in Viva Insights | AI Administrator, business leadership | The newer Agents usage report shows usage within about an hour and covers declarative, SharePoint, and custom engine agents. |
| **Compliance and audit of Copilot and agent activity** | Microsoft Purview audit logs; Microsoft Sentinel for alerting | Security and compliance teams | Audit Reader role to search audit logs. |

| Monitoring need | Recommended tool | Primary audience | Notes for the architect |
|---|---|---|---|
| **Token usage, latency, run success, continuous evaluation for Foundry agents** | Foundry agent **Monitor** tab (Agent Monitoring Dashboard) backed by Application Insights | AI engineers, operators | Recurring evaluations, red team scans, and alerts are preview settings. |
| **Cross-platform agent view (Foundry, Copilot Studio, third party)** | Application Insights **Agent details** view and Agents (preview) blades | Operations | Based on OpenTelemetry Generative AI semantics. |
| **Use of Copilot by service representatives** | Copilot analytics report (Dynamics 365 Contact Center) | Supervisors, service managers | Usage, productivity (handle time, throughput with and without Copilot), and representative ratings. |
| **Fleet inventory, compliance checks, aggregated KPIs** | Copilot Agent Kit (formerly Copilot Studio Kit): Agent Inventory, Agent Insights Hub, Conversation KPIs, Compliance Hub | CoE | Open-source accelerator maintained by Power CAT; complements, not replaces, product analytics. |

The process that ties these tools together is a closed loop. Monitoring tells you something changed; analysis tells you why; tuning changes the agent; evaluation proves the change worked before it reaches users; and monitoring confirms the effect in production.




When you document the process for a customer, assign clear ownership at each stage. Microsoft's Copilot Studio governance guidance positions the built-in analytics for KPIs, Application Insights for custom events and topic telemetry, and Microsoft Sentinel for alerts and incident response. The CoE value guidance adds a cadence: weekly intake and operational review, monthly adoption metrics, quarterly cost, portfolio, and value reporting.

| Stage | Typical owner | Cadence | Artefact |
|---|---|---|---|
| Instrumentation standards (which telemetry scope, what is logged, redaction) | Platform team or CoE | Once per environment, reviewed per release | Telemetry standard, data policy settings |
| Operational monitoring and alerting | Operations or platform team | Continuous | Azure Monitor alerts, workbooks, on-call runbook |
| Quality and feedback review | Agent owner with business SMEs | Weekly | Prioritised backlog linked to transcripts and themes |
| Evaluation and regression | Agent owner or QA lead | Every change, plus scheduled runs against production | Test sets, pass rates, baseline history |

| Stage | Typical owner | Cadence | Artefact |
|---|---|---|---|
| Value and adoption reporting | CoE and business sponsor | Monthly and quarterly | Adoption, value, governance coverage, quality and risk measures |

**NOTE**

Access to analytics is itself a governance decision. An agent owner can grant access to only the Monitor page by sharing the agent with the **Analytics Viewer** sharing role (individuals only, not groups). Reading transcript content, comments, and drill-down lists additionally requires the **Bot Transcript Viewer** security role, and the environment settings that allow transcripts to be saved in Dataverse and viewed by owners must be on.

**COMMON TRAP**

Do not recommend the Copilot Studio Monitor page to measure behaviour during development. The Monitor page does not show activity from the Copilot Studio test panel. Application Insights, by contrast, *does* log test conversations, so queries must filter them out (see 5.1.6).

**PRODUCTION NOTE**

If an organisation wants only centralised, environment-level telemetry, administrators can stop makers from enabling agent-level telemetry by blocking the **Application Insights in Copilot Studio** connector in a data policy in the Power Platform admin center. Microsoft also advises against sending agent-level and environment-level telemetry to the same Application Insights instance.

### 5.1.2 Copilot Studio Monitor page: sessions, outcomes, and sections

The Monitor page is tailored to the agent type. Conversational agents get effectiveness and use metrics; agents with event triggers (autonomous agents) get run outcome and trigger metrics; if an agent has both kinds of session in the reporting period, a **hybrid view** shows them side by side. The page also includes a **Summary** area (an AI-generated summary of key insights), an **Overview** area with KPIs, a **Savings** area for time and cost savings, and a **Custom metrics** area PREVIEW. Time stamps are in UTC.

| | |
|---|---|
| **Conversation** | An ongoing interaction between a user and the agent on a channel. It times out after 30 minutes of inactivity (3 minutes after an End Conversation event on the Telephony channel) and can contain one or more analytics sessions. |
| **Analytics session (conversational)** | The unit that outcomes are measured against. A session starts *unengaged* and becomes *engaged* when the user triggers meaningful work: in classic orchestration, a custom, Escalate, Fallback, or Conversational boosting topic; in generative orchestration, a plan that includes a non-system topic, Escalate, Fallback, a knowledge source, or a tool. |
| **Analytics session (event triggered)** | Runs from when the agent receives a trigger payload through the actions it runs. If the trigger fails and no payload arrives, no session begins, so analytics count only successfully triggered runs. |
| **Active users (DAU, MAU)** | Unique users who interacted at least once in a UTC day or calendar month. Supported only when the agent requires authentication. |

An engaged conversational session ends in one of three outcome categories, and the outcome *reasons* are what make the metric actionable:

| Outcome | Reason | Meaning and architectural reading |
|---|---|---|
| Resolved | Resolved confirmed | End of Conversation topic triggered and the user confirmed success. Strongest evidence of resolution. |
| | Resolved implied | Classic: End of Conversation triggered and the user let it time out. Generative: the session timed out with no remaining active plan waiting for input. Treat as weaker evidence. |
| Escalated | System intended | Escalation by a business rule (for example, a transfer above a monetary threshold). Expected; not a defect. |
| | System unintended | Escalation because thresholds were exceeded, such as repeated failures. Usually means the user was stuck: investigate. |
| | User requested | The user explicitly asked for a person. Investigate patterns, but some demand is legitimate. |
| Abandoned | | An engaged session timed out after 30 minutes without reaching resolved or escalated. |

Makers can also set the outcome from a tool by using the `conversationOutcome` parameter in the tool code editor (for example, `ResolvedConfirmed` or `ResolvedImplied`). This matters for architects because, in generative orchestration, many tasks end inside a tool or flow rather than in the End of Conversation topic; without an explicit outcome, success may only ever be recorded as implied.

The remaining sections of the conversational Monitor page are:

- **Effectiveness**: conversation outcomes, **Reactions** (thumbs up and thumbs down with optional comments), **Customer satisfaction** (CSAT out of 5 from end-of-session surveys), **Sentiment** PREVIEW, and an **Agents** list for child and connected agents with calls, success rate, and status.
- **Use**: **Themes** PREVIEW, **Generated answer rate and quality**, **Tool use** (top five tools, usage trend, success percentage, filterable by main and child agent), and **Knowledge source use** (use trend and error rate per source type).

For autonomous agents, the four core areas are **Run outcomes** (with average duration for successful and failed runs), **Trigger use** (which triggers fire, the most common events, and which events produce the most failed runs), **Tool use**, and **Knowledge source use**. Microsoft notes that long run durations can delay the workflows the agent participates in and suggests more specific instructions at trigger or agent level to improve speed.

**EXAM TIP**

The topic-level Monitor panel (outcomes and CSAT per topic) is available only for agents in **classic** orchestration. For generative orchestration, Microsoft directs you to conversation outcomes and themes instead. If a question describes a generative agent and asks how to find which "topic" is failing, look for themes, answer quality reasons, or tool and knowledge source use rather than topic analytics.

### 5.1.3 Analyse backlog and user feedback of AI and agent usage

User feedback is the cheapest evaluation data you will ever get, but only if it is collected deliberately, stored where analysts can reach it, and turned into a backlog with owners. Design feedback collection across four layers.

| Feedback signal | Where it is captured | Design considerations |
|---|---|---|
| Reactions (thumbs up and thumbs down) and comments on individual agent responses | Copilot Studio; stored in the Dataverse `ConversationTranscript` table | On by default for all channels except the Microsoft Copilot channel. Configure under Settings > User feedback, including a disclaimer about how feedback is used. Viewing comments requires Bot Transcript Viewer. |
| Feedback from agents published to the Microsoft Copilot channel | Copilot Studio Monitor page | Requires admin consent in the Microsoft 365 admin center to share feedback. These reactions aren't counted in the Reactions column of the Themes and Knowledge source use sections. |
| CSAT survey at session end | Copilot Studio Customer satisfaction section | Scores 1–2 map to Dissatisfied, 3 to Neutral, 4–5 to Satisfied. Only sessions where users answered the survey contribute, so sample size matters. |
| Sentiment across a whole session | Copilot Studio Sentiment PREVIEW | AI analysis of a sample of sessions; the figure shown is the percentage of sessions with negative sentiment. Can be turned off in Settings > Advanced. |
| Customer comments summary | Monitor page Summary card PREVIEW | Groups related comments, filters low-value ones, and prioritises insights by volume and significance. |
| Representative ratings of Copilot responses | Copilot analytics report in Dynamics 365 Contact Center | Counts of positive and negative ratings, plus percentage of Copilot responses used (copied). Interaction data is stored in tables such as `msdyn_copilotinteraction` for custom reporting. |
| Employee sentiment about Copilot and agents | Viva Glint Copilot Impact Survey; Copilot Dashboard | Self-reported value complements telemetry, which only shows what the agent did. |

**Retention shapes your analysis window.** Copilot Studio session details and transcripts in the Monitor page cover the last 28 days, downloadable transcripts cover any period within the last 29 days, and in Dataverse a bulk-delete job removes agent transcripts older than 30 days by default. If the customer needs a longer feedback history (for example, to compare quarters), recommend changing the bulk-delete job and exporting to an analytics store, while checking that this meets privacy and retention policies.

Turning raw feedback into a backlog follows a repeatable pattern:

1. **Aggregate**: use themes to cluster questions that triggered generative answers, and the outcomes chart to find where sessions escalate or are abandoned.
2. **Drill down**: from a theme or a poor-quality bar segment, open the list of user questions; from there open the individual response, the reaction, the comment, and the reason for a Poor quality label.
3. **Classify the root cause**: knowledge gap (no source contains the answer), retrieval problem (source exists but wasn't used), instruction or tone problem, tool failure, orchestration routing problem, or a legitimate out-of-scope request.

4. **Prioritise**: weight by volume (total questions in the theme), impact (escalation or abandonment), and risk (compliance, safety). Microsoft's guidance on value reporting recommends a short, balanced set of measures with baselines and targets, which gives you the prioritisation lens.
5. **Convert to a test**: before fixing, turn the failing questions into an evaluation test set (themes and drill-down lists both offer an **Evaluate** action) so the fix is measurable and becomes a regression test.

| Pattern observed in feedback | Likely backlog item | Owner |
|---|---|---|
| **Theme with high question volume but low answered rate** | Knowledge coverage gap: add or restructure sources | Content owner |
| **High answered rate but many Poor quality labels and thumbs down** | Grounding or instruction issue; review retrieved sources and answer reasons | Agent builder |
| **Spike in System unintended escalations after a release** | Regression in a topic or tool; compare with previous baseline evaluation | Agent builder, QA |
| **Knowledge source with rising error rate** | Connection, permissions, or indexing issue | Platform team |
| **Negative comments about tone with correct facts** | Response style instructions | Agent builder |
| **Frequent requests outside the agent's scope** | Product decision: extend scope, add a child agent, or improve deflection messaging | Business owner |

**Themes** PREVIEW deserve particular attention because they bridge feedback and testing. Copilot Studio analyses unclassified questions that triggered generative answers in the past week and suggests up to 10 themes; it then classifies new questions daily and refreshes suggestions weekly. Each theme shows total questions, answered questions, response quality (on a sample), and reactions. Prerequisites matter for exam questions:

- At least 50 questions with generative answers from 50 different sessions in the past seven days before themes are suggested.
- The **Bot Transcript Viewer** security role to see the list and its metrics.
- Environment settings that store conversation transcripts in Dataverse (and, for drill-down, allow owners and editors to see session transcripts).
- Theme drill-down lists include the most recent 100 questions and only questions from the last 30 days.

> **COMMON TRAP**
> A theme can contain *unanswered* questions even though every question in it triggered generative answers. Triggering a generative answer call is not the same as producing an answer; for example, a conversational boosting topic can call a knowledge source that simply doesn't contain the answer. Low answered rate in a theme therefore points at content coverage, not at the orchestrator.

### 5.1.4 Apply AI-based tools to analyse and identify issues and perform tuning

The exam outline specifically asks you to apply *AI-based* tools to diagnose problems. Microsoft now provides AI assistance at every layer of the stack, and an architect should know which one fits which investigation.

| AI-based tool | What it does | Use it when |
|---|---|---|
| **Copilot Studio Summary card** | Uses Copilot to summarise key insights for the reporting period: engagement, sentiment and feedback, and trends since the previous period. | An owner needs a quick, plain-language read of what changed. |
| **Copilot Studio Answer quality** | An AI evaluation of a sample of answered questions for completeness, relevance, and groundedness; labels answers Good or Poor and assigns a reason to Poor answers. | Answer rate is acceptable but users are unhappy with answers. |
| **Copilot Studio Themes** PREVIEW | AI clustering of user questions into suggested categories with per-theme KPIs. | You need to know what users actually ask and which areas underperform. |
| **Copilot Studio Custom metrics** PREVIEW | Define up to three business metrics in natural language, each with up to 10 result categories; AI builds a prompt that classifies a sample of sessions. Preview against up to 20 sessions, inspect the reasoning and the generated prompt, then save. Initial calculation over the past seven days, then nightly. | Standard outcomes don't express the business result (for example, "order placed", "policy explained correctly"). |
| **Copilot Studio Sentiment** PREVIEW | AI sentiment for whole sessions from a sample. | CSAT response rates are too low to be representative. |
| **Copilot Agent Kit: Conversation Analyzer, Agent Debugger, Agent Review Tool, Prompt Advisor** | Analyse transcripts with custom prompts; step through how a recorded conversation produced a response; detect anti-patterns in agent configuration; score and improve prompts. | Deep, repeatable diagnostics across many agents, often driven by a CoE. |
| **Foundry cluster analysis** PREVIEW | Groups evaluation samples with similar problems using embedding-based clustering, generates a diagnostic description and recommendations per cluster and subcluster. | A Foundry evaluation run has many failures and you need to find the dominant failure modes. Download the result: it isn't stored. |
| **Azure Copilot Observability Agent** | Chat with logs, metrics, and traces in natural language; run a deep investigation across application, infrastructure, and platform signals and save findings as Azure Monitor issues; autonomous operations PREVIEW correlate alerts in the background. | An operational incident involves the agent and its dependencies (APIs, Azure resources). Billed by consumption in Azure Agent Credits. |
| **Microsoft Copilot in Azure with Log Analytics** | Generates example KQL queries from prompts. | The team needs help writing queries against Application Insights tables. |

**PRODUCTION NOTE**

The Observability Agent's accuracy depends on the telemetry it can see. Microsoft's best practices say to collect the required signals first, then add service and resource context, follow safe telemetry design rules, and instrument critical workflows. If correlation fields are missing, investigations become less accurate and less well scoped. This is another argument for OpenTelemetry-aligned, environment-level telemetry for Copilot Studio and tracing for Foundry agents.

**From diagnosis to tuning.** Copilot Studio guidance provides an evaluation-driven triage and remediation framework with four layers: interpret scores and assess readiness; triage failures; map failure patterns to remediation; and analyse patterns. Every failure is assigned to one of three root cause types, each with a different owner:

| Root cause type | Owner | Typical fix |
|---|---|---|
| Evaluation setup issue | Evaluation author | Correct the expected answer, test case, or grader; the agent may already be right. |
| Agent configuration issue | Agent builder | Change instructions, topics, knowledge, tools, or escalation settings. |
| Platform limitation | Platform team | Work around, escalate, adjust scope, or document as a known limitation. |

The triage questions are asked in order: is the agent's response actually acceptable (if yes, fix the evaluation); is the expected answer still correct (if no, fix the evaluation); can you identify a specific configuration to change (if yes, fix the agent); does the fix persist after the change (if no, it's a platform limit). After classifying failures, run a **concentration analysis**: if 80% or more failures share one root cause type, fix the category rather than individual cases. If 80% or more are evaluation setup problems, pause agent iteration and calibrate the evaluation suite first.

| Cross-signal pattern | Likely root cause | Tuning action |
|---|---|---|
| **Factual accuracy and knowledge grounding both failing** | Knowledge source wrong, missing, inaccessible, or stale | Check knowledge configuration, indexing status, and content freshness |
| **Tool invocation and trigger routing both failing** | Orchestration configuration: topics and tools not properly connected | Review how topics route to tools; check disconnected or misconfigured flows |
| **Tone failing, accuracy passing** | Delivery, not facts | Adjust style instructions only |
| **Safety passing, accuracy failing** | Over-constrained agent that refuses legitimate questions | Narrow overly broad safety instructions |
| **Accuracy improving while tone degrades** | Instruction conflict (the "instruction budget") | Review recent prompt changes and resolve conflicts |
| **Many evaluation sets degrading at once** | One broad root cause | Check recent system prompt changes, knowledge updates, or platform model updates |

Every remediation should follow the pattern Microsoft describes as *change X, rerun Y, expect Z*. That discipline prevents trial-and-error tuning and gives you evidence for release decisions.

**EXAM TIP**

When an answer option proposes "add more knowledge sources" or "remove knowledge sources" as a blanket fix, be careful. Microsoft explicitly states that reducing the number of knowledge sources isn't guaranteed to improve general quality grading. The correct answer usually diagnoses first (themes, answer quality reasons, knowledge source error rate, traces) and then applies a targeted change.

### 5.1.5 Monitor agent performance and metrics

A useful metric set for agents mixes **leading** signals that move early (active use, answer quality, review pass rate) with **lagging** signals that confirm results (cost avoided, cycle time, return on investment). Microsoft's CoE guidance groups measures into four categories: adoption and delivery; value; governance coverage; and quality and risk. Keep the set short, give each measure a baseline and a target, and review on a cadence.

| Metric | Definition (as documented) | Where | How to set a target |
|---|---|---|---|
| Resolution rate | Share of engaged sessions with outcome Resolved (confirmed or implied) | Copilot Studio Monitor | Baseline over several weeks; track confirmed and implied separately |
| Escalation rate by reason | Share of engaged sessions escalated, split into System intended, System unintended, User requested | Copilot Studio Monitor | Target reductions in System unintended only; System intended reflects policy |
| Abandonment rate | Engaged sessions that timed out after 30 minutes without resolution or escalation | Copilot Studio Monitor | Investigate rises after releases |
| Engagement | Proportion of sessions that become engaged | Copilot Studio Monitor | Low engagement can mean poor greeting, discoverability, or proactive messages users ignore |
| CSAT | Average 1–5 survey score; categories Dissatisfied, Neutral, Satisfied | Copilot Studio Monitor | Report with sample size |
| Answer rate | Answered versus unanswered questions for generative answers | Copilot Studio Monitor, Themes | Set per theme, not only globally |
| Answer quality | Share of sampled answers labelled Good for completeness, relevance, groundedness | Copilot Studio Monitor, Themes | Pair with evaluation thresholds (5.2.1) |
| Tool success rate | Percentage of called tools used successfully | Copilot Studio Monitor; ExecuteTool spans | High targets for transactional tools |
| Knowledge source error rate | Percentage of sessions using a source type that resulted in an error | Copilot Studio Monitor | Near zero; errors are usually configuration or permissions |
| Run outcomes and duration | Success or failure of event-triggered runs, average duration for successful and failed runs | Copilot Studio Monitor (autonomous) | Align to the SLA of the business process the agent serves |
| Agent session success rate PREVIEW | How often an autonomous agent executes its task successfully, or a conversational session is resolved | Power Platform admin center Monitor and Copilot page | Tenant view across agents; top 10 agents by monthly active sessions |

| Metric | Definition (as documented) | Where | How to set a target |
|---|---|---|---|
| Run success rate, latency, token usage | Percentage of successful runs; response time; tokens in period | Foundry Monitor tab | Microsoft's dashboard guidance: below 95% run success warrants investigation; latency above 10 seconds might indicate throttling, complex tool calls, or network issues; high token use might indicate verbose prompts or responses |
| Active users and agents | Unique users of agents (licensed and unlicensed), active agents, per user and per agent usage | Microsoft 365 admin center Agents usage report | Compare against assigned licences and target population |
| Copilot use by representatives | Daily active users, Copilot responses, percentage of responses used, handle time and throughput with and without Copilot | Contact Center Copilot analytics report | Compare with-Copilot and without-Copilot cohorts |

**NOTE**

Microsoft does not publish universal targets for resolution, escalation, or CSAT. Where documentation does give numbers (Foundry run success below 95% and latency above 10 seconds as investigation triggers; risk-based evaluation thresholds in 5.2.1), use them as starting anchors and calibrate to the customer's risk profile and baseline.

To measure business value credibly, Microsoft recommends capturing a **baseline before rollout** (cycle time, cost, error rate, hours spent), keeping a **comparison group** where possible, and combining usage telemetry with **self-reported time**, including where the reclaimed time goes. The Copilot Studio Monitor page's Savings area and the Viva Insights Copilot business impact report support this lagging view.

**EXAM TIP**

Match the report to the population. The Power Platform admin center Copilot Studio page doesn't include agents built with Agent Builder in Microsoft 365 Copilot. To see usage of declarative and SharePoint agents in Microsoft 365, use the Microsoft 365 admin center Agents usage report or Copilot Analytics in Viva Insights.

### 5.1.6 Interpret telemetry data for performance and model tuning

Copilot Studio supports two telemetry scopes in Application Insights. The choice determines the monitoring boundary, owner, data shape, and the queries you write.

| Aspect | Agent-level telemetry | Environment-level telemetry PREVIEW |
|---|---|---|
| **Configured** | Per agent: Settings > Advanced > Application Insights connection string | Once per managed environment: export package of type Copilot Studio in the Power Platform admin center |
| **Owner and privilege** | Maker or developer with access to the agent | Tenant-level Power Platform or Dynamics 365 administrator plus environment administrator |

| Aspect | Agent-level telemetry | Environment-level telemetry PREVIEW |
|---|---|---|
| **Harness support** | Standard harness only | Standard harness and GitHub Copilot harness |
| **Model** | Event based | Trace and span based, aligned with OpenTelemetry GenAI semantic conventions |
| **Main table** | `customEvents` | `dependencies` (spans `InvokeAgent`, `ExecuteTool`, `OutputMessages`) |
| **Best for** | Message activity, topic events (TopicStart, TopicEnd), custom events, single-agent troubleshooting | Agent invocations, tool execution, outputs, dependencies, cross-agent monitoring |
| **Notable limits** | Optional settings control what is logged: Enable logging, Log conversation details, Log sensitive Activity properties, Node execution events | Managed environments only; doesn't capture topic events; `duration` isn't available for standard harness traces; no unauthenticated or multi-tenant agents; delivery can take up to 24 hours on new configurations; large content may be truncated |

**Reading agent-level telemetry.** Events in `customEvents` include names such as `BotMessageReceived`, `BotMessageSend`, `TopicStart`, `TopicEnd`, and `GenerativeAnswers`, with activity data in `customDimensions` (for example `type`, `channelId`, `text`, `designMode`). Because the agent also logs conversations from the Copilot Studio test canvas, every production query should exclude `designMode` = True.

**Listing 5.1: Daily distinct users and sessions, excluding test canvas conversations (agent-level telemetry)**

```
customEvents
| where timestamp > ago(30d)
| where tostring(customDimensions['designMode']) == "False"
| summarize users = dcount(user_Id), sessions = dcount(session_Id)
    by bin(timestamp, 1d)
| render timechart
```

Distinct user counts are meaningful only when users are authenticated and their IDs are consistent across conversations; with anonymous channels a random user ID is generated per conversation.

**Listing 5.2: Generative answers whose result indicates content filtering, by day**

```
customEvents
| where name == "GenerativeAnswers"
| extend cd = todynamic(customDimensions)
| extend result = tostring(cd.Result),
         topic = tostring(cd.TopicName),
         conversationId = tostring(cd.conversationId)
| where result contains "Filtered"
| summarize filtered = count() by topic, bin(timestamp, 1d)
| order by timestamp desc
```

Microsoft's troubleshooting guidance uses this event to confirm whether generative answers were blocked by content moderation. A rising filtered count for one topic is a tuning signal: check whether the knowledge content, the agent instructions, or overly strict moderation settings are causing legitimate answers to be withheld.

**Reading environment-level telemetry.** Each agent turn is a trace with a shared `operation_Id`. The `InvokeAgent` span is the root; `ExecuteTool` and `OutputMessages` spans nest beneath it through `operation_ParentId`. Group by `gen_ai.conversation.id` (in `customDimensions`) to reassemble a multi-turn conversation. Useful keys include `gen_ai.agent.name`, `gen_ai.request.model`, `gen_ai.tool.name`, `gen_ai.tool.type`, `gen_ai.tool.call.arguments`, `gen_ai.tool.call.result`, and `microsoft.channel.name`; the native `success` and `resultCode` (`OK` or `ERROR`) columns give span status. Because the schema evolves, Microsoft recommends discovering keys live.

**Listing 5.3: Discover current customDimensions keys in exported Copilot Studio spans (from Microsoft documentation)**

```
dependencies
| where timestamp > ago(7d)
| mv-expand Key = bag_keys(customDimensions) to typeof(string)
| summarize Events = make_set(name),
            SampleValue = take_any(tostring(customDimensions[Key])) by Key
| order by Key asc
```

**Listing 5.4: Tool failure rate by agent and tool over seven days (environment-level telemetry)**

```
dependencies
| where timestamp > ago(7d)
| where name == "ExecuteTool"
| extend Agent = tostring(customDimensions["gen_ai.agent.name"]),
         Tool  = tostring(customDimensions["gen_ai.tool.name"]),
         Channel = tostring(customDimensions["microsoft.channel.name"])
| where Channel != "Copilot Studio Test Pane"
| summarize Calls = count(), Failures = countif(success == false) by Agent, Tool
| extend FailureRatePct = round(100.0 * Failures / Calls, 1)
| order by FailureRatePct desc
```

**Listing 5.5: Agent turns and errors by model and channel**

```
dependencies
| where timestamp > ago(7d)
| where name == "InvokeAgent"
| extend Agent = tostring(customDimensions["gen_ai.agent.name"]),
         Model = tostring(customDimensions["gen_ai.request.model"]),
         Channel = tostring(customDimensions["microsoft.channel.name"])
| summarize Turns = count(), Errors = countif(resultCode == "ERROR")
    by Agent, Model, Channel
| order by Errors desc
```

**COMMON TRAP**

Don't build latency dashboards for standard harness Copilot Studio agents on the `duration` column of environment-level spans: Microsoft documents that `duration` isn't available for those traces. Also, root `InvokeAgent` spans moved from `requests` to `dependencies` after the private preview, so older traces may still sit in `requests`.

Application Insights also provides prebuilt views so that not every stakeholder needs KQL: the **Copilot Studio Dashboard** workbook PREVIEW for agent-level telemetry (conversations, latency, exceptions, tool usage, topic analytics; editable and shareable with Reader access to the resource), and the **Agents (preview)** blades (Agent Runs, Tools, Models) that read the exported GenAI spans.

**Foundry agents.** Tracing in Microsoft Foundry is built on OpenTelemetry and stored in Application Insights. It captures user inputs and agent outputs, tool calls and results, token consumption, and duration and latency. Multi-agent conventions add spans such as `invoke_agent`, `plan`, `invoke_workflow`, `execute_tool`, and memory operations. Tracing is generally available for prompt and hosted agents; workflow and external agents are in preview. Best practices: use consistent span attributes, correlate evaluation run IDs with traces, and redact sensitive content before it reaches telemetry.

| Telemetry pattern | Interpretation | Performance or model tuning action |
|---|---|---|
| **One tool dominates ExecuteTool failures; result payloads show HTTP errors** | Back-end or connector issue, not reasoning | Fix the API, connection, or authentication; add retry or graceful failure messaging |
| **Tool is rarely invoked where evaluation expected it** | Orchestrator doesn't select the tool | Improve tool name and description, agent instructions; validate with the Tool use test method |
| **Token usage climbing with flat traffic** | Verbose prompts, large retrieved context, or long responses | Tighten instructions and output length; review knowledge chunking; consider a model with better cost profile |
| **Latency high only on turns with a specific tool** | Slow dependency | Optimise the API or move the step to an asynchronous flow; set user expectations |
| **Error spikes after a platform model update across many agents** | Shared root cause with broad impact | Rerun baseline evaluation sets, compare against previous baseline, adjust instructions or model selection |
| **Many GenerativeAnswers events filtered** | Content moderation blocking answers | Review source content and instructions; adjust moderation level only with responsible AI sign-off |

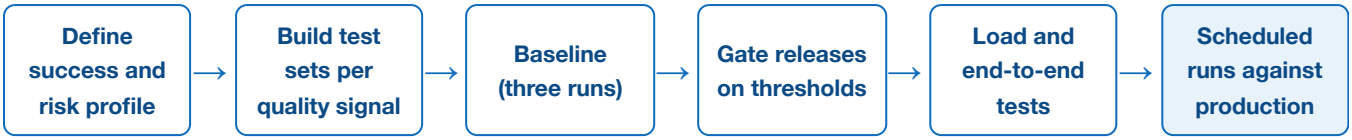
**PRODUCTION NOTE**

Telemetry is personal data. The agent-level setting **Log conversation details** includes user ID, user name, and message text, and with OpenTelemetry tracing it also controls whether tool input arguments and outputs are captured in spans. Environment-level spans can include `user.email`, prompts, and agent replies. Design Application Insights access (RBAC), retention, and region to meet data residency and privacy requirements, and treat trace data with the same controls as production logs.

## 5.2 Manage the testing of AI-powered business solutions

AI solutions need two kinds of testing. **Deterministic tests** check that apps, flows, integrations, and business processes behave the same way every time. **Evaluations** check the quality of non-deterministic outputs (answers, plans, extracted fields, classifications) against criteria and thresholds, using repeated runs and statistical judgement. Microsoft's guidance makes the distinction clear: agent evaluation "examines the quality of the agent's outputs", not just whether code runs. An expert test strategy uses both and wires both into release gates.

### 5.2.1 Recommend the process and metrics to test agents




**Step 1: define success.** Before running evaluations, define what success looks like and which scenarios matter most to business outcomes, and map business goals to measurable evaluation dimensions. Organise test sets by quality signal (for example safety and personal data, compliance wording, factual accuracy, knowledge grounding, tool invocation, trigger routing, escalation, tone) so that scores point to an area, not just a number.

**Step 2: choose the testing surface.** Copilot Studio offers two complementary in-product approaches, plus accelerators and pro-code options.

| Testing surface | Strengths | Limits to know |
|---|---|---|
| **Copilot Studio test chat** | Interactive, full multi-message sessions, behaves like a user | One question at a time; hard to repeat consistently |
| **Copilot Studio agent evaluation (single response test sets)** | Repeatable, many cases at once, aggregate scores, per-case transcript and activity map | Up to 100 test cases per set; results retained 89 days (export to CSV for longer) |
| **Copilot Studio agent evaluation (conversation test sets)** | Tests context retention, clarification, multi-step tasks | Up to 20 test cases, each up to 12 messages (6 question and answer pairs); the conversation test set article lists General quality, Keyword match, Tool use, and Custom as available methods |
| **Copilot Agent Kit test automation** | Batch tests through Direct Line; response match, attachments (Adaptive Cards), topic match, generative answers, multi-turn, and plan validation test types; rubrics; pipeline integration | Accelerator to install and operate; needs Dataverse, AI Builder, and Application Insights for some test types |
| **Foundry evaluations** | Built-in quality, RAG, safety, and agent evaluators; custom evaluators; AI red teaming agent; continuous and scheduled evaluation | Several evaluators and features are preview; consumption billing |
| **Load testing (for example Apache JMeter against the agent)** | Response time and error rate under realistic concurrency, multi-turn scripts | Excessive load can cause message overage and throttling; check throughput limits first |

Agent evaluation runs as the **currently signed-in user**; to test as a different persona, sign in as that user. Only agents set to **No authentication** or **Authenticate with Microsoft** are supported (not *Authenticate manually*), and Fabric data agents aren't supported. Evaluations can also be run through Power Platform REST APIs or connectors so they can be triggered from CI/CD pipelines.

**Step 3: choose test methods (metrics).** Each Copilot Studio test set can combine several methods:

| Test method | Measures | Scoring | Use when |
|---|---|---|---|
| General quality | LLM-judged relevance, groundedness, completeness, and abstention (whether the agent attempted to answer) | Out of 100% | No single correct wording exists; default method; no expected answer needed |
| Content safety | Hatefulness and unfairness, sexual content, violence, self-harm; severity levels 0 (strictest) to 7 | Pass/fail | Responsible AI gate; add one method per content type |
| Compare meaning | Semantic (intent) similarity to the expected answer | Out of 100%; default pass score 50 | Answers may be phrased many ways but meaning must match |
| Tool use | Whether expected tools or topics were used | Pass/fail | Validating orchestration and routing |
| Keyword match | Presence of any or all expected keywords or phrases | Pass/fail | Mandatory terms, disclaimers, product codes |
| Text similarity | Cosine similarity of wording and structure to the expected answer (0 to 1) | Pass threshold | Near-verbatim wording required (for example, legal text) |
| Exact match | Character-for-character match | Pass/fail | Short precise answers: numbers, codes, fixed phrases |
| Custom | Your own evaluation instructions and labels with pass/fail assignment | Pass/fail by label | Policy compliance, or treating an appropriate refusal as a pass |

**COMMON TRAP**

General quality penalises refusals because its abstention criterion checks whether the agent attempted to answer. If the correct behaviour for a test case is to decline (out of scope, legal advice, personal data request), use a **Custom** method with labels that define the expected refusal as a pass. Also note that every method except General quality needs expected answers, keywords, or tools; test cases without them return *Invalid* for that method.

**Step 4: set thresholds by risk.** Microsoft's triage guidance says not to apply the same thresholds to every agent. Consequence of failure, frequency of use, fallback availability, and audience all push thresholds up. The documented example starting points are:

| Risk profile | Safety and compliance | Core business | Capabilities |
|---|---|---|---|
| Low-risk internal tool | 90%+ | 75%+ | 65%+ |
| Medium-risk customer-facing agent | 95%+ | 85%+ | 75%+ |
| High-risk regulated or financial agent | 98%+ | 92%+ | 85%+ |
| Safety-critical agent | 99%+ | 95%+ | 90%+ |

Scores map to four **readiness states**: **Block** (safety, compliance, or critical business failures; the quick reference uses below 95% safety and compliance as a block), **Iterate** (for example core business below 80%), **Conditional deployment** (capability gaps documented and monitored), and **Deploy** (all thresholds met).

Conditional deployment with known gaps is a legitimate outcome if the gaps are logged with owners and timelines.

**Step 5: handle non-determinism.** Language-model agents and graders vary between runs. Microsoft's guidance:

- Run the full evaluation set at least **three times** before treating a score as a baseline, and use the average.
- Up to 5% variance between runs is normal for LLM graders; above 10%, investigate grader reliability before diagnosing the agent.
- With fewer than 30 test cases, one case flipping changes the score by 3% or more; with 50 or more cases, treat a change of 5% or more as meaningful.
- A test case that passes two out of three runs is flaky: decide whether the expectation is too rigid (evaluation setup) or the agent is genuinely inconsistent (configuration).
- Iteration isn't complete if thresholds were met only by removing difficult test cases, or if platform limitations were accepted without documentation.

**Step 6: performance and load.** A conversational agent performance test plan should state an objective, scope, KPIs (for example response time and error rate), scenarios, test data, tools, and success criteria. Microsoft's banking example sets baseline success as 95% of responses under 2 seconds with error rate below 0.5%, and load success as 90% under 3 seconds with error rate below 1%. Scripts must drive **multi-turn conversations**, because some back-end calls only occur after a particular sequence of choices. Design tests for baseline and anticipated peak, not extreme stress, to avoid overage and throttling.

**Step 7: monitor in production.** Evaluation doesn't stop at release. For Foundry agents, configure **continuous evaluation** (sampled live traffic, defined as an evaluation rule on response completion with a `max_hourly_runs` cap, default 100 per hour) and **scheduled evaluation** against test datasets to detect drift, plus scheduled red teaming and alerts. Continuous evaluation rules require the project's managed identity to hold the **Foundry User** role (formerly Azure AI User). For Copilot Studio agents, schedule evaluation runs through the API or connectors and compare against the stored baseline.

> **NOTE**
> Evaluation measures correctness and quality, not ethics. Copilot Studio's safety evaluators and Foundry's risk and safety evaluators help, but Microsoft is explicit that they don't guarantee an agent is safe for every scenario. Pair them with responsible AI reviews, content filters, and, for Foundry, the AI red teaming agent with human-in-the-loop review.

### 5.2.2 Create validation criteria for custom AI models

"Custom AI models" on this exam spans three families: **predictive and extraction models** (AI Builder prediction, category classification, document processing; Dynamics 365 predictive scoring; Finance insights), **custom language and vision models** in Foundry Tools, and **generative models and agents** (prompts, fine-tuned or selected models in Foundry). Validation criteria differ for each, so define them explicitly before training or deployment.

#### Classification and prediction metrics

Using a confusion matrix of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN):

| Metric | Formula or definition | Prioritise when |
|---|---|---|
| Accuracy | (TP + TN) ÷ total predictions | Classes are balanced and errors cost the same |
| Precision (positive predictive value) | TP ÷ (TP + FP) | False positives are expensive; Microsoft's example is ticket triage, where a wrong class sends work to the wrong team |
| Recall (sensitivity) | TP ÷ (TP + FN) | Missing a positive is expensive; Microsoft's example is flagging important email, where a miss matters more than a false alarm |
| F1 score | 2 × (Precision × Recall) ÷ (Precision + Recall), the harmonic mean | You need balance, especially with unbalanced data |
| AUC | Probability the model ranks a random positive above a random negative | Comparing scoring models across thresholds (used by Dynamics 365 Sales predictive scoring) |
| R-squared | Goodness of fit for numerical prediction, 0–100% | Numeric outcomes (AI Builder numerical prediction) |

**Worked example** (from the Dynamics 365 Sales predictive scoring documentation): 1,000 opportunities with TP 650, FP 200, TN 100, FN 50. Accuracy = 750 ÷ 1,000 = 75%. Recall = 650 ÷ 700 ≈ 92.9%. Precision = 650 ÷ 850 ≈ 76.5%. F1 ≈ 2 × 0.765 × 0.929 ÷ (0.765 + 0.929) ≈ 0.84. The model is good at finding winners but over-predicts wins; whether that is acceptable depends on what sellers do with the score.

### How AI Builder expresses model quality

- **Prediction models** show a performance grade A to D. Accuracy ranges for each grade depend on the historical outcome distribution (a score of 92 is a B when the historical on-time rate is 80% but an A when it is 50%). Grade D signals either an **underfit** model (worse than a random guess) or an **overfit** model at or near 100%, usually caused by a column directly correlated with the outcome. Numerical prediction grades use R-squared (A is 85% to below 99%; 99% or above is D).
- **Category classification** shows a performance score from 0 to 100 calculated from precision, recall, and F1.
- **Document processing** shows an accuracy score with recommendations, and returns a **confidence score** from 0 to 1 per extracted field and table cell at run time.
- All model types support evaluation on the details page, and many offer **Quick test** with your own data. AI Builder splits data into training and test sets and calculates scores on the held-out test data.

### Generative and agent evaluation metrics

| Category | Foundry built-in evaluators | Validation criterion example |
|---|---|---|
| General purpose | Coherence, Fluency | Average at or above an agreed score for customer-facing text |
| Textual similarity (needs ground truth) | Similarity, F1 score (token overlap), BLEU, GLEU, ROUGE, METEOR | Use for translation or summarisation against references |

| Category | Foundry built-in evaluators | Validation criterion example |
|---|---|---|
| RAG | Retrieval, Document Retrieval, Groundedness (1–5, model judged), Groundedness Pro PREVIEW (pass/fail via Azure AI Content Safety), Relevance, Response Completeness PREVIEW | No ungrounded claims in a regulated knowledge agent |
| Risk and safety | Hate and Unfairness, Sexual, Violence, Self-Harm, Protected Materials, Indirect Attack (XPIA), Code Vulnerability, Ungrounded Attributes, Prohibited Actions, Sensitive Data Leakage | Zero tolerance defects block release |
| Agent behaviour | Task Adherence, Task Completion, Intent Resolution, Customer Satisfaction (all preview), Task Navigation Efficiency, Tool Call Accuracy, Tool Selection, Tool Input Accuracy, Tool Output Utilization, Tool Call Success; batched Output Quality and Tool Use Quality (preview) | Correct tool chosen with correct parameters in a defined share of cases |
| Custom | Rubric evaluator PREVIEW (weighted criteria, 0–1), Azure OpenAI graders (Model Labeler, String Checker, Text Similarity, Model Scorer), custom evaluators PREVIEW | Domain rules such as "always cites policy number" |

Foundry evaluators run at **turn** level by default; conversation-level evaluation requires data in the `messages` format. Microsoft suggests combinations: for RAG apps, Retrieval + Groundedness + Relevance + Content Safety; for agents, Output Quality + Tool Use Quality + Rubric + Content Safety; and risk and safety evaluators for all applications.

| Validation criteria template element | Example for an invoice extraction model feeding a Finance agent |
|---|---|
| Business objective | Reduce manual keying of vendor invoices |
| Primary metric and threshold | Field-level accuracy on held-out invoices from the top 20 vendors at or above the agreed target |
| Error cost asymmetry | Wrong amount (FP) is worse than missing a field (FN); weight precision on amounts |
| Run-time guardrail | Route fields with confidence below a chosen threshold to human review |
| Representativeness | Test data covers layouts, languages, and scan qualities seen in production |
| Fairness and safety checks | Not applicable to extraction; required for any generated text shown to customers |
| Overfit check | Scores near 100% trigger a data leakage review |
| Retest triggers | New vendor layout, model retraining, or platform model update |

**EXAM TIP**

When a scenario stresses that "missing" a case is costly (fraud, safety incidents, VIP emails), choose **recall**. When it stresses that acting on a wrong prediction is costly (misrouting, wrong approvals), choose **precision**. When it stresses unbalanced classes, choose **F1** (or AUC for ranking) over raw accuracy. A near-perfect score is a warning sign of overfitting, not a reason to celebrate.

### 5.2.3 Validate effective Copilot prompt best practices

This skill has two audiences. End users prompting Microsoft 365 Copilot need guidance and adoption support; makers who embed prompts in agents, apps, and flows need prompts that are tested like code. As an architect you validate both.

**End-user prompting.** Microsoft's Microsoft 365 Copilot training identifies four elements of an effective prompt: **goal** (what you want), **context** (why you need it and how you will use it), **source** (which files, data, or organisational content to use), and **expectations** (format, audience, tone, length). The same elements appear in Security Copilot guidance. Microsoft's do's include being clear and specific, keeping it conversational, giving examples, asking for feedback, writing legibly, checking for accuracy, providing details, and iterating; the don'ts include being vague, giving conflicting instructions, using slang or jargon, requesting inappropriate content, and changing topics abruptly. The Copilot Prompt Gallery lets users discover, save, edit, and share prompts, which is a practical way to standardise validated prompts across a team.

| Prompt element | Validation check | Weak example | Validated example |
|---|---|---|---|
| Goal | Is a single, explicit task stated? | "Tell me about Project Atlas." | "Summarise the progress of Project Atlas" |
| Context | Does it say who it is for and why? | (missing) | "for the leadership steering meeting" |
| Source | Does it point to the right grounding and does the user have access? | (missing) | "using the latest emails, meetings, chats, and files for Project Atlas" |
| Expectations | Is the output format measurable? | (missing) | "Highlight completed milestones, risks, and next steps in five bullet points." |

The validated example mirrors Microsoft's own illustration, which also shows that only the source changes between a supplied-content prompt and an organisational-data prompt (the latter requires a Microsoft 365 Copilot licence and access to the referenced data). Microsoft's application card for Microsoft Copilot adds that users should include details, structure prompts, provide positive instructions, iterate and regenerate, and always review and verify AI-generated responses, and that organisations can extend Microsoft Graph grounding with connectors to improve relevance.

**Maker prompts (prompt actions and AI Builder prompts).** Validate them with **batch testing for prompts** PREVIEW (a production-ready preview), opened from the prompt's **Test hub** in Copilot Studio (Tools) or in the AI hub of Power Apps and Power Automate:

- Create test cases by **uploading** a CSV, **AI-generating** them from the prompt, using recent **activity data**, or adding them manually.
- Set a **passing score** and choose criteria: **Response quality** (clarity, helpfulness, tone), **Response matches** (specific words and meanings), or **JSON correctness** (output follows your schema).
- Run all or selected cases; an **accuracy score** is calculated; **Run history** tracks score progression and regressions across prompt versions.

For prompt quality before testing, the Copilot Agent Kit **Prompt Advisor** returns a confidence score, a detailed analysis, and suggested optimised prompts. For agent instructions, the triage framework in 5.1.4 warns about the "instruction budget": adding instructions for one signal can crowd out another, so every instruction change should be followed by a rerun of all affected evaluation sets.

**EXAM TIP**

If a prompt's output feeds a flow or another system, the decisive criterion is usually **JSON correctness**, not response quality. If the scenario says users get inconsistent results from Microsoft 365 Copilot, the expected answer focuses on prompt elements (goal, context, source, expectations), grounding access, and adoption enablement such as the Prompt Gallery, rather than on changing models.

### 5.2.4 Design end-to-end test scenarios of AI solutions that use multiple Dynamics 365 apps

The Dynamics 365 implementation guide describes the test types most projects use: unit, functional, process, end-to-end, user acceptance (UAT), regression, nonfunctional (including performance), and mock cutover. **End-to-end testing** connects all in-scope business processes and integrations, is "the most comprehensive type of test", requires a full set of migrated data and the latest solution version, starts in the Implement phase and continues into Prepare, runs in **multiple cycles**, and should be tracked in a tool such as Azure DevOps. It is the stage where "you can combine different Dynamics 365 apps and see them work together".

AI components change the design in three ways: each scenario needs **evaluation checkpoints** for generative steps alongside deterministic assertions; test data must include the **grounding data** the AI relies on, with realistic permissions; and scenarios must cover **human-in-the-loop** handoffs and failure paths (low confidence, escalation, tool error).

| Test type | AI-specific focus | Recommended tooling |
|---|---|---|
| **Unit** | Plug-ins, flows, custom connectors, prompt templates | Finance and operations: SysTest framework with the Acceptance test library (ATL); Power Platform: code-level unit tests |
| **Functional** | Copilot features configured as designed (for example, summaries, suggested replies) | Manual scripts plus Copilot Studio agent evaluation for agent features |
| **Process** | Agent or Copilot step inside one business process | UI automation (Playwright samples for model-driven and canvas apps; RSAT for finance and operations until its end of support) |
| **End-to-end** | Agents and Copilot across Sales, Customer Service or Contact Center, Supply Chain Management, Finance, and integrations | Orchestrated suite in Azure DevOps combining UI automation, API checks, and evaluation runs |
| **Integration (data)** | Data that agents use or write across apps | Finance and operations: Data task automation in the data management framework (RSAT isn't for integration tests) |
| **Performance** | Agent response time and error rate under load; back-end latency | Load scripts with multi-turn conversations (for example, Apache JMeter) |
| **UAT** | Users judge usefulness and trust; feedback captured | Business users with scripted and exploratory sessions; reactions and comments enabled |
| **Regression** | After Microsoft updates, model updates, knowledge or instruction changes | Automated UI suites plus stored evaluation test sets with baselines |

**Tooling decisions to get right in 2026.**

- **RSAT** (Regression suite automation tool) lets functional users convert Task recorder recordings into automated tests for finance and operations apps, integrated with Azure DevOps, with parameters in Excel. It suits business cycle and scenario tests (UAT), and also supports Cloud POS in Dynamics 365 Commerce. However, RSAT is **marked for deprecation and won't be supported after May 15, 2027**, and the deprecation notice states it is **not replaced by another feature**. New programmes should plan an alternative (Microsoft's regression tooling guidance lists Microsoft Playwright, EasyRepro, Selenium, and partner tools) and not deepen investment in RSAT.
- **EasyRepro** is an open-source, Selenium-based UI testing library for model-driven apps (Dynamics 365 customer engagement apps). Microsoft guidance still references it for UAT and regression automation of model-driven experiences.
- **Power Platform Playwright samples** provide a TypeScript framework with page objects for canvas apps, model-driven apps, custom pages, and Copilot-generated (Gen UX) apps, dual-domain authentication for Power Apps and Dynamics 365, CI/CD readiness, and AI-assisted authoring with the Playwright MCP server.
- **Power Apps Test Engine** was **deprecated in April 2026**; Microsoft names the Power Platform Playwright samples as the replacement.

**COMMON TRAP**

Answer options that recommend new investment in Power Apps Test Engine or a long-term RSAT-only strategy are likely to be wrong for a design that must remain supported. Read dates carefully: RSAT is still supported until May 15, 2027, so it can remain in use during a transition, but the architecture should include its successor.

**Example scenario: AI-assisted order-to-cash across apps.**

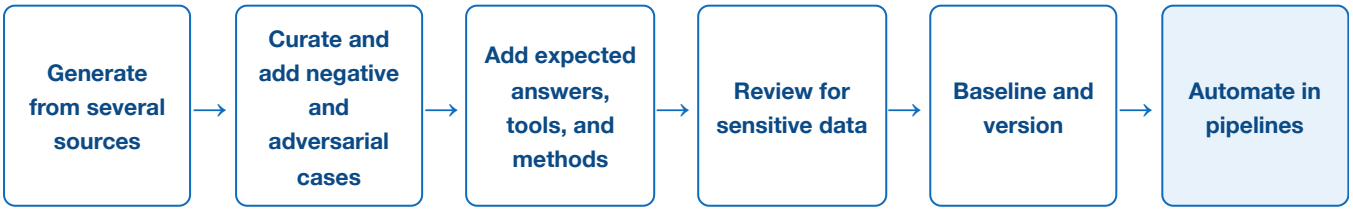
| Step | App and AI component | Deterministic assertion | AI evaluation checkpoint |
|---|---|---|---|
| 1 | Dynamics 365 Sales: seller uses Copilot to summarise the opportunity and prepare a quote | Quote created with correct price list and lines | Summary grounded in opportunity records (groundedness and completeness), no data from other customers |
| 2 | Integration to Supply Chain Management: order released, availability checked | Sales order exists with correct quantities and site | Not applicable |
| 3 | Copilot Studio agent in Teams: customer service representative asks for order status | Agent's tool call returns the order status from the system of record | Tool use method expects the order-status tool; compare meaning against expected status text |
| 4 | Dynamics 365 Contact Center: customer escalates by chat; agent hands off with context | Conversation routed to the correct queue with transcript attached | Escalation outcome recorded as User requested, context preserved |
| 5 | Dynamics 365 Finance: invoice posted, collections agent drafts a reminder | Invoice and customer balance correct | Custom method checks tone and policy compliance; content safety passes |
| 6 | Negative path: tool returns an error in step 3 | No duplicate records created | Agent acknowledges failure and escalates instead of looping |

Run the scenario with migrated, permission-realistic data in an integrated test environment, sign in as the persona each step requires (agent evaluation runs as the signed-in user), and record both test results and evaluation scores against the same Azure DevOps test case so that sign-off covers process correctness and AI quality together.

### 5.2.5 Build the strategy for creating test cases by using Copilot

Generating test cases with AI is fast, but generated tests inherit the blind spots of their source. A sound strategy combines several generators, curates their output, and treats test sets as versioned assets.

| Generator | What it produces | Good for | Blind spot |
|---|---|---|---|
| **Copilot Studio Quick question set (single response) or Quick conversation set** | 10 questions or 10 short conversations from the agent's description, instructions, and capabilities | Fast smoke tests after a change | Shallow; reflects what the maker intended, not what users ask |
| **Copilot Studio Full question set or Full conversation set** | Chosen number of questions from knowledge sources (best for generative orchestration) or topics (best for classic); short or long conversations | Coverage of existing content | Not good for testing information gaps; files up to 5 MB; individual SharePoint files only, not folders |
| **From the test chat** | Test set populated from the latest test chat questions, or a conversation case from the test chat | Capturing a reproduced defect as a regression test | Only as broad as the session |
| **From themes and drill-down lists (production data)** PREVIEW | Up to 100 real user questions from a theme or filtered list | Testing what users actually ask; per-theme quality tracking | Requires transcripts in Dataverse and Bot Transcript Viewer; may contain personal data |
| **Prompt Test hub AI-generate and activity data** PREVIEW | Test cases for a prompt from AI or recent prompt runs | Prompt actions and AI Builder prompts | Needs evaluation criteria and passing score |
| **GitHub Copilot with Playwright codegen and the Playwright MCP server** | UI tests for canvas, model-driven, and Gen UX apps rewritten into toolkit conventions | Deterministic app and process tests | Generated selectors, timeouts, and data need review; recordings often lack assertions |
| **GitHub Copilot testing for .NET ( `@Test` ) in Visual Studio** | Unit tests for C# code such as plug-ins | Code coverage for custom logic | Not for agent quality |




**Strategy rules for an architect:**

1. **Mix sources by purpose.** Knowledge-generated tests measure how well the agent uses what it already has; themes and production questions measure real demand and reveal gaps; manually written cases

cover policies, refusals, edge cases, and prompt injection attempts that no generator will produce on its own.

2. **Curate, don't just accept.** Every method except General quality needs expected responses, keywords, or tools. Generated cases must be reviewed by a subject matter expert, and expected answers must be checked against current policy.
3. **Protect data.** When Copilot Studio generates test cases it uses the connected account's credentials, so generated cases can include sensitive data that account can access, and any maker with access to the agent can view the test sets. Use a test identity with appropriate access and review generated content. Test case generation fails if questions violate the agent's content moderation settings.
4. **Respect limits and retention.** Single response sets hold up to 100 cases (import files up to 100 questions, each up to 1,000 characters, in CSV or text with *Question* and *Expected response* columns); conversation sets hold up to 20 cases. Results are kept 89 days, so export results you need for audit.
5. **Version and automate.** Keep test sets with the solution, trigger evaluations through the REST API or connectors, and use Power Platform pipelines (the Copilot Agent Kit includes a pattern for automated testing and deployment) so that promotion is blocked when thresholds fail.
6. **For UI tests generated with GitHub Copilot**, follow Microsoft's best practices: review selectors, set realistic timeouts for Dataverse-backed galleries, use unique test data, and add meaningful `expect` assertions; keep a well-written reference test and custom instructions in the repository so the assistant follows team conventions.

**PRODUCTION NOTE**

Multilingual agents need language-aware tests. In Copilot Studio evaluation, you select the language for a run; if the compare meaning grader detects a language mismatch, the response fails for language inconsistency. Agent evaluation in GCC has limitations (no user profile in test sets; no text similarity method).

## 5.3 Additional practice questions

**Practice 5.1** MULTIPLE CHOICE

A bank uses a Copilot Studio conversational agent with generative orchestration. Business rules transfer any payment request above a limit to a person. The service manager complains that the overall escalation rate is too high, but you suspect most escalations are expected. You need to identify only the escalations where users became stuck after repeated failures. Which outcome reason should you analyse?

A. Resolved implied
B. System intended
C. System unintended
D. User requested

**Practice 5.2** MULTIPLE CHOICE

A Centre of Excellence manages 40 Copilot Studio agents, some on the standard harness and some on the GitHub Copilot harness, in a managed environment. It needs a single, centrally owned, OpenTelemetry-aligned view of agent invocations and tool executions, and makers must not configure telemetry themselves. What should you recommend?

A. Add an Application Insights connection string to each agent under Settings > Advanced.
B. Create a Copilot Studio export package for environment-level telemetry in the Power Platform admin center and block the Application Insights in Copilot Studio connector in a data policy.
C. Share each agent's Monitor page with the CoE by using the Analytics Viewer role.
D. Deploy the Viva Insights Copilot Studio agents report.

**Practice 5.3** MULTIPLE CHOICE

An agent sends agent-level telemetry to Application Insights. Your daily user chart is inflated by conversations makers run in the Copilot Studio test canvas. Which filter should you add to the `customEvents` query?

A. `where tostring(customDimensions['channelId']) == "msteams"`
B. `where tostring(customDimensions['designMode']) == "False"`
C. `where itemType == "dependency"`
D. `where isempty(operation_ParentId)`

**Practice 5.4** MULTIPLE CHOICE

An AI engineer uses the Python SDK to create a continuous evaluation rule for a Microsoft Foundry agent. Agent traffic is flowing and the rule exists, but no continuous evaluation results appear on the agent's Monitor tab. What should you do first?

A. Increase `max_hourly_runs` to 1,000.
B. Assign the Foundry User role to the Foundry project's managed identity.
C. Switch the evaluation level from turn to conversation.
D. Run a cluster analysis on the latest evaluation run.

**Practice 5.5** MULTIPLE CHOICE

An HR agent must decline to give legal advice and instead point employees to the legal team. In agent evaluation, test cases for legal questions keep failing under General quality even when the agent declines correctly. You need these cases to pass when the refusal is appropriate. What should you do?

A. Lower the pass score of General quality to 40.
B. Add a Custom test method with evaluation instructions and labels that mark an appropriate refusal as Pass.
C. Replace General quality with Exact match.
D. Remove the legal questions from the test set.

**Practice 5.6** MULTIPLE CHOICE

A contracts agent must reproduce a standard indemnity clause with nearly identical wording and sentence structure, but minor punctuation differences are acceptable. Which Copilot Studio test method best validates this requirement?

A. Compare meaning
B. Keyword match (All)
C. Text similarity
D. Exact match

**Practice 5.7** MULTIPLE CHOICE

You assess a medium-risk, customer-facing agent by using Microsoft's risk-based thresholds. Averages over three runs are: safety and compliance 93%, core business 88%, capabilities 72%. What is the readiness decision?

A. Deploy
B. Conditional deployment
C. Iterate
D. Block

**Practice 5.8** MULTIPLE CHOICE

A company trains an AI Builder category classification model to route incoming service emails to specialist teams. Misrouted emails waste significant specialist time, while emails the model can't classify go to a general queue that handles them adequately. Which metric should be the primary validation criterion?

A. Recall
B. Precision
C. R-squared
D. Overall accuracy

**Practice 5.9** MULTIPLE CHOICE

On a held-out test set of 1,000 records, a churn model produces TP = 300, FP = 100, FN = 200, TN = 400. What is the F1 score, rounded to two decimals?

A. 0.60
B. 0.67
C. 0.70
D. 0.75

**Practice 5.10** MULTIPLE CHOICE (CHOOSE TWO)

A manufacturer runs Dynamics 365 Supply Chain Management, Finance, and Sales with Copilot features and a Copilot Studio order-status agent. Its regression suite for finance and operations is built on RSAT, and its roadmap runs to 2029. You need to recommend a supportable testing approach. Which two actions should you recommend? Each correct answer presents part of the solution.

A. Plan a migration of finance and operations UI regression tests away from RSAT, because it isn't supported after May 15, 2027 and has no replacement feature.
B. Move all unit tests into RSAT to consolidate tooling.
C. Keep stored Copilot Studio evaluation test sets with baselines and run them as part of each regression cycle.
D. Use RSAT to test data integrations between Finance and Sales.
E. Standardise on Power Apps Test Engine for model-driven apps.

**Practice 5.11** MULTIPLE CHOICE

A team tests canvas apps and model-driven apps with Power Apps Test Engine YAML files. They want AI-assisted test authoring with GitHub Copilot, page objects, and CI/CD execution, on a supported framework. What should you recommend?

A. Extend Test Engine with custom Power Fx functions.
B. Adopt the Power Platform Playwright samples framework with the Playwright MCP server.
C. Migrate the tests to RSAT.
D. Use Copilot Studio agent evaluation for app UI tests.

**Practice 5.12** MULTIPLE CHOICE

A utility's Copilot Studio agent has been live for three months. Analysts suspect answer quality for billing questions is poor, but generated test sets based on the agent's knowledge sources all pass. You need a test set that reflects what customers actually ask about billing. What should you use?

A. A Quick question set
B. A Full question set generated from the billing knowledge source
C. The Evaluate action on the billing theme on the Monitor page
D. A test set built from the latest test chat

## Answers and explanations

**5.1: C.** System unintended escalations occur when a session exceeds maker-defined thresholds such as repeated failures, which usually means the user was stuck. System intended (B) reflects business rules like the payment limit and is expected. User requested (D) is an explicit request for a person. Resolved implied (A) isn't an escalation.

**5.2: B.** Environment-level telemetry (preview) is configured once per managed environment by administrators, supports both harnesses, and emits OpenTelemetry-aligned spans to the `dependencies` table. Blocking the Application Insights in Copilot Studio connector prevents makers from enabling agent-level telemetry. Option A is decentralised, event based, and standard-harness only; C and D don't provide trace-level telemetry.

**5.3: B.** All agent-level events carry the `designMode` custom dimension, which is True for test canvas conversations. Filtering on a channel (A) would exclude legitimate channels; C and D relate to environment-level spans, not `customEvents`.

**5.4: B.** Continuous evaluation rules require the project managed identity to have the Foundry User role (formerly Azure AI User); Microsoft's troubleshooting table lists this for missing results. Raising `max_hourly_runs` (A) addresses skipped runs after hitting the default limit of 100 per hour, not missing results. C and D don't enable evaluation.

**5.5: B.** General quality includes an abstention criterion, so appropriate refusals can be flagged. Microsoft recommends a Custom method with instructions and labels that define the expected refusal as a pass. Lowering thresholds (A) or removing cases (D) hides the signal; Microsoft notes that iteration isn't complete if scores improve only because difficult cases were removed. Exact match (C) is too brittle.

**5.6: C.** Text similarity uses cosine similarity of wording and structure with a pass threshold, suiting near-verbatim text such as legal clauses. Compare meaning (A) checks intent, not wording. Keyword match (B) checks presence of terms, not construction. Exact match (D) fails on any difference, including punctuation.

**5.7: D.** For a medium-risk customer-facing agent the safety and compliance starting threshold is 95%, and the quick reference treats safety and compliance below 95% as Block. Safety failures must be triaged before considering the other scores.

**5.8: B.** When acting on a wrong positive prediction is costly (misrouting), precision is the priority; Microsoft's custom text classification guidance uses ticket triage as its example. Recall (A) matters when misses are costly. R-squared (C) is for numerical prediction. Accuracy (D) hides the asymmetry.

**5.9: B.** Precision = 300 ÷ 400 = 0.75; recall = 300 ÷ 500 = 0.60; F1 = 2 × 0.75 × 0.60 ÷ 1.35 ≈ 0.67. Accuracy would be 700 ÷ 1,000 = 0.70 (C), which is a distractor.

**5.10: A and C.** RSAT isn't supported after May 15, 2027 and isn't replaced by another feature, so a 2029 roadmap needs a successor. Regression must also cover AI quality, which stored evaluation test sets with baselines provide. Unit tests belong in SysTest with ATL (B), integration testing uses data task automation rather than RSAT (D), and Test Engine was deprecated in April 2026 (E).

**5.11: B.** Microsoft deprecated Test Engine in April 2026 and names the Power Platform Playwright samples as the replacement; it offers page objects, AI-assisted authoring through the Playwright MCP server with assistants such as GitHub Copilot, and CI/CD support. RSAT (C) is for finance and operations apps. Agent evaluation (D) tests agent responses, not app UI.

**5.12: C.** Themes (preview) cluster real user questions that triggered generative answers; the Evaluate action creates a test set of up to 100 questions from that theme. Knowledge-generated sets (B) test how the agent uses existing content and aren't good at revealing information gaps. Quick sets (A) and test chat sets (D) reflect maker intent rather than production demand.

### Key takeaways

- Choose monitoring tools by build surface and audience: Copilot Studio Monitor page for owners, Application Insights (agent-level or environment-level) for engineering and operations, Power Platform admin center and Microsoft 365 admin center for administrators, Copilot Analytics for leadership, and the Foundry Monitor tab for Foundry agents.
- Outcome reasons make metrics actionable: separate System intended, System unintended, and User requested escalations, and Resolved confirmed from Resolved implied. Set `conversationOutcome` in tools when tasks end outside topics.
- Feedback needs design: reactions, comments, CSAT, sentiment, and themes depend on settings, roles (Analytics Viewer, Bot Transcript Viewer), Dataverse transcripts, admin consent for the Microsoft Copilot channel, and retention (28 to 30 days by default).
- AI-based diagnostics exist at every layer: Summary, answer quality reasons, themes, custom metrics, sentiment, Copilot Agent Kit analysers, Foundry cluster analysis, and the Azure Copilot Observability Agent. Many are preview.
- Agent-level telemetry is event based in `customEvents`; environment-level telemetry (preview) is OpenTelemetry-aligned spans in `dependencies`. Filter out test traffic and know the documented gaps.
- Use the triage framework: classify failures as evaluation setup, agent configuration, or platform limitation; fix categories when 80% share a cause; apply "change X, rerun Y, expect Z".
- Set evaluation thresholds by risk, baseline over at least three runs, accept about 5% run-to-run variance, and decide Block, Iterate, Conditional deployment, or Deploy.
- Validate custom models with the metric that reflects error costs: precision, recall, F1, AUC, R-squared; for generative output use groundedness, relevance, completeness, safety, and agent evaluators.
- Prompt validation: goal, context, source, expectations for users; batch testing with passing scores and JSON correctness for maker prompts.
- End-to-end tests across Dynamics 365 apps need migrated data, multiple cycles, Azure DevOps tracking, and AI evaluation checkpoints. RSAT ends support after May 15, 2027 with no replacement; Test Engine was deprecated in April 2026 in favour of Power Platform Playwright samples.
- Generate test cases from several sources (quick, knowledge, test chat, themes, activity data, Playwright codegen with GitHub Copilot), then curate, protect sensitive data, version, and automate.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Exam AB-100: Agentic AI Business Solutions Architect
- Course AB-100T00-A: Architecting agentic AI business solutions
- Monitor, measure, and report value
- Monitor overview (Copilot Studio)
- Monitor conversational agents
- Monitor autonomous agents
- Monitor user questions by theme (preview)
- Monitor your agent with custom metrics (preview)
- Monitor operations, compliance, and capacity (Copilot Studio governance)
- Telemetry with Application Insights overview
- Agent-level telemetry with Application Insights

- Environment-level telemetry with Application Insights (preview)
- Application Insights telemetry with Microsoft Copilot Studio
- Harnesses in Copilot Studio
- About agent evaluation
- Choose evaluation methods
- Create a single response test set
- Create a conversational test set
- Improve agents by using evaluation-driven triage and remediation
- Layer 1: Interpret evaluation scores and assess readiness
- Design and operationalize agent evaluation
- Batch testing for prompts (preview)
- Plan and create a conversational agent performance test
- Copilot Agent Kit overview
- Enhance agent testing with Copilot Agent Kit
- Observability in generative AI (Microsoft Foundry)
- Built-in evaluators reference
- Monitor agents with the Agent Monitoring Dashboard
- Analyze evaluation results with cluster analysis (preview)
- Agent tracing overview
- Azure Copilot Observability Agent
- Monitor AI agents with Application Insights
- Metrics and recommendations for Copilot Studio (Power Platform admin center)
- Track, manage, and scale Copilot adoption in the Power Platform
- Microsoft Copilot reporting options for admins
- Microsoft Copilot Agents usage report
- View Copilot analytics report (Dynamics 365 Contact Center)
- Prediction model performance (AI Builder)
- Understand category classification model performance
- Manage your model in AI Builder
- View the accuracy and performance of predictive scoring models
- Results of machine learning models (Finance insights)
- Characteristics and limitations for using custom text classification
- Write effective prompts to achieve optimal results
- Review prompting best practices
- Application card: Microsoft Copilot (for organizations)
- Types of tests (Dynamics 365 implementation guide)
- Checklist for your test strategy
- Options for regression testing
- Regression suite automation tool (RSAT)
- Removed or deprecated platform features (finance and operations)
- Testing tools for client-side development (EasyRepro)
- Power Platform Playwright samples overview
- AI test authoring with Copilot (Playwright samples)
- AI-assisted testing overview (Playwright samples)
- Important changes (deprecations) coming in Power Platform
- Generate and run unit tests using GitHub Copilot testing for .NET

DOMAIN 3 · PART B

# Application lifecycle management for AI solutions

**Part of Domain 3: Deploy AI-powered business solutions (40–45%)**

This chapter covers the skill group *Design the ALM process for AI-powered business solutions*. It also covers two audience-profile responsibilities: a cohesive application lifecycle management (ALM) strategy for agentic-first solutions, and a cohesive environment strategy for AI-powered solutions that considers third-party AI. Expect to choose the lifecycle vehicle for each artefact, place gates in the right stage, and predict what does and does not travel when you promote an agent, model, dataset, or Copilot feature.

Study it as a set of transport questions. For every artefact ask: where is it authored, what packages it, what deploys it, and what must be reconfigured after deployment. Most traps sit in the last question.

| Official skill (Design the ALM process for AI-powered business solutions) | Covered in |
|---|---|
| Design the ALM process for data used in AI models and agents | 6.3 |
| Design the ALM process for Copilot Studio agents, connectors, and actions | 6.4 |
| Design the ALM process for Microsoft Foundry Agents Service | 6.5 |
| Design the ALM process for custom AI models | 6.6 |
| Design the ALM process for AI in Dynamics 365 apps for finance and supply chain | 6.7 |
| Design the ALM process for AI in Dynamics 365 apps for customer experience and service | 6.8 |
| Audience profile: create a cohesive ALM strategy for agentic-first solutions | 6.1 |
| Audience profile: create a cohesive environment strategy for AI-powered solutions, with consideration for third-party AI solutions | 6.2 |

## 6.1 A cohesive ALM strategy for agentic-first solutions

### 6.1.1 Why ALM for AI differs from ALM for applications

Copilot Studio guidance defines ALM as the processes, people, and tools that govern an agent's lifecycle, from idea through development, testing, deployment, and maintenance until decommissioning. An agentic solution has more moving parts than a classic application, and several change behaviour without a code change:

- **Definitions and configuration**: instructions, topics, tools, agent flows, Dataverse customizations, X++ models, Foundry agent definitions.
- **Models**: base, fine-tuned, and AI Builder models and prompts, introduced and retired on a vendor schedule. An agent that works well with one model can behave very differently with another, even in the same family.
- **Data**: grounding content, indexes, reference data, training and evaluation datasets. Solutions and deployment pipelines generally carry metadata, not data.

- **Runtime and tenant settings**: channels, authentication, telemetry, sharing, Copilot switches, data movement, external model permissions. Many are not packaged and must be reapplied per environment.
- **Quality evidence**: evaluation results, review scores, and approvals, versioned with the release.

A cohesive strategy therefore has one release train with several transport lanes, all gated by the same evaluation and approval policy.

### 6.1.2 Power Platform ALM building blocks you must know

Copilot Studio is built on the Power Platform foundation, so its ALM uses the same building blocks: an environment strategy, solutions, environment variables, connection references, CI/CD for no-code and pro-dev teams, and native Git integration.

| Building block | What it does | Architect's design point |
|---|---|---|
| Unmanaged solution | The developed form, used in development and stored in source control. | Deleting it removes only the container; customizations stay in the default solution. |
| Managed solution | The deployed form for every non-development environment; exported from the unmanaged solution and treated as a build artefact. | Not directly editable; uninstalling removes its customizations and data in custom tables and columns it owns. Can't be imported into the environment holding its unmanaged origin. |
| Publisher and prefix | The publisher owns the components it creates; the prefix avoids collisions. | Define one custom publisher early; a component's publisher can't be changed later. |
| Update, upgrade, patch | Upgrade removes components no longer present and can be staged; patches and updates can't delete components. | Pipelines in Power Platform import as *Upgrade* without overwriting customizations. |
| Environment variables | Per-environment settings: decimal, text, JSON, two options, data source, secret. | Use the secret type backed by Azure Key Vault (the only supported store); text values aren't encrypted. |
| Connection references | Decouple components from each environment's connection. | On import, connections must be owned by, or shared with, the reference owner. |
| Git integration | Native source control with Azure DevOps or GitHub. | Use with developer environments, not test or production. |
| Managed environments | Premium governance: sharing limits, solution checker enforcement, pipelines, routing, environment groups. | Pipeline targets must be managed environments. |

**EXAM TIP**

Memorise the Copilot Studio ALM golden rules: do not customize outside a development environment; always work in solutions; use a custom publisher and prefix; create separate solutions only when you need to deploy components independently; use environment variables for settings and secrets that change across environments; export and deploy as managed unless you are setting up a development environment; automate ALM where possible. Many scenario items are a violation of one of these rules in disguise.

### 6.1.3 Choosing the deployment tool per platform

An agentic solution usually spans several platforms, so choose a deployment mechanism per platform and orchestrate them into one release.

| Platform or artefact | Native lifecycle vehicle | Automation choices | Choose when |
|---|---|---|---|
| **Copilot Studio agents, agent flows, prompts, AI Builder models, Dataverse customizations, Dynamics 365 CE customizations** | Dataverse solution (managed downstream) | Pipelines in Power Platform; Azure DevOps with Power Platform Build Tools; GitHub Actions for Power Platform; pac CLI | See the Copilot Studio comparison in 6.4.5 |
| **Foundry Agent Service hosted agents** | Agent version created by deployment; infrastructure as code (Bicep or Terraform) | `azd` with `azd pipeline config` for GitHub Actions or Azure DevOps; SDK or REST; Terraform for the data-plane resource | Pro-code agents needing gated promotion and canary rollout |
| **Foundry prompt agents** | Immutable agent versions referenced as `name:version` | SDK `create_version`, portal, REST | Declarative agents |
| **Fabric items (lakehouses, semantic models, data agents)** | Fabric Git integration and deployment pipelines (two to 10 stages) | Deployment pipelines REST APIs, Fabric CLI tasks in Azure DevOps | Analytical and grounding assets |
| **Dataverse configuration and reference data** | Configuration Migration tool schema and data file | `pac data export` and `pac data import`; Build Tools data tasks | Records solutions don't carry |
| **Finance and operations apps (X++ and Dataverse)** | Unified package (Dataverse solutions plus X++ modules) in the unified developer experience; deployable package in Lifecycle Services for legacy environments | Power Platform CLI or Azure DevOps tasks; Lifecycle Services Asset library for legacy | Customizations with X++ or ERP metadata |

**COMMON TRAP**

The ALM Accelerator for Power Platform is documented as deprecated: no new features are added and issues are no longer reviewed. If an answer option proposes adopting the ALM Accelerator for a new programme, prefer Pipelines in Power Platform, extended with Azure DevOps or GitHub where needed.

The core Power Platform transport, which recurs throughout this chapter, can be scripted with the Power Platform CLI. The commands below are from the `pac solution` and `pac auth` references.

**Listing 6.1: Export, unpack, check, and import a solution with pac CLI**

```
# Authenticate to the development environment
pac auth create --environment https://contoso-dev.crm.dynamics.com

# Export the unmanaged solution for source control and the managed build artefact
pac solution export --name ContosoAgents --path .\out\ContosoAgents.zip
pac solution export --name ContosoAgents --path .\out\ContosoAgents_managed.zip --managed

# Unpack the unmanaged export into source-controlled files
```

```
pac solution unpack --zipfile .\out\ContosoAgents.zip --folder .\src\ContosoAgents --packagetype
Unmanaged

# Static analysis with solution checker
pac solution check --path .\out\ContosoAgents_managed.zip --outputDirectory .\checker

# Generate a deployment settings file (connection references and environment variables)
pac solution create-settings --solution-zip .\out\ContosoAgents_managed.zip --settings-file
.\config\test.json

# Import into test with environment-specific values, upgrading the existing managed solution
pac auth create --environment https://contoso-test.crm.dynamics.com
pac solution import --path .\out\ContosoAgents_managed.zip --settings-file .\config\test.json --
stage-and-upgrade --async
```

## 6.2 A cohesive environment strategy for AI-powered solutions

### 6.2.1 Environment types and the minimum lifecycle

Copilot Studio ALM guidance sets the floor: at least three environments (development, test, and production). Configure production as a production-type environment and all others, including development and test, as sandbox type, and secure every environment with a Microsoft Entra security group. The Power Platform tenant environment strategy adds the full palette of environment types:

| Type | Characteristics relevant to AI solutions |
|---|---|
| Default | Every tenant has one; used by many Microsoft 365 experiences. Not intended for long-term work beyond personal productivity. |
| Production | For permanent work. Managed production environments support extended backup retention up to 28 days. |
| Sandbox | Non-production; supports copy and reset. Best for testing and ALM build environments. |
| Developer | Personal maker workspaces; up to three per maker; do not consume tenant capacity; removed after 90 days of inactivity if the owner does not respond. Dynamics 365 apps are not available in developer environments. |
| Trial | Short-term proofs of concept; automatically removed. |
| Dataverse for Teams | Created automatically for apps built in Teams; security aligned to the team. |

**COMMON TRAP**

Because Dynamics 365 apps are not available in developer environments, a Copilot Studio agent that extends Dynamics 365 Sales or Customer Service must be developed in a sandbox that has the Dynamics 365 apps installed, not in a maker's personal developer environment.

### 6.2.2 Zoned governance for agents

Copilot Studio guidance segments agent building into three zones based on purpose and risk, and the zone drives the environment and ALM pattern:

| Zone | Who builds | Environments and ALM | Controls |
|---|---|---|---|
| **Zone 1: Citizen development** | Anyone, for personal and team productivity, with Agent Builder in Microsoft Copilot and SharePoint agents | Developer environments, environment routing keeps agents isolated to the maker, sharing disabled or scoped to the maker | Only Microsoft 365 and Power Platform connectors; agents run in the user's context |
| **Zone 2: Partnered development** | IT-approved makers with Copilot Studio | Admin-approved environment provisioning, scoped roles and sharing, ALM pipelines for agent versioning, IT-admin approval to publish | Zone-specific advanced connector policies; environment groups and rules |
| **Zone 3: Professional development** | Pro developers and IT, with Foundry Agent Service and Copilot Studio | Organizational standard ALM and strongest security controls; can carry SLAs; sharing managed through Integrated Apps in the Microsoft 365 admin center | Advanced connector policies plus Microsoft Purview |

### 6.2.3 Mapping one release to several platforms

Map stages explicitly so that "test" means the same thing on every platform.

| Platform | Isolation unit per stage | Documented isolation rule |
|---|---|---|
| **Power Platform, Copilot Studio, Dynamics 365 CE** | Environment (with Dataverse) | Defines data boundaries, security roles, data policies, and lifecycle separation. |
| **Foundry Agent Service** | Foundry project bound to an `azd` environment | Separate project per environment; names alone don't isolate resources, and dependencies and permissions aren't copied. |
| **Microsoft Fabric** | Workspace assigned to a deployment pipeline stage | Each stage has its own workspace; data, permissions, and workspace settings are not copied. |
| **Finance and operations apps** | Unified environment (Power Platform environment with Dataverse and the ERP workload) | Managed in the Power Platform admin center; code and data are copied together on environment copy. |
| **Pipelines in Power Platform host** | Production environment acting as host | Host can't also be a development environment; cross-geo deployment needs the Cross-Geo Solution Deployments setting or separate hosts. |




### 6.2.4 Third-party AI in the environment strategy

The audience profile asks you to consider third-party AI in the environment strategy. In the Microsoft stack it appears in three ways, each controlled at environment or environment-group scope:

- **External models in Copilot Studio.** Makers can select Anthropic, xAI, or Mistral models as an agent's primary model. Two independent admin actions are required: turn on external models in the Power Platform admin center for the environment or environment group, and allow each provider in the Microsoft 365 admin center. Anthropic operates as a Microsoft subprocessor; Mistral models are hosted outside Microsoft under terms admins must accept.
- **Preview and experimental models.** A separate setting. These models might process and store data outside the organization's geography, so the environment also needs data movement across regions turned on. They aren't recommended for production agents.
- **Non-Microsoft tools and agents.** Connectors, custom connectors, MCP servers, and connected agents, governed by data policies and advanced connector policies.

A practical pattern: environment groups per lifecycle stage and risk zone, with external or preview models enabled only where approved (for example the test group of one programme), and production groups restricted until legal, data residency, and responsible AI reviews complete. Group rules lock the matching environment settings, so an environment admin can't quietly re-enable a blocked capability.

> **NOTE**
> The model list a maker sees reflects that environment's admin settings. A model documented as available can be missing in production if the setting is off there. Put these settings in the environment build checklist so test and production don't drift.

### 6.2.5 Governance controls that keep environments consistent

Prefer built-in platform features over custom tooling:

- **Environment groups and rules.** Only managed environments can join; one group per environment; no nesting; groups can span regions. Rules cover sharing, usage insights, maker welcome content, solution checker enforcement, backup retention, and AI-generated descriptions, and lock the matching environment settings.
- **Default environment routing** moves makers into personal developer environments, the first lifecycle stage.
- **Solution checker enforcement** (None, Warn, Block): commonly Block for default, production, and Teams environments; Warn for developer and sandbox.
- **Early release cycle environments** to validate platform and model changes before they reach business-critical environments.

> **PRODUCTION NOTE**
> Starting in October 2026, the admin deployment page notifies admins when pipelines deploy to unmanaged target environments. Admins have 30 days to approve enabling managed environments (with one optional 30-day extension per environment) before further deployments to that target are blocked. Plan to make every pipeline target a managed environment from the start.

## 6.3 Design the ALM process for data used in AI models and agents

### 6.3.1 Classify the data before you design its lifecycle

The single most tested fact in this area is a negative one: the containers you use for application ALM do not move data. The Pipelines in Power Platform FAQ states that pipelines, and solutions in general, don't contain data stored in Dataverse tables. Fabric deployment pipelines copy metadata, not data. An AI Builder model in a solution carries the model executable but not its training data. Start the design by classifying each data asset, because each class has a different lifecycle.

| Data class | Examples in an agentic solution | Lifecycle vehicle | Versioning approach |
|---|---|---|---|
| **Configuration and reference data** | Dataverse records that drive agent behaviour: product catalogues, routing rules, business terms, prompt library records stored in tables | Configuration Migration tool, `pac data`, Build Tools data tasks | Schema file and data file in source control |
| **Grounding content** | SharePoint libraries, public websites, uploaded files, Dataverse tables used as knowledge | Lives in the source system per environment; agent points to it through environment variables or variables in knowledge URLs | Content owner's versioning (for example SharePoint) plus a release record of which source each environment uses |
| **Derived indexes and embeddings** | Azure AI Search indexes, vector stores, chunking outputs | DataOps pipelines (for example Azure Machine Learning pipelines) that rebuild or incrementally update the index | Version the chunking and embedding configuration and the pipeline code; rebuild per environment |
| **Analytical data products** | Lakehouses, warehouses, semantic models, Fabric data agents | Fabric Git integration and deployment pipelines (metadata) plus data pipelines per workspace (data) | Git for item definitions; deployment rules for data source binding |
| **Training and fine-tuning data** | JSONL training and validation files, labelled documents for AI Builder | MLOps pipelines; stored in governed storage | Dataset versions linked to the model version they produced |
| **Evaluation datasets** | Test queries, ground truth, rubric definitions | Source control alongside the agent; consumed by CI evaluation steps | Same commit as the agent definition they validate |

Architecture Center GenAIOps guidance calls this DataOps: reproducible workflows with data versioning. For retrieval-augmented generation (RAG), versioning lets you compare chunking and embedding strategies and roll back. Chunking pipelines aren't part of traditional MLOps, so extend operations to cover grounding data stores.

### 6.3.2 Dataverse configuration data

Dataverse records that drive agent behaviour must move with the release even though the solution doesn't carry them. The Configuration Migration tool defines a schema (tables, columns, relationships, uniqueness rules, plug-in handling), exports data and schema to a zip, and imports in multiple passes so foundation data

loads before dependent data. Define uniqueness rules carefully, rehearse in a production-like environment, and back up production first. Calendar and image columns aren't supported.

**Listing 6.2: Move configuration data with pac data (schema created with the Configuration Migration tool)**

```
# Export reference data from the source environment
pac auth create --environment https://contoso-dev.crm.dynamics.com
pac data export --schemaFile .\data\agent-config-schema.xml --dataFile .\data\agent-config-data.zip
--overwrite

# Import into the target environment after the managed solution that defines the tables
pac auth create --environment https://contoso-test.crm.dynamics.com
pac data import --data .\data\agent-config-data.zip
```

**EXAM TIP**

Import the managed solution that defines the tables before the data. Power Platform Build Tools include export data and import data tasks, so data can move in the same pipeline run.

### 6.3.3 Grounding data and knowledge sources across environments

Each stage should ground on its own content. Two documented mechanisms avoid editing the agent after import:

- **Environment variables** for URLs and identifiers that differ per environment, populated by deployment settings files or pipelines.
- **Variables in knowledge source URLs.** Copilot Studio accepts a global string variable in a public website or SharePoint knowledge URL, appending a path or replacing the URL. It can be set from an environment variable; switching between development, test, and production sites is a documented scenario. URL depth limits still apply.

Search indexes must stay fresh and valid, be rebuilt when they can't be updated in place, and honour compliance requests such as right to be forgotten. Promote the index build pipeline, not the index contents.

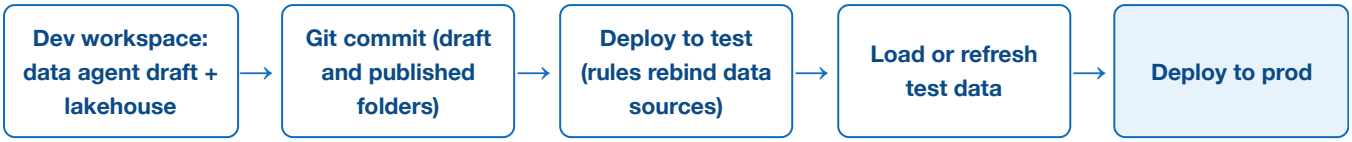
**COMMON TRAP**

Don't copy production content into test without a data protection decision. Grounding data inherits its source's sensitivity; use a governed or masked copy and record the decision.

### 6.3.4 Fabric deployment pipelines and Fabric data agents

A Fabric deployment pipeline has two to 10 stages (three by default), each a workspace. Deployment copies item metadata and overwrites paired items; data sources and parameters can be overridden by deployment rules. **Not** copied: data, URLs, IDs, permissions, workspace settings, and app content, plus semantic model role assignments, refresh schedules, and credentials.

Source control for Fabric data agents is in preview PREVIEW. Git stores a draft folder and a published folder (edit the draft; publishing regenerates the published folder). Deployment pipelines promote the agent across workspaces and can auto-create test and production workspaces. Service principals are supported for data agents only in these ALM scenarios. The Azure DevOps extension for Fabric runs Fabric CLI commands for scripted promotion.




**NOTE**

Because data isn't copied, a data agent promoted to an empty production workspace is correct ALM and still useless. Plan data loading as a release task.

### 6.3.5 Evaluation and test datasets are release artefacts

Every evaluation gate depends on a dataset of queries, expected outcomes, and evaluators. Keep it in the agent's repository, change it through pull requests, record the last baseline run, and add new production failure scenarios to it.

## 6.4 Design the ALM process for Copilot Studio agents, connectors, and actions

### 6.4.1 Solutions in Copilot Studio

Copilot Studio agents are created in the default solution automatically. To export, import, and manage agents across environments you must create a custom solution, or set a **preferred solution** so new agents and components are created in it from the start. The Copilot Studio solution explorer supports the usual tasks: create a custom solution, add components, import and export, upgrade or update a solution with an agent, remove an unmanaged layer from a managed agent, and create and manage solution pipelines. Exporting and importing agents requires at least the System Customizer security role.

Key behaviours documented for agents powered by the standard harness:

- Add new topics or flows to the agent in the unmanaged solution before export (**Advanced**, **Add required objects**).
- Instructions are included; some knowledge sources and components might not be if stored as separate resources or Dataverse tables.
- Conversation ID, bot ID, environment ID, and comments don't transfer; icon and channel details can be empty.
- Removing an agent from a solution doesn't remove its components; don't remove topics directly from the solution unless you also remove the agent.
- A solution can't be exported if any topic name contains a period.

**NOTE**

Copilot Studio now distinguishes harnesses: the standard harness (rule-based agents and agent flows), the GitHub Copilot harness (reasoning-heavy agents and workflows), and the Copilot chat harness (extending Microsoft Copilot Chat). The export and import article is scoped to the standard harness. Check the harness-specific documentation before assuming identical ALM behaviour for agents on the other harnesses, and record the harness for each agent in your inventory, as the model lifecycle guidance recommends.

### 6.4.2 What is solution-aware and what is not

This is the decision table to memorise. The Copilot Studio ALM guidance lists items that aren't solution-aware and require post-deployment steps on the downstream environment.

| Item | Travels in the solution? | What to do in the target environment |
|---|---|---|
| **Agent, instructions, topics, agent flows, prompts** | Yes (add required objects before export) | Nothing, beyond publishing the agent |
| **Environment variables** | Definition yes; current value can be removed from the solution | Supply values through the deployment settings file or pipeline prompt |
| **Connection references** | Yes | Bind to connections in the target (deployment settings file, pipeline, or interactive) |
| **Custom connectors** | Yes, when added to a solution | Import the custom connector first, then the connection reference with the agent solution |
| **Component collections (shared topics, actions, knowledge, entities)** | Yes, through their own solution | Import collection versions that match those used in development |
| **Azure Application Insights settings** | No | Reconfigure |
| **Manual authentication settings** | No | Reconfigure user authentication after import |
| **Direct Line and web channel security settings** | No | Reconfigure secrets and security |
| **Deployed channels** | No | Publish and configure channels per environment |
| **Sharing with makers or end users** | No | Share per environment (pipelines can propose sharing targets for approval) |
| **Data in Dataverse tables** | No | Move with configuration data tooling |

**EXAM TIP**

After import, publish the agent before it can be shared and configure user authentication again. "Users can't reach the agent after a successful import" usually hinges on publish, channel, authentication, or sharing.

### 6.4.3 Connectors, tools, and MCP servers

Agents act through tools: connectors, custom connectors, agent flows, prompts, MCP servers, and connected agents. Three ALM rules apply:

1. **Connections never travel; connection references do.** Each environment binds the reference to its own connection. Pipelines let makers supply connections up front and admins preconfigure them; service principal connections work for connectors that support them, including custom connectors.
2. **Custom connectors go first.** Import the custom connector before the connection reference and agent solution. Custom connectors can use environment variables, including Key Vault secrets.

3. **MCP tools are dynamic.** When an MCP server changes its tools, Copilot Studio reflects it automatically, so the solution version doesn't pin the tool surface. Give the MCP server its own versioned lifecycle and per-stage endpoints selected through environment-specific connections.

> **COMMON TRAP**
> Pipelines can't update connection references that have no value in the solution or target; previously deployed values can be updated. Seed them on the first deployment.

### 6.4.4 Component collections and multi-team delivery

A component collection is a reusable set of agent components (topics, knowledge, actions, and entities) shared among multiple agents in an environment. Exported through solutions, collections let different teams build parts of agents in different environments with independent release cadences. The documented pattern has IT and HR teams maintaining their collections in their own development environment, the product team building the agent in its development environment against the same collection versions, and test and production importing stable collection versions (for example IT collection v1.0 and HR collection v1.0). Best practices are reuse over duplication, version control for consistency and rollback, modular design, and independent release cadence.

### 6.4.5 Automating the Copilot Studio pipeline

The Copilot Studio ALM guidance compares three automation tools:

| Tool | Best for | Key capabilities | Setup complexity |
|---|---|---|---|
| **Pipelines in Power Platform** | Organizations empowering citizen developers | Centralized deployment visibility, pipeline and security management, quick setup | Low |
| **GitHub Actions for Power Platform** | Developer and admin teams managing multiple environments | Import and export solutions, deploy downstream, provision and deprovision environments, solution checker, backup, restore, copy, reset | Moderate |
| **Azure DevOps with Power Platform Build Tools** | Enterprise teams needing full ALM control | Azure Repos source control, Azure Pipelines CI/CD, Build Tools tasks, Dataverse Git integration with Azure Repos | High |

Pipelines in Power Platform have governance properties worth memorising. The same exported artefact passes through every stage in order, can't be tampered with, and isn't re-exported, so changes can't bypass QA. Managed and unmanaged copies are kept in the host for every deployment. Approvals use delegated deployments running as a service principal or pipeline owner, which then owns the deployed objects. Pipelines can't deploy across tenants, can't deploy unmanaged solutions, and deploy one solution per request; redeploying earlier versions is possible when that setting is on. Developers can run pipelines from the Power Platform CLI and extend them with Power Automate, Azure DevOps, or GitHub.



For fully automated pipelines, use a deployment settings file so that connection references and environment variables are populated without interactive prompts. Generate it with `pac solution create-settings`, fill in the target values, store it in source control, and pass it on import.

**Listing 6.3: Deployment settings file for the test environment**

```
{
  "EnvironmentVariables": [
    {
      "SchemaName": "contoso_KnowledgeSiteUrl",
      "Value": "https://contoso.sharepoint.com/sites/policies-test"
    },
    {
      "SchemaName": "contoso_EnvironmentType",
      "Value": "Test"
    }
  ],
  "ConnectionReferences": [
    {
      "LogicalName": "contoso_SharePointConnRef",
      "ConnectionId": "ef3d1cbb2c3b4e7987e02486584689d3",
      "ConnectorId": "/providers/Microsoft.PowerApps/apis/shared_sharepointonline"
    },
    {
      "LogicalName": "contoso_OrderApiConnRef",
      "ConnectionId": "4445162937b84457a3465d2f0c2cab7e",
      "ConnectorId": "/providers/Microsoft.PowerApps/apis/shared_contoso-5forderapi-
5f1234567890abcdef"
    }
  ]
}
```

**Listing 6.4: Azure Pipelines stage that checks and imports the agent solution with the settings file**

```
steps:
- task: microsoft-IsvExpTools.PowerPlatform-BuildTools.tool-installer.PowerPlatformToolInstaller@2
  displayName: 'Power Platform Tool Installer'

- task: microsoft-IsvExpTools.PowerPlatform-BuildTools.checker.PowerPlatformChecker@2
  displayName: 'Power Platform Checker'
  inputs:
    PowerPlatformSPN: 'Dataverse service connection'
    RuleSet: '0ad12346-e108-40b8-a956-9a8f95ea18c9'

- task: microsoft-IsvExpTools.PowerPlatform-BuildTools.import-
solution.PowerPlatformImportSolution@2
  displayName: 'Import agent solution to Test'
  inputs:
    authenticationType: PowerPlatformSPN
    PowerPlatformSPN: 'Contoso Test service connection'
    SolutionInputFile: '$(Pipeline.Workspace)/drop/ContosoAgents_managed.zip'
    AsyncOperation: true
    UseDeploymentSettingsFile: true
    DeploymentSettingsFile: '$(Pipeline.Workspace)/drop/config/test.json'
```

GitHub Actions for Power Platform offers the same building blocks as discrete actions (for example `export-solution`, `unpack-solution`, `pack-solution`, `check-solution`, `import-solution`). They support only Dataverse environments with a database, run on Windows or Linux runners, and should authenticate with a service principal because username and password authentication does not support multifactor authentication.

Listing 6.5: GitHub Actions job that exports a managed solution build artefact

```
jobs:
  export:
    runs-on: ubuntu-latest
    steps:
    - name: Install Power Platform Tools
      uses: microsoft/powerplatform-actions/actions-install@v1

    - name: Export managed solution
      uses: microsoft/powerplatform-actions/export-solution@v1
      with:
        environment-url: 'https://contoso-dev.crm.dynamics.com'
        app-id: ${{ vars.PP_APP_ID }}
        client-secret: ${{ secrets.PP_CLIENT_SECRET }}
        tenant-id: ${{ vars.PP_TENANT_ID }}
        solution-name: ContosoAgents
        solution-output-file: 'out/ContosoAgents_managed.zip'
        managed: true
```

### 6.4.6 Quality gates for Copilot Studio releases

Gates for Copilot Studio agents combine static checks, agent-specific review, and human approval:

- **Solution checker** through managed environment enforcement (Warn or Block) and as a pipeline step.
- **Agent Review Pipeline**, an automated gate for agents deployed through Pipelines in Power Platform. A pre-deployment step triggers a cloud flow in the pipelines host that dispatches a GitHub Actions workflow; it evaluates the agent in the solution (deterministic and AI-driven checks), scores it, produces a PDF report, and the flow approves or rejects the stage against a threshold.
- **Delegated deployment approvals** for production, with the deployment identity separated from the maker.
- **Model lifecycle gate.** Treat a model change as a migration: discover, inventory, evaluate against a baseline, approve, deploy through ALM, monitor. For high-risk and high-volume agents, select a specific model rather than the default, so upgrades don't arrive silently.

**PRODUCTION NOTE**

Keep a model and agent inventory: the Power Platform admin center shows each agent's model, the inventory API queries agents by model tenant-wide, and `pac copilot list` lists agents with solution context (without the model). Record owners, criticality, harness, and regression set so a retirement becomes a planned release.

## 6.5 Design the ALM process for Microsoft Foundry Agent Service

### 6.5.1 Agent types and versioning

Foundry Agent Service has three agent types: prompt-based agents (declarative: model, instructions, tools), voice-based prompt agents, and hosted agents (containerized or source-deployed code built with supported frameworks or custom code). The development lifecycle documented for Foundry is create, test, add tools and data, save versions, trace, evaluate, optimize, publish, and monitor.

Versioning is native and is the foundation of Foundry ALM:

- **Immutability.** Each saved version is immutable; any modification creates a new version.

- **Addressing.** In code you reference an agent as `<agent_name>:<version>`. The agent name cannot be changed after creation.
- **Comparison.** You can compare versions by setup, by chat output on identical inputs, and by YAML definition.
- **Routing.** Requests can be directed to specific versions, enabling controlled deployment and rollback.
- **Get-or-create.** Create each agent once and reuse its identifier; treat a 404 for a previously valid ID as terminal and recreate rather than retry.

For prompt agents defined in code, the SDK's `create_version` call creates the agent or a new version if it already exists, which makes it a natural CI step.

**Listing 6.6: Create a new prompt agent version from CI (Python, azure-ai-projects)**

```
import os
from azure.identity import DefaultAzureCredential
from azure.ai.projects import AIProjectClient
from azure.ai.projects.models import PromptAgentDefinition

endpoint = os.environ["FOUNDRY_PROJECT_ENDPOINT"]   # one project per environment

with (
    DefaultAzureCredential() as credential,
    AIProjectClient(endpoint=endpoint, credential=credential) as project_client,
):
    agent = project_client.agents.create_version(
        agent_name=os.environ["AZURE_AI_AGENT_NAME"],
        definition=PromptAgentDefinition(
            model=os.environ["AZURE_AI_MODEL_DEPLOYMENT_NAME"],
            instructions=open("agent/instructions.md", encoding="utf-8").read(),
        ),
    )
    print(f"Agent: {agent.name}, Version: {agent.version}")
```

**EXAM TIP**

Keep instructions in a versioned file in the repository and read them at deployment time, as in Listing 6.6. This puts prompts under pull-request review and ties every agent version to a commit.

### 6.5.2 Environments, infrastructure as code, and azd

Hosted-agent deployment with the Azure Developer CLI is documented end to end; the articles include items marked preview PREVIEW, so confirm each command's status before production use. Design rules:

- **One Foundry project per environment.** Create `azd` environments (development, test, production), each bound to a separate existing Foundry project with its own subscription, region, model deployments, and connections. Environment names alone don't isolate resources.
- **Infrastructure as code.** Initialize with `--infra=bicep` or `--infra=terraform`; commit IaC and the provider lockfile, never Terraform state, saved plans, or `.azure/` state. Use remote state with a key per environment and restrict access to it.
- **Separation of duties.** `azd provision` updates infrastructure; `azd deploy` builds and pushes the container (or uploads source) and creates a new agent version. Application-only releases run `azd deploy` alone.

- **Same source, separate packages.** Promote the same tested commit, but package and deploy per environment rather than sharing one build artefact.

`azd pipeline config` detects GitHub or Azure DevOps, creates a service principal, stores values as repository secrets or variables, and generates a workflow that signs in with OpenID Connect federated credentials.

**Listing 6.7: GitHub Actions workflow for a hosted agent (provision and deploy with azd)**

```
name: Azure Developer CLI
on:
  push:
    branches: [ main ]
  workflow_dispatch:

permissions:
  id-token: write
  contents: read

concurrency:
  group: hosted-agent-deployment
  cancel-in-progress: false

jobs:
  build:
    runs-on: ubuntu-latest
    env:
      AZURE_CLIENT_ID: ${{ vars.AZURE_CLIENT_ID }}
      AZURE_TENANT_ID: ${{ vars.AZURE_TENANT_ID }}
      AZURE_SUBSCRIPTION_ID: ${{ vars.AZURE_SUBSCRIPTION_ID }}
      AZURE_ENV_NAME: ${{ vars.AZURE_ENV_NAME }}
      AZURE_LOCATION: ${{ vars.AZURE_LOCATION }}
    steps:
      - uses: actions/checkout@v4
      - uses: Azure/setup-azd@v2
      - name: Install Foundry extensions
        run: azd ext install microsoft.foundry
      - name: Sign in to Azure (federated credentials)
        run: azd auth login --client-id $AZURE_CLIENT_ID --federated-credential-provider github --
tenant-id $AZURE_TENANT_ID
      - name: Provision infrastructure
        run: azd provision --no-prompt
      - name: Deploy agent
        run: azd deploy --no-prompt
```

**NOTE**

Always pass `--no-prompt` in CI so a missing value fails fast instead of hanging the job. A GitHub concurrency group serializes this workflow only; it is not a service-wide lock, so coordinate any other repositories or tools that deploy the same agent. CI provisioning can require the Foundry Owner role in addition to Contributor. The Foundry roles were renamed from Azure AI User, Azure AI Owner, Azure AI Account Owner, and Azure AI Project Manager to Foundry User, Foundry Owner, Foundry Account Owner, and Foundry Project Manager, with unchanged role IDs and permissions.

### 6.5.3 Evaluation gates

Foundry positions evaluation as the quality gate in the deployment pipeline and continuous evaluation as the production monitor. Establish a baseline during development and set acceptance thresholds before release (the documentation's example is an 85% task adherence passing rate). Two documented ways to gate CI:

- `azd ai agent eval run --no-prompt` runs a stored evaluation (from `eval.yaml` or a supplied config) against the current agent for regression checks; `azd ai agent eval generate` creates the suite and requires a deployed agent.
- The AI agent evaluation GitHub Action PREVIEW (and an equivalent Azure DevOps extension PREVIEW) invokes one or more agents, identified as `name:version`, with a dataset of queries, runs evaluators from the Foundry catalog (agent, RAG, risk and safety, general purpose, OpenAI graders, custom), and reports scores with confidence intervals and pairwise statistical comparison against a baseline agent version.

**Listing 6.8: Evaluation gate comparing the production version with the candidate**

```yaml
name: "AI Agent Evaluation"
on:
  workflow_dispatch:

permissions:
  id-token: write
  contents: read

jobs:
  evaluate-candidate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Azure login using federated credentials
        uses: azure/login@v2
        with:
          client-id: ${{ vars.AZURE_CLIENT_ID }}
          tenant-id: ${{ vars.AZURE_TENANT_ID }}
          subscription-id: ${{ vars.AZURE_SUBSCRIPTION_ID }}
      - name: Run evaluation
        uses: microsoft/ai-agent-evals@v3-beta
        with:
          azure-ai-project-endpoint: ${{ vars.FOUNDRY_PROJECT_ENDPOINT }}
          deployment-name: ${{ vars.EVAL_MODEL_DEPLOYMENT }}
          agent-ids: "order-agent:7,order-agent:8"
          baseline-agent-id: "order-agent:7"
          data-path: ${{ github.workspace }}/eval/order-agent-dataset.json
```

**Listing 6.9: Evaluation dataset file (stored with the agent source)**

```json
{
  "name": "order-agent-regression",
  "evaluators": [
    "builtin.task_adherence",
    "builtin.fluency",
    "builtin.violence"
  ],
  "data": [
    { "query": "What is the status of order 10452?" },
    { "query": "Change the delivery address on my last order." }
  ]
}
```

**EXAM TIP**

The documentation advises not to run evaluation on every commit, to control cost. Typical placements are on pull requests that change instructions, tools, or models, and before promotion to production. Remember that readiness is not correctness: `azd deploy` waits for version activation, but the smoke test must check expected response content, not just a non-empty reply.

### 6.5.4 Promotion and progressive rollout

By default a Foundry agent endpoint follows the latest version, so a pipeline that deploys and then tests can shift production traffic before the test runs. For gated releases, pin production to its existing version before deployment, capture the version the deployment creates, validate that exact candidate in a fresh version-pinned session, and promote only after approval. Canary rollout uses fixed-ratio version selection rules on the agent endpoint; endpoint updates change routing without creating another agent version.

**Listing 6.10: Canary routing in azure.yaml (90% stable version 1, 10% candidate version 2)**

```
agentEndpoint:
  versionSelector:
    versionSelectionRules:
      - type: FixedRatio
        agentVersion: "1"
        trafficPercentage: 90
      - type: FixedRatio
        agentVersion: "2"
        trafficPercentage: 10
```

Apply the rule with `azd ai agent endpoint update my-agent -e production --no-prompt`, verify with `azd ai agent endpoint show`, and test the candidate with `azd ai agent invoke` using `--version 2 --new-session --new-conversation`. Increase the share in steps (for example 90/10, then 50/50), keeping the total at 100%. To roll back, set the stable version back to 100%; no rebuild occurs. Save the final routing configuration with the release records so a later run does not reapply an outdated split.




Use dependent jobs (GitHub Actions) or dependent stages with deployment jobs (Azure DevOps) for development, test, and production; scope variables, secrets, and service connections to each environment; and configure environment protection rules or approvals and checks on production.

### 6.5.5 Identity, permissions, and publishing

Treat agent configuration like application code: least privilege through role assignments, secrets in a managed store referenced through connections (for example a Key Vault connection), never in code, configuration, or prompts. Each hosted agent version gets a dedicated Microsoft Entra agent identity when it is created. Publishing an agent as an agent application gives it a stable endpoint, and permissions assigned to the project

identity don't automatically transfer to the published agent: reassign the required privileges to the agent application's identity and remove any access it no longer needs. Make this a scripted post-deployment step in each environment rather than a manual fix after users report failures.

**COMMON TRAP**

A tool that worked in the playground fails after publishing. The usual cause is identity: the published agent no longer runs with the project identity's permissions. The fix is to grant the agent application's identity the required role on the downstream resource, not to embed a key.

## 6.6 Design the ALM process for custom AI models

### 6.6.1 Which custom models need their own lifecycle

"Custom AI model" covers several artefacts with very different transport rules. The architect's first decision is to identify which kind is in scope.

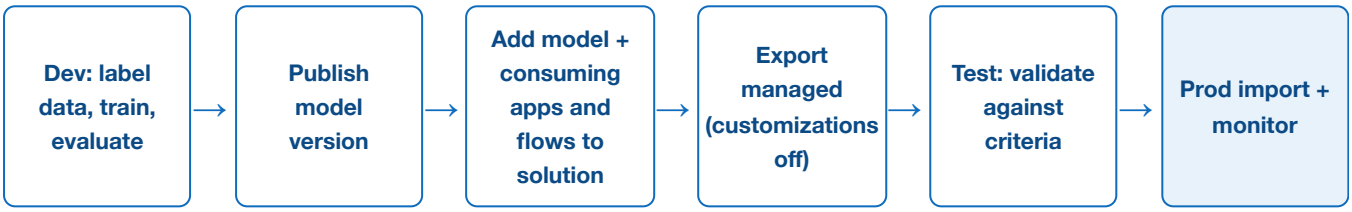
| Model artefact | Where it lives | How it moves between environments | What does not move |
|---|---|---|---|
| **AI Builder custom model (for example document processing, object detection, category classification)** | Dataverse environment | Solution component; only the published version is included | Training data; some models cannot be retrained after import |
| **Prompt (prompt builder, used in Copilot Studio and Power Platform)** | Dataverse environment | Solution component, added through **Add existing** then **AI models** | Nothing extra, but editing after import creates unmanaged customizations |
| **Fine-tuned model in Foundry** | Foundry resource and project | Deploy the fine-tuned model (or a checkpoint) to a serving deployment per environment; evaluation and deployment scripted in MLOps pipelines | Deployments are per resource; inactive deployments are deleted |
| **Prebuilt models and platform models** | Microsoft-managed | Nothing to transport; select and configure per environment | Version changes are driven by Microsoft's model lifecycle |
| **Dynamics 365 Finance insights models (for example customer payment predictions)** | Dataverse (AI Builder based) | Train in a sandbox with recent production data, then deploy with managed solutions | Training data |

### 6.6.2 AI Builder models and prompts in solutions

All AI Builder environments need Dataverse. Model states (draft, training, trained, publishing, published, unpublishing, training error, importing, import error) are stored in the AI Configuration table. Develop in a sandbox or development environment, qualify in test, and deploy to production with managed solutions. Documented constraints:

- Only a model with a published version can be added to a solution, and only that version is installed in the target.
- The solution carries the model executable, not the training data; imported document processing, object detection, and entity extraction models can't be retrained.
- Add consuming apps and flows and the model explicitly; the model isn't treated as their dependency.
- Editing an imported model creates unmanaged customizations that can block updates; turn off **Allow customizations** in managed properties before export.
- Import object detection and document processing models within one month of export (unless the source is unchanged); they can show **Importing** for several minutes.
- SharePoint and Syntex document processing models don't support environment-to-environment transport through Dataverse solutions.

Prompts follow the same pattern: develop in a sandbox, deploy managed, and avoid edits after import.




### 6.6.3 Fine-tuned models in Microsoft Foundry

A fine-tuned model must be deployed before it can be called. Design for these documented behaviours:

- **Deployment options.** Serverless, Managed Compute PREVIEW (LoRA adapters on a compatible base deployment), and Fireworks on Foundry PREVIEW. Deployment needs Foundry Owner (or deployment write permission) and quota.
- **Checkpoints.** Deploy the final model or a checkpoint, so you can promote the best-evaluated one.
- **Inactivity deletion.** A customized model deployment with no calls for a continuous 15-day period (after being deployed more than 15 days) is deleted. The model itself remains and can be redeployed, so pipelines should recreate deployments idempotently, especially in low-traffic test environments.
- **Hosting cost.** Deployed fine-tuned models incur hourly hosting cost even when idle; remove unneeded deployments.
- **Evaluation.** Test the fine-tuned deployment (not the base model), then run repeatable evaluations and compare candidates before promotion.

> **COMMON TRAP**
> "The fine-tuned model disappeared from test" usually means the deployment was deleted after 15 idle days. Redeploy; don't repeat the fine-tuning job.

### 6.6.4 MLOps, GenAIOps, and the model change process

Architecture Center guidance expects model updates, through retirement or for better features, price, or performance. Prompts and hyperparameters are designed for specific model versions, so they must always ship with the matching model. It recommends three automated pipeline families: MLOps for fine-tuning, GenAIOps for changes to models, hyperparameters, prompts, and orchestration, and DataOps for changes to RAG grounding data. GenAIOps adds inner-loop stages (DataOps, experimentation, evaluation) and outer-loop

stages (deployment, monitoring, feedback), optionally with human-in-the-loop validation before deployment. Teams with MLOps can reuse it for fine-tuning and extend it for prompts and RAG, with separate maturity criteria for data and prompt versioning, evaluation, deployment automation, monitoring, and content safety.

| Change | Gate before production |
|---|---|
| **New base model or model version for an agent** | Regression evaluation against the stored baseline, instruction and tool refinement, approval, then deploy through ALM and monitor |
| **New fine-tuned model or checkpoint** | Offline evaluation with task metrics and safety evaluators; compare with the current production model; validation criteria agreed with the business |
| **Prompt or instruction change** | Pull request review plus evaluation run on the regression dataset |
| **Grounding data or chunking change** | RAG evaluators (retrieval and groundedness) on the updated index in test |

**EXAM TIP**

Use-category mismatch is a documented upgrade mistake in Copilot Studio: moving a high-volume FAQ agent from a *general* model to a *deep* model can raise latency and credit consumption sharply for a marginal quality gain. When an item asks which model change to approve, weigh quality, latency, and cost together, not the evaluation score alone.

## 6.7 Design the ALM process for AI in Dynamics 365 apps for finance and supply chain

### 6.7.1 From Lifecycle Services to unified environments

Finance and operations apps are moving administration from Microsoft Dynamics Lifecycle Services to the Power Platform admin center. Environments managed there are called **unified environments** because they combine Dynamics 365 ERP workloads, Dynamics 365 customer engagement apps, and low-code artefacts in one Power Platform environment backed by Dataverse, with no Lifecycle Services footprint. The ALM implications are significant:

| Aspect | Lifecycle Services | Power Platform admin center (unified) |
|---|---|---|
| Customization deployment | Deployable packages uploaded to the Asset library, installed in a sandbox; production updated from the sandbox through the Lifecycle Services UI, with no automation for the production deployment | Unified packages that contain Dataverse solutions and X++ modules, built and installed with Power Platform CLI or Azure DevOps tasks in any target (sandbox or production); full automation with service principal authentication |
| Copy | Database refresh copies data only | Copy includes code and data, giving an exact replica |
| Backup | Database export | Backup kept in the cloud, never downloaded as a .bak or .bacpac |
| Convert sandbox to production | Not applicable | Supported for unified environments |

| Aspect | Lifecycle Services | Power Platform admin center (unified) |
|---|---|---|
| Monitoring | Per environment tools with at least 15 minutes of delay | Application Insights is the preferred tool for near-real-time health |

### 6.7.2 Building and deploying unified packages

The unified developer experience uses Visual Studio for X++ development and Dataverse for low-code assets. In Azure Pipelines, the **Create Deployable Package** task (version 3) can generate both the Lifecycle Services format and the Power Platform unified package format, and the Power Platform Build Tools deploy the result. Existing Lifecycle Services packages can be converted to the unified format with `ModelUtil.exe -convertToUnifiedPackage`. The Power Platform CLI also exposes ERP-aware package commands PREVIEW.

**Listing 6.11: Deploy a unified finance and operations package with pac CLI**

```
# Authenticate to the unified sandbox environment
pac auth create --environment https://contoso-fo-test.crm.dynamics.com

# Deploy the package (Dataverse solutions and X++ modules); runs on the .NET Full Framework pac CLI
pac package deploy --package .\out\ContosoUnifiedPackage.zip --logFile .\logs\deploy.log --
logConsole

# ERP-specific options (Preview): classify as a release package, which forces a full database sync
pac package deploy --package-type erp --package .\out\ContosoErpModels.zip --release-type Release
```

> **NOTE**
> Release packages force a full database synchronization on the server. For troubleshooting, operation logs are available in the Finance and Operation Package Manager app (Operation History) in the Dataverse organization. Unified developer environments also support code-only package deployment in administration mode, which skips database synchronization, as a recovery path.

### 6.7.3 Copilot prerequisites and feature enablement per environment

Copilot in finance and operations apps combines UX components, application code, and Azure-hosted AI services reached through Dataverse, delivered as sidecar features (the chat pane) and embedded summary features. Every target environment must meet the prerequisites:

- **Power Platform integration** must be enabled. With it, the required Copilot features are usually deployed automatically.
- **Required Dynamics 365 apps** in the Power Platform environment: *Copilot for finance and operations apps* (generative help and guidance and the sidecar bot), *Copilot in Dynamics 365 Supply Chain Management* (summaries in Supply Chain Management and Commerce), and *Copilot in Dynamics 365 Finance* (summaries in Finance). Check and install them under the environment's Dynamics 365 apps in the Power Platform admin center.
- **Sidecar dependencies**: Bing Search for generative answers, cross-region data movement where back-office AI services are not in region, and the tenant setting that allows publishing bots with AI features.
- **Feature-specific setup**: for example generative help and guidance, Immersive Home (feature management), Finance and Supply Chain Management summaries, and the Procurement Agent (production-ready preview) setup wizards.

**COMMON TRAP**

Copilot and AI agents aren't supported in developer environments deployed through Lifecycle Services. Build and test Copilot extensions in a unified developer environment from the Power Platform admin center.

### 6.7.4 Extending Copilot in finance and operations apps: a two-track release

Extensions to the finance and operations Copilot span two lifecycles that must be released together:

- **X++ track.** Client plugins (client actions) expose X++ business logic to Copilot. You create an X++ class, decorated with the action definition attribute, in a unified developer environment. It travels in the unified package.
- **Copilot Studio track.** In the Dataverse environment linked to the finance and operations environment, the agent named *Copilot for finance and operations apps* is extended with topics. For a client action, one topic uses an **Event activity** whose name matches the X++ action identifier, and another topic with an **Event received** trigger handles the response. These topics are Dataverse components and travel in a solution.

Generative help and guidance knowledge lives in the same agent: upload files (PDF, RTF, Word) on the Knowledge tab, then test and publish. Other knowledge types such as SharePoint require your own topic. AI general knowledge is off by default and needs careful testing. Virtual entities from finance and operations apps aren't supported as knowledge sources.




**PRODUCTION NOTE**

Release the X++ package before or with the Copilot Studio solution; a topic calling a missing action fails at runtime. Track both under one release and change record.

## 6.8 Design the ALM process for AI in Dynamics 365 apps for customer experience and service

### 6.8.1 Customizations travel in solutions; Copilot switches often do not

Sales, Customer Service, and Contact Center customizations follow standard Dataverse solution ALM. The AI difference is that many Copilot capabilities are environment-level admin settings in the Power Platform admin center and the app's admin center. Unless a setting is documented as a solution component, treat it as part of the environment build and post-deployment runbook, reviewed like code and verified per environment.

Dataverse **settings** are solution components with three parts: a setting definition (default value and the levels at which it can be overridden), a setting environment value, and a setting app value. You can include the definition and the environment value in a solution, which makes them useful for custom feature flags that switch agent experiences on or off.

### 6.8.2 Copilot feature settings across environments

| App | Setting | Where | Scope |
|---|---|---|---|
| **Dynamics 365 Sales** | AI capabilities: **Copilot** (Copilot chat and skills such as lead and opportunity summaries) and **AI Agents** (for example Sales Qualification Agent and Sales Close Agent) | Power Platform admin center, Copilot, Settings, Dynamics 365 Sales | Per environment or per environment group; on by default |
| **Dynamics 365 Sales** | Turn Copilot on or off in the sales app; consent to cross-region data movement where needed | Sales Hub app settings and Power Platform admin center | Per environment and app |
| **Dynamics 365 Customer Service** | **Enable AI-powered Copilot features** (required for Customer Service Hub and custom apps) | Power Platform admin center, Copilot, Settings, Dynamics 365 Customer Service | Per environment |
| **Dynamics 365 Customer Service** | Opt in, choose Copilot immersive or side pane, configure the Customer Support agent (overview, immersive settings, email settings, prompts, extend agent), Admin Management agent PREVIEW, translation, transcript recording | Copilot Service admin center | Per environment |
| **Dynamics 365 Customer Service** | Which representatives get which Copilot features in Copilot Service workspace | Experience profiles | Per user group |
| **All** | Data movement across regions; optional data sharing with Microsoft | Power Platform admin center | Per environment (tenant settings for data sharing) |
| **All** | Custom security roles with Copilot privileges (for example `prvIntelligenceUsage` and Copilot tables) | Solution (security roles are solution components) | Per role |

> **NOTE**
> In Customer Service, opting out of Copilot features erases the training data, and opting back in requires consenting to the terms again. Do not toggle opt-in casually in a shared test environment that other teams depend on.

> **EXAM TIP**
> Because Sales AI capabilities can be set per environment group, a clean design is to put development, test, and production environments into stage-specific groups and control Copilot and AI agent availability at group level. This also keeps Copilot settings aligned with the external model and preview model decisions made in 6.2.4.

### 6.8.3 Validation after each promotion

The documented Customer Service validation checklist maps to a post-deployment test stage: confirm the environment switch, opt-in and data movement status, selected Copilot experiences, and experience profile membership; then sign in as a test representative, use a feature such as case summary, and repeat for Customer Service Hub, custom apps, and custom security roles. Automate what you can and keep the rest as a signed checklist.




### 6.8.4 Agents that extend Dynamics 365 CE

When you extend Copilot in these apps with Copilot Studio (for example through the Customer Support agent's **Extend agent** tab, or a custom agent for a Contact Center channel), the rules from 6.4 apply in full. Develop in a sandbox with the Dynamics 365 apps installed, and split customizations and agent extensions into separate solutions only if they need independent deployment.

## 6.9 Key terms

| | |
|---|---|
| **Managed solution** | The deployed form of a Dataverse solution; components can't be edited directly, and uninstalling removes them. |
| **Connection reference** | A solution component that points to a connection, so each environment binds its own connection. |
| **Environment variable** | A solution component holding environment-specific values; supports secret values stored in Azure Key Vault. |
| **Deployment settings file** | A JSON file with connection reference and environment variable values passed on import for unattended deployments. |
| **Pipelines in Power Platform** | In-product CI/CD with a host environment, stage approvals through delegated deployments, and managed environment targets. |
| **Component collection** | A reusable set of Copilot Studio topics, knowledge, actions, and entities that can be shared across agents and moved through solutions. |
| **Harness** | The Copilot Studio runtime between an agent design and its model: standard, GitHub Copilot, or Copilot chat. |
| **Agent version (Foundry)** | An immutable snapshot of a Foundry agent, referenced as `name:version`. |
| **GenAIOps** | Operational practices that extend MLOps to prompts, orchestration, RAG data, evaluation, and monitoring of generative AI. |
| **Unified environment** | A Power Platform environment with Dataverse hosting finance and operations apps, managed in the Power Platform admin center. |
| **Environment group** | A set of managed environments governed by shared rules that lock the corresponding environment settings. |

## 6.10 Additional practice questions

**Practice 6.1** MULTIPLE SELECT

Fabrikam deploys a Copilot Studio agent from test to production by importing a managed solution. Import succeeds, but the agent is not reachable on the company website and Application Insights shows no telemetry from production. You need to identify the items that must be configured in the production environment after import because they are not transported by the solution. Which three items should you identify? Each correct answer presents part of the solution.

A. Agent topics and instructions
B. Direct Line and web channel security settings
C. Azure Application Insights settings
D. Environment variable definitions
E. Deployed channels

**Practice 6.2** MULTIPLE CHOICE

Contoso uses Azure DevOps to deploy a solution that contains an agent, two agent flows, three connection references, and four environment variables. Imports to test currently stop for manual input. You need a fully unattended deployment to test and production with environment-specific values kept in source control. What should you do?

A. Export the solution as unmanaged and set the current values in the target after import.
B. Generate a deployment settings file with `pac solution create-settings`, populate it per environment, and enable **Use deployment settings file** on the import task.
C. Store the connection IDs as environment variables of type text in the solution.
D. Use the ALM Accelerator deployment configuration files.

**Practice 6.3** MULTIPLE CHOICE

A business unit has 40 trained makers building Copilot Studio agents in Zone 2. IT requires that no maker deploys directly to production, that production deployments run under a non-personal identity after approval, and that setup effort is minimal. What should you recommend?

A. GitHub Actions for Power Platform with a manual approval job
B. Pipelines in Power Platform with delegated deployments to managed target environments
C. Manual export and import by the CoE team
D. Dataverse Git integration connected to the production environment

**Practice 6.4** MULTIPLE CHOICE

A hosted agent in Foundry Agent Service runs version 4 in production. A pipeline will deploy a new version from an approved commit. Business owners require that no production user reaches the new version until it passes a smoke test in production, followed by a gradual rollout. What should the pipeline do first?

A. Run `azd provision` to create a new production project.
B. Set version selection rules so version 4 receives 100% of traffic, apply the endpoint update, then deploy and test the candidate with a version-pinned invocation.
C. Deploy, then immediately run the smoke test against the default endpoint.
D. Delete version 4 so that the endpoint has a single version.

**Practice 6.5** MULTIPLE SELECT

You design CI/CD for a Foundry prompt agent used in three stages. Auditors require that each stage is isolated and that no change reaches production unless it scores at least as well as the current version on a regression dataset. Which two actions should you include? Each correct answer presents part of the solution.

A. Use one Foundry project and separate environments by agent name suffix.
B. Bind development, test, and production to separate Foundry projects through separate azd environments.
C. Run an evaluation step that compares the candidate `name:version` with the production version as baseline on a versioned dataset before promotion.
D. Run the evaluation on every commit to every branch.
E. Rely on `azd deploy` version activation as the quality signal.

**Practice 6.6** MULTIPLE CHOICE

An AI Builder document processing model was imported into production in a managed solution. Accuracy has dropped on a new invoice layout, and the operations team asks to add samples and retrain the model in production. What should you recommend?

A. Retrain the imported model in production and republish it.
B. Retrain and publish the model in development with the new samples, export a new managed solution version, and import it into production.
C. Export the managed solution from production, edit it, and reimport it.
D. Copy the training data into the production solution.

### Practice 6.7 MULTIPLE CHOICE

A fine-tuned model was deployed in the test Foundry resource five weeks ago. Testing paused for three weeks. Today the deployment is missing, but the fine-tuned model is still listed. You need to restore testing with minimum cost and effort and prevent recurrence. What should you do?

A. Repeat the fine-tuning job and redeploy the result.
B. Redeploy the existing fine-tuned model from the pipeline, and make the pipeline recreate the deployment when it is absent.
C. Open a support case to restore the deleted model.
D. Move the model to the production resource.

### Practice 6.8 MULTIPLE CHOICE

A partner develops client plugins for Copilot in Dynamics 365 Supply Chain Management on a cloud-hosted developer environment deployed through Lifecycle Services. The Copilot sidecar does not appear in that environment. You need the team to build and test the extension end to end. What should you recommend?

A. Enable dual-write in the developer environment.
B. Use a unified developer environment provisioned in the Power Platform admin center.
C. Install Copilot Studio on the developer VM.
D. Turn on Immersive Home in feature management.

### Practice 6.9 MULTIPLE CHOICE

Finance users want generative help and guidance in Dynamics 365 Finance to answer from internal procedures stored in a SharePoint site. The change must be promoted from test to production. What should you do in the test environment first?

A. Add the SharePoint site as a Dataverse virtual entity knowledge source.
B. In Copilot Studio, in the Dataverse environment linked to the finance and operations environment, add your own topic to the Copilot for finance and operations apps agent that uses the SharePoint knowledge, then test and publish.
C. Turn on AI general knowledge for the agent.
D. Upload the SharePoint site URL on the Knowledge tab as a file.

### Practice 6.10 MULTIPLE SELECT

After deploying a managed solution with Customer Service customizations to a new UAT environment, representatives using Customer Service Hub and Copilot Service workspace can't use Copilot. You need Copilot to work for the UAT representatives. Which three actions should you perform? Each correct answer presents part of the solution.

A. Select **Enable AI-powered Copilot features** for the UAT environment in the Power Platform admin center.
B. Opt in to Copilot features in the Copilot Service admin center after checking data movement status.
C. Assign the representatives to an experience profile that has Copilot features turned on.
D. Reimport the solution as unmanaged.
E. Create a Fabric deployment pipeline for the UAT workspace.

**Practice 6.11** MULTIPLE CHOICE

A Fabric data agent grounded on a lakehouse is promoted from the development workspace to the production workspace with a Fabric deployment pipeline. Deployment succeeds, but the agent in production returns no results. What is the most likely cause?

A. Data agents aren't supported in deployment pipelines.
B. Deployment pipelines copy item metadata but not data, so the production lakehouse hasn't been loaded.
C. The data agent published folder was edited directly.
D. The pipeline has more than three stages.

**Practice 6.12** MULTIPLE CHOICE

An innovation team wants to evaluate an Anthropic model as the primary model for a Copilot Studio agent in its test environments. Production must remain limited to Microsoft-hosted models until a risk review completes. Test environments are in one environment group and production environments in another. What should you do?

A. Turn on preview and experimental AI models for the test group.
B. Turn on external models for the test environment group in the Power Platform admin center, and allow the Anthropic provider in the Microsoft 365 admin center.
C. Turn on external models in each production environment and ask makers not to select them.
D. Allow the Anthropic provider in the Microsoft 365 admin center only.

## Answers and explanations

**6.1: B, C, E.** Copilot Studio guidance lists Application Insights settings, manual authentication, Direct Line and web channel security, deployed channels, and sharing as not solution-aware. Topics, instructions, and environment variable definitions travel in the solution.

**6.2: B.** A deployment settings file created with `pac solution create-settings` pre-populates connection references and environment variables for unattended imports. Unmanaged deployment downstream breaks ALM rules, connection IDs are not environment variables, and the ALM Accelerator is deprecated.

**6.3: B.** Pipelines in Power Platform are designed for citizen developers with low setup effort; delegated deployments add approval and run as a service principal or pipeline owner. Targets must be managed environments. Git integration is for developer environments, not production.

**6.4: B.** A Foundry endpoint follows the latest version by default. Pin the stable version to 100% before deploying, then validate the candidate with `--version` and increase traffic in steps. Provisioning a new project or deleting the stable version removes the rollback path.

**6.5: B, C.** Each environment should be bound to its own Foundry project because names don't isolate resources. The evaluation gate compares the candidate with the production baseline on a versioned dataset. Evaluating every commit is discouraged for cost, and activation isn't a quality signal.

**6.6: B.** Imported document processing models can't be retrained because the training data isn't in the solution, and changing imported models creates unmanaged customizations. Retrain in development, publish, export a new version, and import.

**6.7: B.** Deployments of customized models are deleted after 15 days of inactivity, but the model remains and can be redeployed. Retraining wastes cost; an idempotent deployment step prevents recurrence.

**6.8: B.** Copilot and AI agents aren't supported in developer environments deployed through Lifecycle Services. Unified developer environments in the Power Platform admin center support the integrated Dataverse and Copilot Studio experience.

**6.9: B.** Uploading files covers file knowledge; other knowledge types such as SharePoint require your own topic in the Copilot for finance and operations apps agent. Virtual entities aren't supported as knowledge sources, and general knowledge doesn't add SharePoint content.

**6.10: A, B, C.** Customer Service Hub and custom apps need the environment-level switch in the Power Platform admin center; Copilot Service workspace needs opt-in and experience profiles with Copilot features. These are environment settings, not solution content.

**6.11: B.** Fabric deployment pipelines copy metadata; data isn't copied. The production lakehouse needs its own data loading. Data agents are supported in deployment pipelines.

**6.12: B.** External models need two independent actions: enable external models for the environment or environment group in the Power Platform admin center, and allow each provider in the Microsoft 365 admin center. Preview model settings are separate. Scoping to the test group leaves production blocked by policy rather than by maker discipline.

## Key takeaways

- One release train, separate lanes for definitions, models, data, settings, and evaluation evidence, all under one gate policy.
- Unmanaged in development and source control, managed elsewhere; custom publisher, environment variables (Key Vault secrets), connection references.
- Pipelines in Power Platform for governed maker deployments; Azure DevOps or GitHub Actions for enterprise control and cross-tenant needs.
- Solutions and Fabric pipelines don't move data: plan `pac data` jobs, index rebuilds, and data loads per environment.
- Copilot Studio post-import: authentication, channels, Direct Line security, Application Insights, sharing, publishing.
- Foundry: immutable `name:version`, one project per environment, IaC with azd, evaluation gates, pinning, canary, rollback.
- Custom models: AI Builder models move without training data; idle fine-tuned deployments are deleted but models remain; every model change is an evaluated migration.
- Finance and operations: unified packages automate deployment; Copilot needs Power Platform integration and required apps; extensions ship as X++ plus Copilot Studio solution.
- Customer experience and service: customizations travel in solutions; Copilot switches, opt-in, and experience profiles are per-environment settings.
- Third-party AI is an environment decision: external and preview models are enabled per environment or group, external providers also in the Microsoft 365 admin center.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Solution Architect series: Plan application lifecycle management for Power Platform
- Establish an application lifecycle management strategy (Copilot Studio)
- Export and import agents using solutions
- Create and manage solutions in Copilot Studio

- Implement a zoned governance strategy
- Manage the AI model lifecycle for Copilot Studio agents
- Use the Agent Review Pipeline as a CI/CD gate
- Harnesses in Copilot Studio
- Add SharePoint as a knowledge source
- Extend your agent with Model Context Protocol
- Choose an external model as the primary AI model
- Distribute your prompt using a solution
- Solution concepts
- Overview of pipelines in Power Platform
- Overview of Git integration in Power Platform
- Pre-populate connection references and environment variables for automated deployments
- Microsoft Power Platform Build Tools tasks
- GitHub Actions for Microsoft Power Platform
- Available GitHub Actions for Microsoft Power Platform development
- pac solution
- pac data
- pac package
- Move configuration data across environments and organizations with the Configuration Migration tool
- Develop a tenant environment strategy to adopt Power Platform at scale
- Allow external language models for generative responses
- Environment variables for Power Platform overview
- Use settings to provide customized app experiences
- Distribute your model using a solution (AI Builder)
- Administer AI Builder
- Agents in Microsoft Foundry
- Agent development lifecycle
- Set up CI/CD for hosted agents with the Azure Developer CLI
- Promote hosted agents to production with the Azure Developer CLI
- Agent runtime components (create an agent with the SDK)
- How to run an evaluation in GitHub Action
- Evaluate your AI agents
- Deploy fine-tuned models in Microsoft Foundry
- Customize a model with fine-tuning
- Generative AI operations for organizations with MLOps investments
- Design to support foundation model life cycles
- Introduction to deployment pipelines (Fabric)
- The deployment pipelines process (Fabric)
- Source control, CI/CD, and ALM for Fabric data agent
- Overview of unified admin experience for finance and operations apps
- Administration for partners and systems implementers
- Continuous integration and deployment (unified developer experience)
- Enable Copilot capabilities in finance and operations apps
- Add knowledge to generative help and guidance with Copilot
- Create client plugins for Copilot in finance and operations apps
- Model management lifecycle (Finance insights)
- Manage Copilot features in Customer Service
- Track, manage, and scale Copilot adoption in the Power Platform
- Turn on and set up Copilot in Dynamics 365 Sales

DOMAIN 3 · PART C

# Responsible AI, security, governance, risk, and compliance

**Part of Domain 3: Deploy AI-powered business solutions (40–45%)**

This chapter covers the skill group *Design responsible AI, security, governance, risk management, and compliance*. It is the last part of the heaviest domain on the exam, and it is where the architect role is most visible: you are rarely asked how to click through a single setting, and often asked which control, on which platform, at which layer, closes a specific risk with the least collateral impact. Expect scenario stems that mix Microsoft 365 Copilot, Copilot Studio, Microsoft Foundry and Dynamics 365 in one solution, and that ask you to pick the control that belongs to the right administrative plane (Microsoft 365 admin center, Power Platform admin center, Microsoft Entra admin center, Microsoft Purview portal, Azure portal or the Foundry portal).

Study approach: first learn the **layers** (identity, data, tools and connectors, model, network, monitoring), then learn which product owns each layer on each platform, then learn the threat catalogue (direct and indirect prompt injection, data exfiltration, oversharing, excessive agency) and map each threat to a preventive and a detective control. Finish with data residency and audit trails, which are heavy on precise facts (what moves, where it is stored, which log records which change, for how long).

| Official skill (as of October 14, 2026) | Covered in |
|---|---|
| Design security for agents | 7.1 |
| Design governance for agents | 7.2 |
| Design model security | 7.3 |
| Analyze solution and AI vulnerabilities and mitigations, including prompt manipulation | 7.4 |
| Review solution for adherence to responsible AI principles | 7.5 |
| Validate data residency and movement compliance | 7.6 |
| Design access controls on grounding data and model tuning | 7.7 |
| Design audit trails for changes to models and data | 7.8 |

| | |
|---|---|
| **Agent identity** | A special service principal in Microsoft Entra ID, created from an agent identity blueprint, that represents an AI agent. It has no credentials of its own; the blueprint acquires tokens on its behalf. |
| **Agent identity blueprint** | A reusable template that records metadata shared by all agent identities of one kind and holds the credentials (federated credentials, certificates or secrets) used to obtain tokens for those identities. |
| **Sponsor** | The human user or group accountable for an agent identity. If a sponsor leaves, sponsorship transfers to their manager. |
| **Data policy (DLP)** | A Power Platform policy that classifies or blocks connectors, including the Copilot Studio feature connectors that govern authentication, knowledge sources, HTTP, skills, channels |

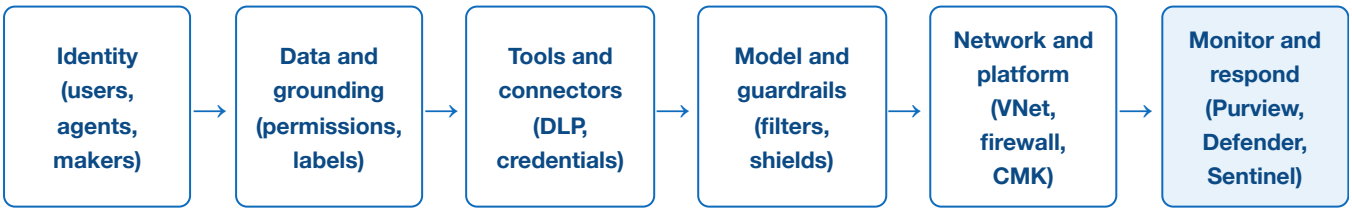
| | |
|---|---|
| | and triggers. |
| **UPIA / XPIA** | User prompt injection attack (direct, typed by the user, including jailbreaks) and cross-domain prompt injection attack (indirect, instructions hidden in documents, emails, web pages or tool output). |
| **Guardrail (Foundry)** | A named collection of controls; each control defines a risk, the intervention points to scan and the action to take. |
| **Data Zone deployment** | A Foundry model deployment type that processes prompts and responses only within a Microsoft-defined data zone (United States or European Union). |
| **Flex routing** | An EU Data Boundary option that allows LLM inferencing outside the boundary during peak load to preserve the Copilot experience. |
| **Restricted Content Discovery (RCD)** | A SharePoint Advanced Management setting that stops a site from surfacing in organisation-wide search and Copilot answers unless the user owns or recently interacted with the content, without changing site permissions. |

# 7.1 Design security for agents

## 7.1.1 The agent attack surface and the shared responsibility model

An agent is more than a chat front end on a model. It has an identity, it reads grounding data, it calls tools (connectors, MCP servers, APIs, flows, other agents), and it may act autonomously on triggers. Each of these is an attack surface. Microsoft's AI agent shared responsibility guidance stresses the *action* dimension that is unique to agents: an attacker no longer needs to make a model say something harmful when they can make it *do* something harmful.

The responsibility split depends on the platform. With Microsoft 365 Copilot and prebuilt Dynamics 365 Copilot features, Microsoft runs the orchestration, the model and most safety systems, and you own identity, data permissions, labels and tenant configuration. With Copilot Studio, you also own agent design (authentication mode, knowledge, tools, credentials, sharing, channels). With Microsoft Foundry, you own the most: model choice and deployment type, guardrail configuration, network isolation, RBAC, keys or managed identities, and the application code. The more you build, the more you secure.




Use these six layers as the skeleton for every security design answer in this domain. A good exam answer usually adds the control at the lowest layer that fully closes the risk (for example, fixing permissions at the data layer rather than hoping a content filter catches a leak).

## 7.1.2 End-user authentication for Copilot Studio agents

Copilot Studio offers three user authentication options under **Settings > Security > Authentication**:

| Option | Behaviour | Use when | Watch-outs |
|---|---|---|---|
| **No authentication** | Anyone with the link can chat. The agent can only access public information and resources. | Public website FAQ or anonymous self-service. | Cannot use user-scoped knowledge or tools. Flagged by the automatic security scan. Blocked when a data policy requires authentication. |
| **Authenticate with Microsoft (default for new agents)** | Integrated Microsoft Entra ID sign-in with single sign-on in Teams, SharePoint, Power Apps and Microsoft Copilot. | Internal employee agents in Microsoft clients. | `User.AccessToken` and `User.IsLoggedIn` are not available; use manual authentication if topics need the token. |
| **Authenticate manually** | Maker configures the identity provider: Microsoft Entra ID V2 with federated credentials, with certificates or with client secrets, Microsoft Entra ID, or Generic OAuth 2. | Custom websites or apps, non-Microsoft identity providers, or when topics or HTTP calls need the user token. | Prefer federated credentials or certificates over client secrets. Generic OAuth 2 can be blocked by admin policy. |

Administrators can enforce the choice centrally in two ways. The established approach is a data policy that blocks the connector **Chat without Microsoft Entra ID authentication in Copilot Studio**; after that, makers can only use *Authenticate with Microsoft* or *Authenticate manually*, and agents that violate the policy cannot be published. The newer Power Platform admin center setting **Configure authentication for agents** PREVIEW maps tenant choices to maker options: *No authentication* (makers can choose anything, including anonymous), *Require Microsoft authentication* (only Authenticate with Microsoft) and *Require Entra authentication* (Microsoft authentication or manual Entra ID; blocks no authentication and Generic OAuth 2).

> **COMMON TRAP**
> Sharing limits and Conditional Access do not stop a maker from publishing an anonymous agent to a public website. The control that prevents unauthenticated agents is the data policy on the *Chat without Microsoft Entra ID authentication* connector (or the authentication admin control). Conditional Access only evaluates sign-ins that actually happen.

### 7.1.3 Agent identities with Microsoft Entra Agent ID

Historically, agents borrowed identities: a maker's connection, an app registration or a service principal shared by many tools. Microsoft Entra Agent ID introduces purpose-built identity objects for agents so they can be authenticated, authorised, governed and protected like other identities, but with controls designed for non-human actors. The key objects are:

- **Agent identity blueprint**: the template and credential holder. Blueprints can be multitenant; the agent identities they create are always single-tenant and can only get tokens in their home tenant.
- **Agent identity**: the service principal that represents one agent. It has an object ID, display name, optional sponsor and owners, and no credentials of its own.
- **Agent user (agent's user account)**: an optional account paired one-to-one with an agent identity for systems that require a user object, such as a mailbox or Teams chat participation.

Agent identities support three token patterns: an app token where the agent identity is the subject (autonomous agents), an incoming token whose audience is the agent identity (agent as a protected API, for example an A2A endpoint), and a user token where the subject is the user and the actor is the agent identity

(attended, on-behalf-of flows). For the exam, map them as follows: **attended agent acting for a user** means on-behalf-of with the user's permissions; **autonomous agent** means its own agent identity with narrowly scoped permissions; **agent that needs a mailbox** means an agent user.

Platform integration matters for design questions:

- **Copilot Studio**: before May 2026, Copilot Studio created an app registration per agent; after May 2026 it automatically creates a Microsoft Entra Agent ID for each new agent. Existing app-registration agents are being migrated by Microsoft; an optional early migration path exists PREVIEW. Authentication activity for agents with Agent IDs is logged in Microsoft Entra ID.
- **Microsoft Foundry**: Foundry provisions a blueprint and agent identities, assigns RBAC roles to the agent identity, and performs the token exchange when the agent calls a tool, so developers do not embed secrets in prompts, code or connection strings.
- **Agent 365**: agents discovered and inventoried through Agent 365 can be given a first-class Entra identity for lifecycle and policy enforcement.

Conditional Access for agents extends the policy engine to agent identities and agent users. Microsoft recommends at least one block-unless-approved policy for agent workloads, because agent token requests carry no interactive session, device or MFA signal. Example policies include blocking agent identities when Microsoft Entra ID Protection reports high agent risk (agent risk is PREVIEW), and requiring a compliant device for computer-using agent users via the *Agent execution environments* condition PREVIEW. Conditional Access for agents and ID Governance for agents require Microsoft 365 E7, or a Microsoft Agent 365 licence paired with at least Microsoft Entra ID P1 or Microsoft 365 E3. Agent ID itself is available to all Microsoft Entra customers.

> **EXAM TIP**
> Conditional Access policies that target *agent identities* do not apply to *agent user accounts*, and vice versa; they are separate principal types. If the stem says the agent "has its own mailbox" or "participates in Teams as a team member", target the agent user.

### 7.1.4 Tool credentials and the confused deputy problem

Every Copilot Studio tool (connector action, flow, MCP tool) runs with a credential. The default for connectors and flows is **End user credentials**, so the user's own permissions apply. Makers can switch to **Maker-provided credentials**, which means every user of the agent acts with the maker's (or a service account's) access. That is the textbook *confused deputy*: the agent uses a privileged identity to do something the requesting user could not. Copilot Studio's automatic security scan warns makers before publishing when they select maker-provided credentials, set *No authentication*, or share the agent with everyone in the organisation, because each relaxes a secure default.

Design guidance: keep end-user credentials for anything that reads or writes business data, and use a dedicated, least-privilege service identity (ideally an agent identity) only for genuinely autonomous scenarios. Store any remaining secrets in Azure Key Vault through environment variables rather than in topics. For Foundry agents and incoming A2A calls, Entra authentication is mandatory; choose on-behalf-of when the downstream system must enforce per-user permissions, and service identity (agent identity, managed identity or service principal) for back-end agent-to-agent work.

### 7.1.5 Platform hardening: network, encryption and operator access

| Control | What it protects | Key facts |
|---|---|---|
| Virtual network support (Power Platform) | Connector traffic to private resources | Connectors run in the context of your virtual network so internal resources are reachable without public exposure. |
| IP firewall (Dataverse) | Access to Dataverse from disallowed locations, token replay, exfiltration through client tools | Enforced only in managed environments; users also need an eligible Microsoft 365 E5/A5/G5 or compliance add-on licence. |
| Customer-managed keys (CMK) | Data at rest, including Copilot Studio data | Key vault admin creates key and enterprise policy; Power Platform admin assigns managed environments to the policy. When CMK is on for a Copilot Studio environment, all Copilot Studio data is encrypted with the customer key. |
| Customer Lockbox | Microsoft operator access to customer data | Supported by Copilot Studio, but does not cover Copilot Studio security audit telemetry (processed by the Purview audit pipeline) or Agent 365 governance and audit events. |
| Web channel security | Embedding agents in unauthorised sites | Require a secret or token to start a conversation on the web channel. |
| Continuous access evaluation | Stale tokens after critical events | Forces token refresh on events such as account disablement or network location change. |
| Microsoft Entra network controls via Agent 365 | Agent egress and ingress | Apply Entra network controls to Copilot Studio agents through Agent 365. |

**PRODUCTION NOTE**

The Copilot Studio secure-project checklist is a good architecture review template: assign Copilot Studio licences and environment access through Entra groups, assign Dataverse security roles through group teams, apply restrictive data policies, enable only relevant tenant, environment and agent settings, store secrets in Key Vault, and require a gated release process from development to test to production.

## 7.2 Design governance for agents

### 7.2.1 Three levels of control in Copilot Studio

Copilot Studio governance is applied at the tenant, environment and agent levels. Knowing which level owns a control is a common discriminator between answer options.

| Level | Feature controls | Maker access controls |
|---|---|---|
| **Tenant** | Data policies applied to all environments (unauthenticated usage, channels, knowledge sources, connectors, Azure Bot Service skills, Application Insights). Allow or block publishing of agents that use generative AI. | Copilot Studio licence assignment, pay-as-you-go author settings, self-service trials, access to the Copilot Studio Teams app. |

| Level | Feature controls | Maker access controls |
|---|---|---|
| **Environment** | Scope data policies to environments, allow or block public data sources such as Bing, allow or block generative AI where no regional capacity exists (data movement), network isolation (VNet, IP firewall). | Security group on the environment, security roles for create, read, update and delete of agents (Copilot, Copilot Subcomponent and Conversation Transcript tables). |
| **Agent** | Turn generative orchestration, AI knowledge, generative answers and intelligent authoring on or off; set authentication; enforce web channel security. | Share or unshare for co-authoring. System Administrators can read and update all agents and transcripts. |

### 7.2.2 Data policies for agents (Power Platform DLP)

Copilot Studio capabilities are exposed as connectors in the Power Platform admin center data policy designer (**Security > Data and privacy > Data policy**). A data policy can be applied to a single environment, an environment group or the whole tenant. Typical uses:

| Goal | Data policy action |
|---|---|
| **Require user authentication** | Block *Chat without Microsoft Entra ID authentication in Copilot Studio*. |
| **Stop local file uploads as knowledge** | Block *Knowledge source with documents in Copilot Studio*. This does not block SharePoint or OneDrive uploads. |
| **Stop SharePoint or OneDrive knowledge** | Block *Knowledge source with SharePoint and OneDrive in Copilot Studio* (and the SharePoint and OneDrive connectors), or use endpoint filtering to allow only specific sites. |
| **Allow only approved public websites as knowledge** | Endpoint filtering on the public website knowledge connector. |
| **Stop connectors being used as tools** | Block the specific Power Platform connectors. Blocking a connector also blocks MCP server tools that rely on it. |
| **Stop arbitrary outbound calls** | Block the *HTTP* connector, or endpoint-filter it to approved hosts. |
| **Control skills, channels, Application Insights, event triggers** | Block the corresponding Copilot Studio connectors (for example, block event triggers to govern autonomous agents). |

When a violation exists, the maker sees an error banner and the **Publish** button becomes unavailable; the downloadable details file lists each violation. Copilot Studio also surfaces connector dependency insights so admins can understand data movement risk before deployment.

For stricter environments, **advanced connector policies** (ACP) enforce a strict allowlist: every connector not explicitly allowed is blocked as soon as the policy is saved (single environment) or published (environment group). Microsoft recommends starting in mixed mode alongside classic data policies before moving to ACP-only mode PREVIEW. ACP can also allow individual connector actions and can be managed programmatically through the Power Platform API.

> **NOTE**
> Do not confuse **Power Platform data policies** (connector governance, formerly just called DLP policies) with **Microsoft Purview Data Loss Prevention** (content inspection by sensitive information type or sensitivity label, including the Microsoft 365 Copilot policy location). Exam stems use both; the first governs what an agent can connect to, the second governs what content Copilot can process.

### 7.2.3 Managed environments, environment groups, rules and sharing limits

Managed environments unlock the governance features most agent designs depend on: sharing limits, usage insights, IP firewall, IP cookie binding, solution checker enforcement and the Power Platform advisor actions. **Environment groups** let you apply **rules** to many managed environments at once, which is the scalable answer whenever a stem mentions "all development environments" or "every environment in the EMEA business unit". Rules relevant to agents include:

- **Sharing agents with Editor permissions** and **Sharing agents with Viewer permissions**.
- **Knowledge sources for agents**, **Skills in Copilot Studio**, **AI prompts**.
- **Generative AI settings** (including the *Move data across regions* consent at scale).
- **External models** (OpenAI, Anthropic, Mistral AI, xAI models in Copilot Studio) and **Preview and experimental AI models**.
- **Accessing transcripts from conversations in Copilot Studio agents**.
- **Advanced connector policy**, **Default deployment pipeline** PREVIEW, **Solution checker enforcement**, **Unmanaged customizations**.
- **Maker welcome content** and **Maker guidelines** PREVIEW, and **Cost controls** (draw from tenant credit pool).

Agent sharing limits work through two assignments: **Editor** (edit, configure, share and publish) and **Viewer** (chat only). Editor permission can only be granted to individuals, never to security groups.

| Sharing rule | Effect when selected |
|---|---|
| Let people grant Editor permissions when agents are shared | Owners and editors can add individual editors. |
| Let people grant Viewer permissions when agents are shared | Owners and editors can share with individual viewers and security groups. When cleared, they can share with neither. |
| Only share with individuals (no security groups) | Viewer sharing restricted to individuals. |
| Limit number of viewers who can access each agent | Caps viewers per agent; available when sharing is restricted to individuals. |

Sharing rules are enforced when someone tries to share; they do not remove access that already exists, and enforcement can take up to an hour to begin. If an agent becomes non-compliant, the only allowed sharing change is to stop sharing until it complies.

**Environment routing** sends new makers to a personal developer environment instead of the default environment, giving them a safe space to build; a **maker welcome message** informs them of privacy and compliance expectations.

### 7.2.4 Zoned governance

Copilot Studio guidance recommends a zoned (layered) environment strategy that separates experimentation from production, with policy strictness tuned to the zone:

| Zone | Data policies | Managed environment rules | Typical stance |
|---|---|---|---|
| **Personal development** | Strict | Strict | No unauthenticated agents; most channels and connectors blocked. |
| **Dedicated (team) development** | Relaxed | Strict | No unauthenticated agents, but the project's channels and connectors allowed. |
| **Dedicated test and production** | Relaxed | Relaxed | Reviewed agents may use the channels they need (including public, unauthenticated web where justified), reached only through a gated release. |

The agent adoption maturity model reinforces the point: at level 300 (Defined), agents are explicitly classified by purpose, criticality and autonomy, and the zoned model is adopted; at level 400 (Capable), governance is risk-based and partially automated, productivity agents move fast with light controls while mission-critical agents follow full enterprise ALM, security and compliance rigour. Universal anti-patterns to recognise in a stem include no inventory or ownership, guidance-only controls that are not enforced technically, treating every agent the same, not treating connector governance as an "agent safety boundary", and treating audit and monitoring as afterthoughts.

### 7.2.5 Agent inventory and lifecycle: Microsoft 365 admin center and Agent 365

The **Agent Registry** in the Microsoft 365 admin center (**Agents > All agents**) is the tenant inventory of agents. Basic governance actions are included with Microsoft 365 Enterprise, Business, Education and Frontline plans:

- **Publish, deploy (install), uninstall, block and unblock**: blocking a Copilot Studio or Agent Builder agent affects its availability in Microsoft Copilot and other hosts such as Teams and Outlook.
- **Activate**: allow only selected users or groups to create instances of an agent.
- **Delete, restore, permanently delete**: deletion has a 30-day recovery window.
- **Start and stop**: start or stop the underlying Azure infrastructure of Foundry agents.
- **Assign or add an owner**: the dashboard surfaces **agents without owners** (for example after the creator leaves) so admins can reassign, block or delete them.
- Rule-based lifecycle automation and synchronisation of agents from external platforms into the registry.

**Microsoft Agent 365** is the control plane to *observe*, *govern* and *secure* agents, generally available for the Commercial segment since May 1, 2026, licensed per user (included in Microsoft 365 E7, or as an add-on). It adds policy templates, observability and monitoring of agent activity, the agent map, and tenant-wide control over which tools (including Microsoft MCP servers) agents can access. It works with Microsoft Entra (identity and Conditional Access), Microsoft Purview (information protection, DLP, audit) and Microsoft Defender (threat detection) to secure agents end to end.

**EXAM TIP**

"Find every agent in the tenant and who owns it" points to the Agent Registry (with Agent 365 for cross-platform observability). "Make a human accountable for an autonomous agent's access" points to the Entra Agent ID **sponsor**. "Stop makers sharing broadly" points to managed environment sharing limits. Three different planes, three different answers.

### 7.2.6 Governance controls by platform

| Concern | Microsoft 365 Copilot and agents | Copilot Studio | Microsoft Foundry | Dynamics 365 (Copilot and agents) |
|---|---|---|---|---|
| **Who can build** | Agent Builder availability, Microsoft 365 admin center agent settings | Licences via groups, environment security groups, security roles, environment routing | Foundry RBAC (Foundry User, Project Manager, Account Owner, Owner) | Security roles; Copilot feature enablement per app and environment |
| **Who can use** | Deploy, activate, block in Agent Registry | Sharing limits, authentication, channel policy | Foundry Agent Consumer role at project or agent scope | Security roles, feature settings, user licensing |
| **What it can reach** | User permissions, labels, RCD, Purview DLP | Data policies, ACP, endpoint filtering, knowledge rule | Connections, RBAC on downstream resources, egress controls, private networking | Dataverse security model, data policies |
| **Inventory and lifecycle** | Agent Registry, Agent 365 | Agent Registry, Agent 365, Power Platform inventory | Agent Registry (start and stop), Foundry control plane, Agent 365 | Power Platform admin center, Copilot Hub |
| **Audit** | Purview audit, DSPM | Purview audit, DSPM, Sentinel | Azure activity log, resource logs, Purview for prompts | Dataverse auditing, Purview activity logging |

## 7.3 Design model security

### 7.3.1 Foundry role-based access control

Model security starts with who can deploy, change, call and tune models. Microsoft Foundry adds built-in roles on top of Azure Owner, Contributor and Reader. The roles were recently renamed: **Foundry User**, **Foundry Owner**, **Foundry Account Owner** and **Foundry Project Manager** were previously Azure AI User, Azure AI Owner, Azure AI Account Owner and Azure AI Project Manager (role IDs and permissions unchanged).

| Role | Create accounts and projects | Build in project (data actions) | Assign roles | Manage models | Publish agents | Call agent endpoints |
|---|---|---|---|---|---|---|
| Foundry Agent Consumer | No | No | No | No | No | Yes |
| Foundry User | No | Yes | No | No | No | Yes |

| Role | Create accounts and projects | Build in project (data actions) | Assign roles | Manage models | Publish agents | Call agent endpoints |
|---|---|---|---|---|---|---|
| Foundry Project Manager | No | Yes | Foundry User only | No | Yes | Yes |
| Foundry Account Owner | Yes | No | Foundry User, ACR and monitoring roles | Yes | No | No |
| Foundry Owner | Yes | Yes | Foundry User, ACR and monitoring roles | Yes | Yes | Yes |
| Azure Contributor | Yes | No | No | Yes | No | No |

A sample enterprise mapping: IT admin is Owner at subscription scope; managers are Foundry Account Owner on the resource (deploy models, audit compute and connections, create shared connections, but cannot build); lead developers are Foundry Project Manager (create projects, publish agents, assign Foundry User); developers are Foundry User on the project plus Reader on the resource; consuming users and services get Foundry Agent Consumer at project scope, or at agent scope for per-agent control. Note that Azure Contributor can deploy models, so do not hand it out casually on AI resources.

**COMMON TRAP**

Fine-tuning needs both data plane permissions (upload data, run the job) and control plane permissions (deploy the fine-tuned model). The only single built-in role with both is **Foundry Owner**. For separation of duties, combine **Foundry User** (data plane) with **Foundry Account Owner** (control plane). Foundry Project Manager cannot manage models.

### 7.3.2 Keyless authentication and managed identities

API keys are bearer secrets: anyone holding one can call the model, and key use is hard to attribute to a person or agent. The recommended design is Microsoft Entra ID authentication end to end:

- Applications and agents call Foundry with Entra tokens (for example `DefaultAzureCredential` in code), using managed identities or agent identities instead of stored keys.
- Disable local (key) authentication per resource (`DisableLocalAuth`) once all consumers support Entra ID, and enforce it at scale with the Azure Policy definition *Azure AI Services resources should have key access disabled (disable local authentication)*.
- Grant the Foundry resource's managed identity data roles on dependent resources, for example **Storage Blob Data Reader** on a storage account, or the appropriate Azure AI Search roles, instead of connection strings.
- Use Entra authentication for agent trace ingestion into Application Insights PREVIEW.

**Listing 7.1: Disable key-based access on a Foundry resource after moving all clients to Entra ID**

```
Connect-AzAccount
Set-AzCognitiveServicesAccount -ResourceGroupName "rg-ai-prod" `
    -Name "fdy-contoso-prod" -DisableLocalAuth $true
```

### 7.3.3 Network isolation for models and agents

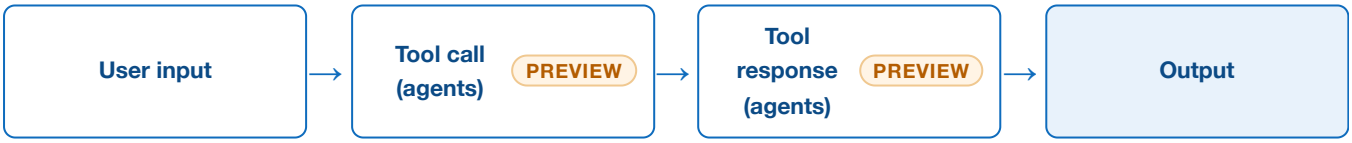
Foundry supports inbound isolation (disable public network access and use private endpoints for the account; projects inherit the account's network posture) and outbound isolation for the Agent Service and evaluations, with two options:

| Option | Choose when | Trade-off |
|---|---|---|
| **Managed virtual network** | You want a secure default without building or operating your own network; Microsoft provisions the network that secures the Agent Service compute, and you add private endpoints to approved Azure resources. | Less control over topology; check supported regions. |
| **Bring your own VNet (virtual network injection)** | You need full control and integration with an existing hub-and-spoke, firewalls and private DNS. | For a Standard agent with end-to-end isolation, you must bring your own Storage, Azure AI Search and Azure Cosmos DB resources and manage DNS zones. |

Two current limitations are worth remembering: for hosted agents deployed with azd, the agent endpoint URL itself stays publicly addressable in the current preview (sessions are isolated by identity) PREVIEW; and hosted agents support **network egress controls** configured in the same guardrail (RAI policy) to restrict outbound destinations PREVIEW.

### 7.3.4 Guardrails and content safety

Foundry guardrails apply to model deployments and to agents (agent guardrails are PREVIEW). Each control defines a risk, the intervention points to scan and the action (for example annotate, or annotate and block). There are four intervention points:




| Risk | Models | Agents | Purpose |
|---|---|---|---|
| Hate, sexual, self-harm, violence | Yes | Yes | Harm category filters with severity thresholds |
| User prompt attacks (Prompt Shields) | Yes | Yes | Detect jailbreak and direct injection in user input |
| Indirect attacks (Prompt Shields document attacks) | Yes | Yes | Detect instructions embedded in documents and, for agents, in tool responses |
| Spotlighting | Yes PREVIEW | No | Base-64 encode documents so the model treats them as less trusted; Chat Completions API only; adds tokens |
| Protected material (text and code) | Yes | Yes | Detect known copyrighted text and public code |
| Groundedness, PII, task adherence, custom block lists | Varies | Varies | Detect ungrounded output, sensitive data, agents straying from their task, organisation-specific terms |

Third-party guardrail integrations can plug partner safety systems into the same intervention points. Models sold by Azure are stateless: prompts and completions are not stored in the model and are not used to train, retrain or improve base models.

### 7.3.5 Model choice governance in low-code platforms

In Copilot Studio and Power Platform, the model is a governed setting rather than a deployment you own. Admins control whether external models (Anthropic, Mistral AI, xAI, and OpenAI models in Copilot Studio) and preview or experimental models are allowed per environment or through environment group rules. In Microsoft 365 Copilot, Purview audit records include the model provider and model name for each request, which supports governance when multiple providers are enabled. Some models in Copilot Studio might process data outside the environment's geography even when cross-geo movement is restricted, so model selection is also a residency decision (see 7.6).

### 7.3.6 Threat detection for models and agents

**Microsoft Defender for Cloud AI threat protection** (generally available) works with Prompt Shields and Microsoft threat intelligence to raise alerts for jailbreak attempts, data leakage, data poisoning, credential theft (for example credentials appearing in model responses), and more, with optional prompt evidence. Alerts integrate with Defender XDR for incident correlation. Defender for Cloud also provides AI security posture recommendations that flag misconfigurations. For agents managed by Agent 365, **Microsoft Defender** detects threats to AI agents in near real time (jailbreak and XPIA attempts, secret leakage, LLM reconnaissance, suspicious access) PREVIEW.

> **NOTE**
> Foundry's own compliance guidance frames model security as three cooperating products: Foundry guardrails at the model layer, Defender for Cloud for posture and threat detection, and Microsoft Purview for auditing, sensitive information classification and DSPM for AI over prompt and response data. Purview data security policies require Entra user-context tokens to attach to user interactions.

## 7.4 Analyse solution and AI vulnerabilities and mitigations, including prompt manipulation

### 7.4.1 Threat taxonomy for agentic solutions

Microsoft maps agent risks to the OWASP Top 10 for LLM applications, the OWASP agentic AI threats, MITRE ATLAS and the MSRC AI severity classification. For the exam, be fluent in these:

- **Direct prompt injection and jailbreak (UPIA)**: the user types instructions designed to override the system prompt or safety rules.
- **Indirect prompt injection (XPIA)**: malicious instructions hidden in content the agent processes (email, document, web page, image, code, MCP tool output, another agent's reply). The user may be innocent. Microsoft calls this "prompt injection to action" when it causes tool calls.
- **Data exfiltration**: sensitive data leaves through the response, a tool call (for example an HTTP request or email action), markdown links or images, or a public channel.

- **Oversharing**: the agent faithfully returns content that the user technically has access to but should not see, because permissions are too broad ("Everyone except external users", "Anyone" links, broken inheritance).
- **Excessive agency**: the agent has more tools, permissions or autonomy than the task requires (OWASP LLM06).
- **Confused deputy**: the agent uses a privileged identity to do something the requesting user cannot.
- **Memory poisoning**, **unbounded loops and cost exhaustion**, **multi-agent trust failures**, **rogue or impersonated agents**, and **rug-pull** attacks where a trusted MCP server or tool changes behaviour after approval.

### 7.4.2 Threat to mitigation to product control

| Threat | Design mitigation | Product controls |
|---|---|---|
| **Direct prompt injection / jailbreak** | Detect and block at input; keep system instructions minimal and non-secret; never rely on the prompt as a security boundary. | Foundry Prompt Shields (user prompt attack) at user input; built-in UPIA protection in Copilot Studio; Microsoft 365 Copilot block lists and jailbreak classifiers; Defender for Cloud jailbreak alerts. |
| **Indirect prompt injection** | Treat all retrieved, tool and agent output as untrusted; isolate instructions from data; scan tool responses; gate high-impact actions. | Prompt Shields document attacks (including tool response for Foundry agents PREVIEW); spotlighting for models; built-in XPIA protection in Copilot Studio; external threat detection for Copilot Studio PREVIEW; Defender XPIA detections for agents PREVIEW. |
| **Data exfiltration via tools** | Least functionality; deny outbound channels the task does not need; inspect proposed tool calls. | Power Platform data policies (block HTTP, endpoint filtering, block channels); ACP allowlists; Foundry tool call intervention point and egress controls PREVIEW; Copilot Studio real-time risk assessment and security scan; managed VNet. |
| **Data exfiltration via prompts and responses** | Classify data; stop sensitive data reaching or leaving the model. | Purview DLP for Microsoft 365 Copilot (block processing of labelled files and emails; block prompts with SITs PREVIEW; block web search when prompts contain SITs); sensitivity labels without EXTRACT; Endpoint DLP for third-party AI sites. |
| **Oversharing** | Fix permissions at source; reduce discoverability while remediating; label sensitive content. | SharePoint Advanced Management (data access governance reports, content management assessment, Restricted Content Discovery); Purview DSPM data risk assessments; sensitivity labels; agent sharing limits. |
| **Excessive agency** | Least privilege per tool; narrow instructions; human approval for consequential actions; step, iteration and budget limits. | End-user credentials for tools; scoped agent identity and RBAC; approvals in agent flows, Power Automate or Logic Apps; Azure Policy for critical operations; Foundry task adherence; Agent 365 tool access policies. |
| **Confused deputy** | On-behalf-of tokens and per-action authorisation; no standing broad identity. | Copilot Studio end-user credentials; Entra Agent ID OBO flows; Foundry OBO for A2A. |
| **Rogue, impersonated or ownerless agents** | Strong agent identity, inventory, accountable ownership, risk-based access. | Entra Agent ID with sponsors; Conditional Access with agent risk PREVIEW; Agent Registry ownerless view; Agent 365. |

| Threat | Design mitigation | Product controls |
|---|---|---|
| **Harmful or infringing output** | Filter and ground; disclose limitations. | Harm category filters, protected material detection, groundedness detection; Copilot Studio content moderation. |

### 7.4.3 Defence in depth against indirect prompt injection

Indirect prompt injection cannot be fully solved by input validation, because the model cannot reliably distinguish user intent from instructions embedded in data. Microsoft's Secure Future Initiative pattern therefore layers probabilistic and deterministic controls:




The architecture implication is that **deterministic controls must bound what a successful injection can achieve**. If the agent cannot send email outside the tenant (data policy), cannot reach unapproved hosts (endpoint filtering or egress control), uses the user's own permissions (end-user credentials, OBO) and must ask a human before paying an invoice (approval step), an injected instruction has little to work with even if the classifier misses it.

> **COMMON TRAP**
> Spotlighting is a model-level setting for the Chat Completions API; it is not applicable to agents. For a Foundry agent that reads emails or web pages through tools, the correct control is a **document attack** (indirect attack) control at the **tool response** intervention point, plus least-privilege tools.

### 7.4.4 Copilot Studio runtime protections

Custom agents in Copilot Studio are secure by default with built-in protection against UPIA and XPIA; attacks of these types are blocked at runtime. For more oversight, organisations can register an **external threat detection system** PREVIEW: a REST endpoint that the orchestrator calls each time it considers invoking a tool, which can block the tool call. It only applies to agents using generative orchestration (classic agents are skipped). Makers also see:

- An **automatic security scan** and warnings before publishing when secure defaults are relaxed.
- **Real-time risk assessment** findings while configuring knowledge, tools and actions, to fix exfiltration risks before publishing.
- **Agent runtime protection status** on the Agents page, complemented by Agent 365 inventory and ownership telemetry.

Copilot Studio content moderation covers predefined harmful-content categories; there is no tenant-configurable policy that blocks conversations on arbitrary administrator-defined subjects, so topic restrictions belong in instructions, topic design and knowledge scope.

### 7.4.5 Oversharing in Microsoft 365 Copilot and agents

Microsoft 365 Copilot and SharePoint-grounded agents only return content the user can already access. That is exactly why oversharing is the dominant data risk: Copilot makes overshared content easy to find. The recommended sequence is:

1. **Identify** high-risk sites and content: Purview DSPM data risk assessments; SharePoint Advanced Management data access governance reports (sharing links, sensitivity labels, Everyone except external users, permission state) and the content management assessment (oversized audiences, EEEU, broken inheritance, ownerless or inactive sites).
2. **Apply interim protection**: **Restricted Content Discovery** on high-risk sites, which removes them from organisation-wide search and Copilot answers (except for content the user owns or recently interacted with) without changing permissions.
3. **Remediate** permissions, sharing links and ownership; apply sensitivity labels and auto-labelling.
4. **Protect continuously**: Purview DLP for Copilot, sensitivity labels with appropriate usage rights, DSPM monitoring.

> **COMMON TRAP**
> **Restricted SharePoint Search** (an allow list of curated sites for organisation-wide search and Copilot) was always a short-term measure and is retiring: starting July 31, 2026, new enablement is blocked. Microsoft directs customers to Restricted Content Discovery. Also note that when Restricted SharePoint Search is enabled, SharePoint knowledge in Copilot Studio is blocked.

Sensitivity labels with encryption add a second gate: Copilot and agents only return encrypted content to a user who has the **EXTRACT** usage right (shown as "Copy" in Office). The person who applied encryption always has EXTRACT as Rights Management owner. Purview DLP for the Microsoft 365 Copilot policy location can prevent Copilot from processing items with specific labels, block responses when prompts contain sensitive information types PREVIEW, and stop external web search grounding when prompts contain sensitive data. Copilot Studio agents grounded on SharePoint display the highest-priority sensitivity label of the sources used in a response.

### 7.4.6 Testing for vulnerabilities before and after release

Microsoft frames AI risk work with the NIST AI RMF functions Govern, Map, Measure and Manage. Red teaming (manual, by people with diverse backgrounds and adversarial mindset) uncovers harms; systematic measurement then validates mitigations. The Foundry **AI Red Teaming Agent** automates adversarial probing so teams can "shift left" from costly incidents to pre-deployment testing; local runs are documented as PREVIEW. Repeat red teaming and safety evaluations at every major change: new model version, new tool, new knowledge source or broader audience.

## 7.5 Review the solution for adherence to responsible AI principles

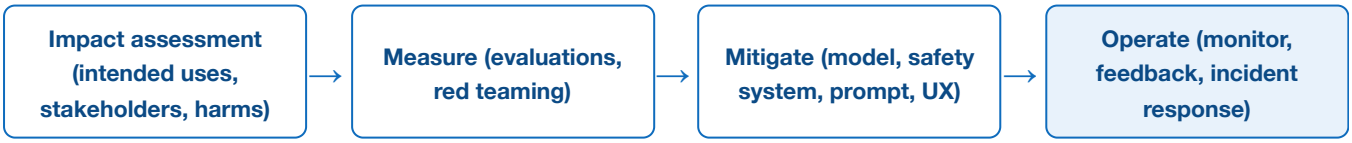
### 7.5.1 The six principles as review questions

The Microsoft Responsible AI Standard is grounded in six principles: fairness, reliability and safety, privacy and security, inclusiveness, transparency, and accountability. A review turns each into concrete questions and evidence.

| Principle | Review question for an agent | Evidence and controls |
|---|---|---|
| **Fairness** | Does the agent treat comparable people comparably (for example in triage, eligibility or prioritisation)? | Disaggregated evaluation results, bias testing across groups, human review of adverse decisions. |
| **Reliability and safety** | Does it perform as intended, fail safely and stay on task? | Evaluation suites, groundedness checks, task adherence, red teaming results, fallback and escalation paths, loop and budget limits. |
| **Privacy and security** | Is data minimised, permission-trimmed, protected and resident where required? | Authentication, DLP, labels, residency configuration, CMK, threat protection, DPIA. |
| **Inclusiveness** | Can people with different abilities, languages and channels use it? | Accessibility testing, supported languages, voice and text options. |
| **Transparency** | Do users know they are using AI, what it can and cannot do, and where answers come from? | AI disclosure in the greeting, citations, application cards and transparency notes, documented limitations. |
| **Accountability** | Is a named human accountable for the agent and its outcomes? | Owner and sponsor, AI council approval, audit trails, incident response plan, human-in-the-loop for consequential actions. |

### 7.5.2 The responsible AI lifecycle: identify, measure, mitigate, operate

For generative solutions, Microsoft's practice guidance runs in stages. **Identify** potential harms as early as possible through an impact assessment, iterative red teaming, stress testing and analysis, producing a prioritised list of harms per scenario. **Measure** their frequency and severity with clear metrics and test sets. **Mitigate** at several layers: model, safety system (guardrails), system message and grounding, and user experience. **Operate** with monitoring, feedback, incident response and phased rollout. The Cloud Adoption Framework governance guidance adds the organisational step: understand each AI workload's purpose, data sources and intended outcomes, document assumptions and limitations, and evaluate each workload against the responsible AI principles to identify risks.




### 7.5.3 Transparency artefacts

Microsoft publishes Transparency Notes and, increasingly, **application cards** (for example the application cards for Microsoft Copilot and for Copilot Studio) that describe capabilities, intended uses, limitations and best practices. Foundry provides transparency notes for agents and models. Use them in two ways: as inputs to your own impact assessment (the platform's known limitations are your risks too), and as a template for the user-facing transparency your solution needs. Pair them with the Microsoft Enterprise AI Services Code of Conduct when judging whether a use case is acceptable at all.

### 7.5.4 Operationalising responsible AI

The agent maturity guidance recommends four actions: define a responsible AI standard (adapted from the Microsoft principles or the NIST AI RMF) with practical tools such as impact assessments, bias testing and monitoring; establish a cross-functional AI council (business, IT, security, risk, legal and compliance) that

reviews high-impact use cases; embed responsible AI into delivery, starting every project with a responsible AI kickoff that asks how the system could cause harm or unfairness; and build team habits so it is everyone's job. Pitfalls to recognise: treating responsible AI as only security or compliance, and treating it as a one-time pre-deployment review rather than an ongoing operational responsibility.

**EXAM TIP**

If a stem asks what to do *first* for a high-impact use case (credit, employment, health, safety), the answer is the impact assessment and harm identification, not a content filter. Filters are mitigations; you cannot choose mitigations before you know the harms.

**PRODUCTION NOTE**

Build the review into ALM: make the impact assessment, evaluation results and red team report required artefacts of the production gate in your deployment pipeline, and require a re-review when the model version, tools, knowledge or audience change.

## 7.6 Validate data residency and movement compliance

### 7.6.1 Separate storage at rest from processing

Residency questions become simple once you separate two things: **where customer data is stored at rest** (governed by the Product Terms data residency commitments and add-ons such as Advanced Data Residency and Multi-Geo) and **where prompts and responses are processed** by the language model (governed by service-specific routing, the EU Data Boundary, Power Platform data movement consent and Foundry deployment types). A solution can store everything in-region and still process inference elsewhere if you allow it.

### 7.6.2 Microsoft 365 Copilot

- Microsoft 365 Copilot is a covered workload in the Product Terms data residency commitments (added March 1, 2024). Advanced Data Residency and Multi-Geo extend commitments to local region geographies and satellite geographies.
- LLM calls route to the closest datacentres in the region but can use other regions during high utilisation. For EU users, traffic stays within the **EU Data Boundary**, while worldwide traffic may be processed in the EU and other regions.
- Models provided by Anthropic as a subprocessor are currently excluded from the EU Data Boundary.
- **EUDB and Multi-Geo are mutually exclusive**: tenants with Multi-Geo subscriptions are not in scope for the EUDB even when the tenant is in the EU or EFTA.
- When users collaborate across regions, each user's prompts and responses are stored in that user's preferred data location, while the original document stays where it is stored.

### 7.6.3 Power Platform, Copilot Studio and Dynamics 365

Copilot Studio stores data in the geography of the environment, supports the EU Data Boundary, and supports GDPR data subject requests. Generative features rely on Azure OpenAI and, optionally, Bing. Where in-region capacity is limited, the **Move data across regions** consent in the environment's **Generative AI features** card

lets prompts and outputs be processed where the feature is hosted.

| Environment region | Azure OpenAI processing | Bing Search processing |
|---|---|---|
| United States, government clouds (GCC, GCC High) | In region | United States |
| Europe, France, Germany, Norway, Sweden, Switzerland | In EU Data Boundary | United States |
| United Kingdom | In region or in EU Data Boundary | United States |
| Asia, Singapore, Australia, India | In region or United States | United States |
| Brazil, Canada, Japan, Korea, South Africa, United Arab Emirates | United States | United States |

Facts to memorise:

- Only Global admins and Power Platform (or Dynamics 365) admins can grant consent. The checkbox is not shown for US or government cloud environments, or where full in-region capacity exists.
- Data moves only if the model is not deployed locally, capacity overflows, or a reliability issue affects the local model; Microsoft does not log, store or retain the input or output during this process, and Product Terms commitments for data at rest are unaffected.
- **Data movement that already occurred cannot be reversed** by clearing the checkbox.
- The **Bing search** and **Microsoft 365 services** checkboxes require *Move data across regions* to be selected first where it is displayed. Bing-powered features are governed by the Microsoft Services Agreement and Privacy Statement, and Bing processing happens in the United States for every region.
- For EU Data Boundary environments, **Allow flex routing during periods of peak load** lets inferencing and limited pseudonymised data go outside the EUDB at peak. Clear it to keep all inferencing inside the EUDB. It defaults from the Microsoft 365 admin center toggle and is on by default for eligible tenants created after March 25, 2026.
- Use the **Generative AI settings** environment group rule to manage the setting at scale.
- Clearing the checkbox does not turn off Copilot; it may disable only the features that need cross-region capacity. Use feature-specific controls to turn features off.
- Some Copilot Studio models might process data outside the environment's geography even when cross-geo movement is restricted; check model availability by region before selecting a model.

Dynamics 365 Sales, Customer Service and Customer Insights Copilot features, Power Apps and Power Automate Copilot features, and AI Builder prompts are among the features that depend on this consent. Prompts and responses in Dynamics 365 and Power Platform Copilot are not used to train Microsoft models unless an admin opts in to data sharing.

### 7.6.4 Microsoft Foundry deployment types

| Processing option | Where prompts and responses are processed | Choose when |
|---|---|---|
| **Global (Standard or Provisioned)** | Any geography where the model is deployed | Highest throughput, broadest and earliest model availability, no processing-location constraint. |

| Processing option | Where prompts and responses are processed | Choose when |
|---|---|---|
| **Data Zone (Standard or Provisioned)** | Anywhere within the data zone: United States, or European Union (EU Data Boundary) | Regulatory need to keep processing in the US or EU, with more capacity than a single region. |
| **Regional / Azure geography (Standard or Provisioned)** | The Azure geography of the resource | Strict in-country processing requirements. |

For Global and Data Zone deployments, data stored at rest (uploaded files, fine-tuning data, stored completions, threads) stays in the customer-designated geography of the Foundry resource. These deployment types apply to serverless API deployments; models on managed compute run where you deploy them.

### 7.6.5 Residency decision summary

| Requirement | Microsoft 365 Copilot | Copilot Studio / Dynamics 365 | Foundry |
|---|---|---|---|
| **Data at rest in country** | Product Terms; ADR or Multi-Geo add-ons | Environment region | Resource region (geography) |
| **Inference within EU only, even at peak** | EUDB without flex routing; avoid Anthropic models | EU environment; clear flex routing; do not enable Bing search | Data Zone (EU) or regional deployment in an EU region |
| **No inference outside region at all** | Not configurable beyond EUDB and commitments | Do not select Move data across regions (accept reduced features) | Regional deployment type |
| **Maximum availability** | Default | Select Move data across regions | Global deployment type |

**COMMON TRAP**

Turning on *Move data across regions* does not change where Dataverse data or transcripts are stored; it changes where prompts and outputs may be processed. And turning it off later does not undo movement that has already happened.

## 7.7 Design access controls on grounding data and model tuning

### 7.7.1 The principle: retrieval must be permission-aware

A grounded agent is only as safe as its retrieval layer. The design goal is that every query retrieves only what the calling user (or, for autonomous agents, the scoped agent identity) is authorised to read, enforced by the data source or index, not by instructions in the prompt. Three patterns cover most scenarios: **delegated access** (the agent queries the source with the user's token), **permission-trimmed index** (permissions are synchronised into the index and enforced at query time) and **curated scope** (the knowledge source contains only data every authorised user may see).

### 7.7.2 Access controls by grounding source

| Grounding source | How access is enforced | Design notes |
|---|---|---|
| **SharePoint and OneDrive (Microsoft 365 Copilot, Copilot Studio)** | User's own permissions through Microsoft authentication; sensitivity labels and EXTRACT usage right | Copilot Studio preconfigures Microsoft authentication for SharePoint sources in Teams, Power Apps and Microsoft Copilot; manual authentication requires a configured Entra app registration. Generative answers from SharePoint are not available to guest users in SSO-enabled apps. Fix oversharing first (7.4.5). |
| **Uploaded documents (Copilot Studio)** | Everyone who can use the agent can receive answers from the files | Only upload content that all intended users may see; govern with the documents knowledge connector in data policies. |
| **Dataverse tables (Copilot Studio)** | Dataverse security model: security roles, business units, teams, record sharing, column security | Requires Dataverse search. Use group teams mapped to Entra groups for role assignment. Access is cumulative across roles, teams and shares. |
| **Power Platform connectors as real-time knowledge** PREVIEW | Runtime calls use the user's authentication tokens; only metadata is indexed | Source-system access controls are retained; governed by the same data policies as tools. |
| **Copilot connectors (formerly Graph connectors), for example SharePoint Server** | Access permission mode in the connector | Choose *Only people with access to the content in the data source* rather than *Everyone*; requires identity sync. |
| **Azure AI Search / Foundry IQ** | Document-level access control | See 7.7.3. |

Dataverse security in brief: a **security role** combines privileges (Create, Read, Write, Delete, Append, AppendTo, Assign, Share) with access levels (Global, Deep, Local, Basic, None). Record ownership is either organisation-owned (all or nothing) or user or team owned (user, business unit, parent-child business unit or organisation scope). **Business units** are security boundaries; Microsoft recommends one Entra security group and one Dataverse group team per business unit, with matrix access through multiple group memberships. Use record sharing only for exceptions. Dynamics 365 Copilot features inherit these security and compliance boundaries.

### 7.7.3 Document-level security in Azure AI Search

For custom RAG and Foundry agents, Azure AI Search (which underpins **Foundry IQ** knowledge bases) offers several patterns:

| Pattern | How it works | When to use |
|---|---|---|
| **POSIX-like ACL and RBAC scopes** PREVIEW | ACLs and RBAC scopes from ADLS Gen2 (indexer) or pushed documents are stored with each document and compared with the caller's Entra token claims | Data lake sources with existing ACLs |

| Pattern | How it works | When to use |
|---|---|---|
| **SharePoint in Microsoft 365 ACLs** PREVIEW | Indexer extracts document permissions | Indexing SharePoint for a custom agent |
| **Microsoft Purview sensitivity labels** | Label metadata indexed and enforced at query time | Label-driven access |
| **Security trimming filters** | A string field of user or group IDs, filtered by the application at query time | Sources without built-in ACL support; simple string comparison, the app must resolve groups |

With native token-based enforcement PREVIEW, the application passes the user's token in the `x-ms-query-source-authorization` header; the service extracts user, group and scope claims, compares them with the synchronised permission metadata and returns only authorised documents. Benefits: no custom permission code for nested groups, alignment with Entra, Purview and SharePoint permissions. Remember that enforcement is only as fresh as the last ACL synchronisation.

**EXAM TIP**

If a stem says "each employee must only see HR documents they can open in SharePoint" for a Foundry or custom agent, the answer is document-level access control with token-based query enforcement, not separate indexes per department and not a prompt instruction.

### 7.7.4 Protecting model tuning

Fine-tuning data is often the most sensitive data in an AI programme (labelled customer interactions, internal documents), and the resulting model encodes it. Controls:

- **Who can tune and deploy**: Foundry Owner, or Foundry User plus Foundry Account Owner for separation of duties (7.3.1). Keep Contributor off production AI resources.
- **Where training data lives**: training, validation and results files are stored in the Foundry resource in your tenant, in the resource's geography, encrypted with Microsoft-managed AES-256 by default with optional CMK (double encryption for fine-tuned models). Training data is not used to train any Microsoft or third-party model except as you direct.
- **Who can read the datastore**: grant the Foundry managed identity **Storage Blob Data Reader** on the storage account rather than sharing keys, and restrict human access to the container.
- **Who can use the tuned model**: fine-tuned models are available exclusively to you, cannot be downloaded (for serverless deployments) and can be deleted at any time. Control callers with Foundry RBAC and keyless authentication.
- **Kill switch**: revoking the CMK (disable, delete or remove permissions) makes encrypted fine-tuning assets inaccessible: downloads, new fine-tuning jobs and deployments that depend on the encrypted data fail.
- **Lifecycle**: delete files and fine-tuned models when no longer needed through the DELETE APIs, and align with your retention schedule.

## 7.8 Design audit trails for changes to models and data

### 7.8.1 What must be auditable

An auditor will ask five questions about an AI solution: who changed the agent (instructions, knowledge, tools, authentication, sharing), who changed the model (deployment, version, guardrail, fine-tune), who changed the data (records, files, index), who changed the policies (data policies, environment settings, Conditional Access), and what users and agents actually did (interactions and tool calls). No single log answers all five, so the design must name a source for each and route them to a central store with adequate retention.




### 7.8.2 Copilot Studio and Microsoft 365 Copilot in Microsoft Purview

Copilot Studio logs administrative, maker and user activities to Microsoft Purview **Audit** (also available through the Office 365 Management API). Administrative activity collection is enabled by default and cannot be disabled at the source, although audit can be turned off tenant-wide in Purview. Selected authoring events:

| Event label | Records |
|---|---|
| `BotCreate`, `BotDelete`, `BotDeleteCleanup` | Agent creation and deletion |
| `BotUpdateOperation-BotAuthUpdate` | Changes to authentication settings |
| `BotUpdateOperation-BotPublish`, `BotUpdateOperation-BotShare` | Publishing and sharing |
| `BotComponentCreate` / `Update` / `Delete` | Topics, skills and other components |
| `AIPluginOperationCreate` / `Update` / `Delete` | AI plugin (tool) changes |
| `EnvironmentVariableCreate` / `Update` / `Delete` | Environment variables used by the agent |
| `BotAppInsightsUpdate` | Application Insights logging configuration |
| `CopilotInteraction` | User interactions with the agent (metadata, channel as `AppHost`, transcript thread ID) |

Key details: the Audit solution does **not** contain the conversation text, only the transcript thread ID; **DSPM for AI** retrieves the chat text and accessed resources for `CopilotInteraction` events. To prevent retention of message text, create a Purview Data Lifecycle Management retention policy of 0 days for Copilot experiences. Users need a Microsoft 365 licence for events to be recorded, and FedRAMP tenants are excluded. Copilot Studio audit logs can also be monitored and alerted on in Microsoft Sentinel. For Microsoft 365 Copilot, Purview audit records interactions and, for Microsoft Copilot and Copilot Chat, the model provider and model name.

DSPM one-click collection policies extend prompt and response capture to enterprise AI apps, including apps connected through Microsoft Entra or Microsoft Foundry, so they can be managed with eDiscovery and Data Lifecycle Management.

### 7.8.3 Power Platform and Dataverse

Power Platform activity logging in Purview covers connectors (API calls and connector events), Copilot Studio, Dataverse and model-driven apps, Power Apps, Power Automate and data policy activity. For business data changes, **Dataverse auditing** is configured at three levels: environment, table and column; turn on the environment first, then each table and column whose changes must be logged. Environment settings include *Log access* (sign-in access) and *Read logs* (sent to Purview, for production environments only). Retention defaults to **forever**, with a custom maximum of 24,855 days. System Administrator or System Customizer rights are needed. Audit history answers who created or updated a record, which fields changed, previous values and who deleted it; it is not available through the Dataverse TDS (SQL) endpoint.

Because Copilot Studio agents, knowledge configuration and tools are Dataverse solution components, a disciplined ALM process (solutions, pipelines, GitHub source control and deploy-from-Git with deployment history) gives you a second, version-controlled audit trail of *what* changed, complementing Purview's record of *who* changed it.

### 7.8.4 Foundry and Azure

- **Azure activity log** records control plane operations (creating or deleting a Foundry resource, deploying a model, changing network or key settings, role assignments). It is the only place that stores the creator of a resource and is retained for **90 days** by default; export it with a diagnostic setting to a Log Analytics workspace (table `AzureActivity`), storage or Event Hubs for longer retention and SIEM integration.
- **Resource logs** capture data plane operations and must be enabled with diagnostic settings.
- **Model versioning**: each deployment has an upgrade policy: *Opt out of automatic model version upgrades* (manual; the deployment stops working when the version retires), *Upgrade once new default version becomes available*, or *Once the current version expires*. For regulated workloads, opt out of automatic upgrades, pin the version, and promote new versions through your evaluation gate so each change is deliberate and logged.
- **Infrastructure as code** (Bicep or Terraform) for deployments, guardrails and network settings gives a reviewable history of model configuration in Git.
- **Microsoft Entra sign-in and audit logs** record agent identity authentication and changes to agent identities and blueprints.

### 7.8.5 Audit source map

| Change or activity | Primary audit source | Retention consideration |
|---|---|---|
| **Copilot Studio agent created, edited, published, shared** | Purview Audit (Copilot Studio events); solution history in Git | Purview audit retention per licence and policy |
| **User and agent conversations** | Purview Audit `CopilotInteraction` plus DSPM for AI for text; Copilot Studio transcripts in Dataverse | DLM retention policies; transcript access rule |
| **Dataverse record changes used for grounding** | Dataverse auditing (table and column) | Default forever; custom up to 24,855 days |
| **Data policy or environment setting changes** | Purview Power Platform activity logs | Purview audit retention |
| **Foundry model deployment, version, guardrail, network changes** | Azure activity log; IaC in Git | 90 days unless exported |

| Change or activity | Primary audit source | Retention consideration |
|---|---|---|
| **Fine-tuning data and jobs** | Activity log for control plane, resource logs and storage logs for data access | Export via diagnostic settings |
| **Agent identity sign-ins and changes** | Microsoft Entra sign-in and audit logs | Export to Log Analytics or Sentinel |
| **Agent threats** | Defender for Cloud and Defender XDR alerts, Sentinel | Incident retention in SIEM |

**NOTE**

Customer Lockbox does not cover Copilot Studio security audit telemetry (agent invocation, tool and action calls, policy enforcement decisions) or Agent 365 governance and audit events. If a stem requires Lockbox coverage for "all agent data", that is a gap to call out.

## 7.9 Additional practice questions

**Practice 7.1** MULTIPLE CHOICE

Contoso has 40 Power Platform environments. Several makers have published Copilot Studio agents to public websites with the **No authentication** option, and some of those agents use connectors to internal systems. You need to prevent any new or updated agent in the tenant from being published without user authentication, while allowing makers to keep using manual Microsoft Entra ID authentication for custom websites. What should you do?

A. Configure managed environment sharing limits to block sharing with security groups.
B. Create a tenant-wide data policy that blocks the *Chat without Microsoft Entra ID authentication in Copilot Studio* connector.
C. Create a Conditional Access policy that requires MFA for all cloud apps.
D. Turn off publishing of agents that use generative AI at the tenant level.

**Practice 7.2** MULTIPLE SELECT

Fabrikam hosts its Copilot Studio environment in the Europe region within the EU Data Boundary. The data protection officer requires that all LLM inferencing for the agent occurs inside the EU Data Boundary at all times, including peak periods, and that no prompt text is processed in the United States. You need to configure the environment's generative AI features. Which two actions should you take? Each correct answer presents part of the solution.

A. Clear the **Allow flex routing during periods of peak load** checkbox.
B. Do not enable the **Bing search** checkbox.
C. Change the environment's region to United Kingdom.
D. Enable customer-managed keys for the environment.
E. Turn on Customer Lockbox.

**Practice 7.3** MULTIPLE CHOICE

A data scientist has the **Foundry User** role on a Foundry project. She can upload training files and run a fine-tuning job, but deploying the resulting fine-tuned model fails with an authorisation error. You need to grant the minimum additional built-in role so that she can deploy fine-tuned models, while preserving separation between data plane and control plane roles. Which role should you assign on the Foundry resource?

A. Foundry Agent Consumer
B. Foundry Project Manager
C. Foundry Account Owner
D. Reader

**Practice 7.4** MULTIPLE CHOICE

You are designing a Foundry agent that reads supplier emails and web pages through tools, summarises them and can create purchase requisitions. Security testing shows that hidden text in a web page can instruct the agent to create requisitions. You need a guardrail control that detects this attack on the content returned to the agent. What should you configure?

A. Spotlighting on the model deployment
B. A user prompt attack control at the user input intervention point
C. A document attack (indirect attack) control at the tool response intervention point
D. Protected material detection at the output intervention point

**Practice 7.5** MULTIPLE CHOICE

Before rolling out Microsoft 365 Copilot, an assessment finds that a finance SharePoint site is shared with Everyone except external users. Finance needs the site to remain accessible to its members while permissions are remediated over the next quarter, but the content must stop appearing in Copilot answers and organisation-wide search for other employees immediately. What should you implement?

A. Restricted SharePoint Search with an allow list that excludes the finance site
B. Restricted Content Discovery on the finance site
C. A sensitivity label without the EXTRACT usage right applied to all files on the site
D. A Copilot Studio data policy that blocks SharePoint knowledge sources

**Practice 7.6** MULTIPLE CHOICE

An internal HR agent built in Copilot Studio started answering anonymous users from the intranet. You need to determine who changed the agent's authentication configuration and when. Where should you look?

A. Microsoft Purview Audit, filtering for the `BotUpdateOperation–BotAuthUpdate` activity
B. The Azure activity log for the subscription
C. Application Insights telemetry connected to the agent
D. Dataverse audit history of the Conversation Transcript table

**Practice 7.7** MULTIPLE CHOICE

A compliance investigator found a `CopilotInteraction` event in Microsoft Purview Audit for a Copilot Studio agent and needs to read what the user asked and what the agent answered, together with the resources accessed. What should the investigator use?

A. Export the audit record and read the `BotUpdateDetails` field
B. Data Security Posture Management (DSPM) for AI
C. The Azure activity log
D. Microsoft Entra sign-in logs for the agent identity

**Practice 7.8** MULTIPLE CHOICE

A European insurer is deploying a GPT model in Microsoft Foundry for claims summarisation. Regulation requires prompts and responses to be processed only within the EU, but the workload needs more throughput than a single regional deployment can provide. Which deployment type should you choose?

A. Global Standard
B. Data Zone Standard in an EU-located Foundry resource
C. Regional Standard in a US region with a private endpoint
D. Developer deployment

**Practice 7.9** MULTIPLE CHOICE

Your organisation wants to ensure that, across all development environments in an environment group, makers can share agents only with named individuals as viewers, with a maximum of 20 viewers per agent, and can never grant editing rights. What should you configure?

A. A data policy that blocks the Teams and Microsoft 365 channel connector
B. Environment group rules for agent sharing: clear *Let people grant Editor permissions*, select *Only share with individuals* and set the viewer limit to 20
C. Security roles that remove the Share privilege on the Copilot table for all makers
D. Block the agents in the Microsoft 365 admin center Agent Registry

**Practice 7.10** MULTIPLE CHOICE

Autonomous Copilot Studio agents in your tenant now have Microsoft Entra Agent IDs. The security team wants these agents to be blocked from all resources whenever Microsoft Entra ID Protection rates the agent as high risk. Your tenant has Microsoft 365 E3 and Microsoft Agent 365 licences. What should you create?

A. A Conditional Access policy that targets all agent identities, uses the agent risk condition set to High, and grants Block
B. A Conditional Access policy that targets all users and requires MFA
C. A Power Platform data policy that blocks event triggers
D. An Azure Policy assignment that denies Foundry deployments

**Practice 7.11** MULTIPLE SELECT

A Copilot Studio agent helps employees update supplier bank details in an ERP system. The ERP connector is configured with maker-provided credentials of a service account that has full write access. A review flags excessive agency. Which two changes reduce the risk most directly? Each correct answer presents part of the solution.

A. Switch the connector to end-user credentials so each user's ERP permissions apply.
B. Route bank detail changes through an agent flow with a human approval step before the update is committed.
C. Enable spotlighting on the agent's model.
D. Increase the content filter severity threshold for violence.
E. Share the agent with everyone in the organisation so more users can report issues.

**Practice 7.12** MULTIPLE CHOICE

A bank plans an autonomous agent that pre-screens customer hardship requests and recommends whether to offer payment relief. The project is in the design phase and no components have been built. According to Microsoft responsible AI practice, what should the architect do first?

A. Configure Foundry guardrails with the highest harm severity thresholds.
B. Run the AI Red Teaming Agent against the production endpoint.
C. Conduct an impact assessment that defines intended uses, stakeholders and potential harms, including fairness harms, and review it with the AI council.
D. Publish a transparency note on the customer portal.

## Answers and explanations

**7.1: B.** Blocking the *Chat without Microsoft Entra ID authentication in Copilot Studio* connector in a tenant-scoped data policy prevents publishing agents that use No authentication, while still allowing Authenticate with Microsoft and Authenticate manually. Sharing limits govern sharing, not authentication; Conditional Access only applies when a sign-in happens; turning off generative AI publishing is far broader than required.

**7.2: A and B.** For EU Data Boundary environments, Azure OpenAI processing is already inside the EUDB; flex routing is what allows inferencing outside the boundary at peak, so it must be cleared. Bing search processing happens in the United States for every region, so it must not be enabled. Changing region, CMK and Lockbox do not affect processing location.

**7.3: C.** Deploying a fine-tuned model is a control plane action. Combining Foundry User (data plane) with Foundry Account Owner (control plane) is the documented alternative to the single Foundry Owner role. Foundry Project Manager cannot manage models; Agent Consumer and Reader add no deployment rights.

**7.4: C.** The attack is indirect prompt injection in content returned by tools. Foundry agents support document attack detection at the tool response intervention point (preview). Spotlighting applies only to models through the Chat Completions API, not to agents; a user prompt attack control inspects only user input; protected material addresses copyrighted output.

**7.5: B.** Restricted Content Discovery removes the site from Copilot and organisation-wide search without changing permissions, which suits an interim measure. Restricted SharePoint Search is retiring and new enablement is blocked from July 31, 2026; it also restricts all sites outside the allow list. Removing EXTRACT would also stop Copilot answering finance members. A Copilot Studio data policy does not affect Microsoft 365 Copilot.

**7.6: A.** Copilot Studio logs authoring changes, including `BotUpdateOperation–BotAuthUpdate`, to Purview Audit. The Azure activity log covers Azure resources; Application Insights holds runtime telemetry; transcripts do not record configuration changes.

**7.7: B.** Audit records contain only the transcript thread ID. DSPM for AI retrieves the chat text and links to accessed resources for `CopilotInteraction` events. `BotUpdateDetails` describes configuration updates, not conversations.

**7.8: B.** Data Zone deployments process prompts and responses only within the data zone (the EU Data Boundary for EU resources) while offering more capacity than a single region. Global may process anywhere; a US regional deployment violates the requirement; Developer is for fine-tuned model evaluation.

**7.9: B.** Agent sharing limits are managed environment controls that can be applied at scale as environment group rules: clearing Editor sharing, restricting to individuals and setting a viewer cap meets every requirement. Removing the Share privilege would block all sharing; blocking in the Agent Registry stops use entirely.

**7.10: A.** Conditional Access for agents can target agent identities with the agent risk condition (preview) and a Block grant. The licensing (Agent 365 with Microsoft 365 E3 or Entra P1) is satisfied. A policy targeting users does not apply to agent identities.

**7.11: A and B.** End-user credentials remove the confused deputy (each user acts with their own rights), and a human approval gate bounds the consequence of an erroneous or injected action. Spotlighting and violence thresholds do not address agency, and broader sharing increases exposure.

**7.12: C.** The responsible AI lifecycle starts with identifying harms through an impact assessment, which then drives measurement and mitigation. Guardrails, red teaming against production and transparency publications come later and depend on the identified harms.

### Key takeaways

- Design security in six layers: identity, data and grounding, tools and connectors, model and guardrails, network and platform, monitoring and response. Put each control at the lowest layer that fully closes the risk.
- Copilot Studio authentication: No authentication, Authenticate with Microsoft (default) or Authenticate manually. Enforce authentication with a data policy on the *Chat without Microsoft Entra ID authentication* connector.
- Microsoft Entra Agent ID gives agents first-class identities (blueprint, agent identity, optional agent user, human sponsor). Copilot Studio creates Agent IDs for new agents since May 2026; Conditional Access for agents needs Agent 365 plus Entra P1 or E3, or Microsoft 365 E7.
- Govern Copilot Studio at tenant, environment and agent level; use data policies, ACP allowlists, managed environments, environment group rules, sharing limits and a zoned environment strategy. Inventory and lifecycle live in the Microsoft 365 admin center Agent Registry and Agent 365 (GA May 1, 2026).
- Foundry model security: least-privilege Foundry roles (renamed from Azure AI roles), keyless Entra authentication with local auth disabled, private endpoints and managed or BYO VNet, guardrails at four intervention points, Defender for Cloud AI threat protection.
- Prompt manipulation is mitigated by classifiers (Prompt Shields, built-in UPIA and XPIA protection) and bounded by deterministic controls (least privilege, end-user credentials, data policies, egress limits, human approval).
- Oversharing is fixed at the source: SAM reports, Restricted Content Discovery as interim protection, permission remediation, sensitivity labels and Purview DLP for Copilot. Restricted SharePoint Search is retiring.
- Responsible AI review: six principles, impact assessment first, then measure, mitigate and operate; use application cards and transparency notes; make an AI council and named owners accountable.
- Residency: separate storage at rest from processing. Know the Power Platform region table, irreversibility of data movement, Bing in the US, flex routing in the EUDB, EUDB versus Multi-Geo, and Foundry Global, Data Zone and Regional types.
- Audit: Purview Audit for Copilot Studio and Copilot (DSPM for AI for text), Dataverse auditing at environment, table and column level, Azure activity log (90 days unless exported) for Foundry, Entra logs for agent identities, Git and IaC for versioned change history, and pinned model versions for controlled upgrades.

### Microsoft Learn references

- Study guide for Exam AB-100: Agentic AI Business Solutions Architect
- Key concepts - Copilot Studio security and governance
- Manage your Copilot Studio projects, an overview
- Implement a zoned governance strategy
- Secure your Copilot Studio projects
- Security FAQs for Copilot Studio
- Configure user authentication in Copilot Studio
- Configure authentication for agents (preview)
- Configure data policies for agents
- Add unstructured data as a knowledge source
- Add Power Platform connectors as knowledge (preview)

- Advanced connector policies
- Control how agents are shared
- Limit sharing
- Rules for environment groups
- Allow external language models for generative responses
- Automatic security scan in Copilot Studio
- Enable external threat detection and protection for Copilot Studio custom agents (preview)
- Microsoft Entra Agent IDs for Copilot Studio agents
- Audit Copilot Studio activities in Microsoft Purview
- Configure data movement across geographic locations for generative AI
- Geographic data residency in Copilot Studio
- Use SharePoint content for generative answers
- Add Dataverse tables as a knowledge source
- Application Card: Microsoft Copilot Studio
- Move data across regions for Copilots, AI agents, and generative AI features
- IP firewall in Power Platform environments
- Manage your customer-managed encryption key
- Ownership-based security in Microsoft Dataverse
- Security concepts for developers (Dataverse)
- Manage Dataverse auditing
- Power Platform activity logs and auditing in Microsoft Purview
- FAQ for Copilot data security and privacy for Dynamics 365 and Power Platform
- What is Microsoft Entra Agent ID?
- Agent identities in Microsoft Entra Agent ID
- Manage agent identities in your organization
- Governing agent identities
- Conditional Access for agents
- Secure autonomous agents with Conditional Access
- Secure agents that act as users with Conditional Access
- Microsoft Entra licensing
- Overview of Microsoft Agent 365
- Microsoft Agent 365 service description
- Governance and lifecycle actions for agents in Microsoft 365 admin center
- Manage agent registry in Microsoft 365 admin center
- Pillar 3: AI governance and security (agent adoption maturity model)
- Role-based access control for Microsoft Foundry
- Configure keyless authentication with Microsoft Entra ID
- Disable local authentication in Foundry Tools
- How to configure network isolation for Microsoft Foundry
- Configure managed virtual network for Microsoft Foundry projects
- Set up private networking for Foundry Agent Service
- Agent identity concepts in Microsoft Foundry
- Guardrails and controls overview in Microsoft Foundry
- Intervention points
- Prompt Shields in Microsoft Foundry
- Manage compliance and security in Microsoft Foundry
- Deployment types for Microsoft Foundry Models
- Model versions in Microsoft Foundry Models
- Data, privacy, and security for Models sold by Azure in Microsoft Foundry
- Customer-managed key encryption in Microsoft Foundry
- AI Red Teaming Agent
- Planning red teaming for large language models and their applications

- Overview of responsible AI practices for Azure OpenAI models
- Document-level access control in Azure AI Search
- Security filters for trimming results in Azure AI Search
- AI threat protection in Microsoft Defender for Cloud
- Alerts for AI services
- Detect and investigate threats to AI agents using Microsoft Defender (Preview)
- AI agent shared responsibility model
- Defend against indirect prompt injection attacks
- Excessive agency (agents)
- Activity log in Azure Monitor
- Guidance to set up your organization's AI governance process
- Artificial Intelligence overview (Microsoft responsible AI program)
- Data, Privacy, and Security for Microsoft Copilot
- Microsoft Copilot prompt defense in depth
- Application card: Microsoft Copilot (for organizations)
- Data Location FAQ
- Get ready for Microsoft 365 Copilot and agents with SharePoint Advanced Management
- Restricted SharePoint Search
- Use Microsoft Purview to manage data security and compliance for Microsoft Copilot Studio
- Learn about using Microsoft Purview DLP to protect interactions with Microsoft 365 Copilot
- Considerations to manage Microsoft 365 Copilot for security and compliance
- Audit logs for Copilot and AI applications
- Setup tasks for Data Security Posture Management

CROSS-DOMAIN PRACTICE

# Case study practice

**Contoso Industrial: an agentic-first order-to-cash programme**

AB-100 case studies describe an enterprise with several Microsoft products, business goals, and constraints. Each item targets one requirement. The wrong options usually solve the requirement while breaking another, or use a heavier platform than needed.

## C.1 A method for architecture case studies




1. **Current estate matters.** If the company already runs Dynamics 365 and Microsoft 365, prebuilt Copilot features and agents are usually the first answer.
2. **Constraints decide platform.** Residency, custom models, channels, autonomy, and maker skills each point to a different platform.
3. **Deploy-domain items** ask for the process: the right telemetry, the right test metric, the right ALM artifact, the right governance control.
4. **Review before leaving** the case study; you cannot return.

## C.2 Case study: Contoso Industrial

### Overview

Contoso Industrial manufactures pumps and sells through distributors in the EU and North America. It runs Dynamics 365 Sales, Customer Service, Contact Center, Finance, and Supply Chain Management, plus Microsoft 365 E5 with Microsoft 365 Copilot for 2,000 users. The CIO wants an agentic-first order-to-cash process within 12 months.

### Current environment

- Product manuals and warranty policies are in SharePoint; pricing and orders are in Dynamics 365; quality data is in Microsoft Fabric.
- A legacy distributor portal has no API.
- Makers in each region build Power Apps in a single shared default environment.
- Service agents complain that Copilot answers in Customer Service use internal jargon inconsistently.
- A data science team has labelled 200,000 historical warranty claims as valid or fraudulent.

### Requirements

- (K1) Sales representatives must get account summaries and next-best actions inside the tools they already use.
- (K2) Incoming warranty claims must be triaged automatically when they arrive, with humans approving payouts above EUR 5,000.
- (K3) Distributor order status must be retrieved from the legacy portal without new development on the portal.
- (K4) Fraud scoring must use Contoso's labelled claims and reach a precision of at least 0.9.
- (K5) EU customer data processed by AI must stay within the EU.
- (K6) Agents must move from development to production through a repeatable, audited process.
- (K7) The business must see, monthly, whether agents resolve cases and how much time they save.
- (K8) Agents must resist instructions hidden in customer emails and attachments.

## Requirement-to-design map

| Requirement | Design | Why the tempting alternative fails | Ch. |
|---|---|---|---|
| **K1 sales assistance** | Copilot in Dynamics 365 Sales and Microsoft 365 Copilot sales capabilities, configured with connectors to Contoso data | A custom Foundry agent duplicates prebuilt capability and adds cost. | 3, 4 |
| **K2 automatic triage with approval** | Copilot Studio autonomous agent triggered on new claims, with an approval step (human in the loop) above the threshold | A prompt-and-response agent waits for a user; fully autonomous payouts break the approval rule. | 3 |
| **K3 legacy portal** | Computer Use in Copilot Studio to operate the portal's UI | A custom connector needs an API the portal does not have. | 4 |
| **K4 fraud scoring** | Custom model trained on labelled claims (AI Builder or Foundry), validated against precision and recall criteria before release | A generic LLM prompt cannot be validated against the labelled ground truth as reliably. | 2, 5 |
| **K5 EU residency** | EU-located environments and Foundry Data Zone (EU) deployments; review cross-region data movement settings for generative features | Global deployments may process data in any region. | 7 |
| **K6 repeatable releases** | Solutions with environment variables and connection references; Power Platform pipelines or GitHub Actions; separate dev, test, prod environments per region | Building directly in the shared default environment has no promotion path. | 6 |
| **K7 business visibility** | Copilot Studio analytics (resolution, engagement, satisfaction, savings) plus Application Insights telemetry reported monthly | Counting sessions alone does not show resolution or value. | 5 |
| **K8 injection resistance** | Prompt Shields and content safety on model calls, least-privilege actions, and human approval for high-impact actions | Longer system prompts do not reliably stop indirect injection. | 7 |

## Practice items

### Practice C.1 MULTIPLE CHOICE

You need to meet K2. Which agent design should you recommend?

A. A prompt-and-response agent published to Teams
B. An autonomous agent with an event trigger for new claims and a human approval step above EUR 5,000
C. A declarative agent in Microsoft 365 Copilot
D. A Power BI report of new claims

### Practice C.2 MULTIPLE CHOICE

You need to meet K3 with the least development effort. What should you use?

A. A custom connector to the portal's REST API
B. Computer Use in Copilot Studio
C. An MCP server built by the portal vendor
D. A nightly CSV export

### Practice C.3 MULTIPLE RESPONSE

You need to meet K6. Which two actions should you include? Each correct answer presents part of the solution.

A. Package agents, flows, and connection references in solutions with environment variables
B. Continue building in the default environment and share agents to production users
C. Deploy managed solutions to test and production through Power Platform pipelines or a CI/CD tool
D. Export agents as screenshots for documentation
E. Give all makers system administrator in production

### Practice C.4 MULTIPLE CHOICE

Service agents report inconsistent jargon in Copilot answers in Customer Service. What should you configure first?

A. Business terms (glossary) for Copilot in Customer Service
B. A new fine-tuned model
C. A larger context window
D. Computer Use

### Practice C.5 MULTIPLE CHOICE

The fraud model must reach a precision of at least 0.9 (K4). What does precision measure?

A. Of claims predicted fraudulent, the proportion that are actually fraudulent
B. Of all fraudulent claims, the proportion the model found
C. The proportion of all predictions that are correct
D. The model's latency

**Practice C.6** MULTIPLE CHOICE

Regional makers currently share the default environment. Which governance change best supports K5 and K6?

A. Keep one environment and rely on naming conventions

B. Create regional developer, test, and production environments as managed environments with DLP policies, and route makers to their own development environments

C. Disable Copilot Studio for all makers

D. Move all data to one global region

## Answers and explanations

**C.1: B.** Autonomous agents act on triggers without a user prompt; the approval step preserves human judgement for high-value payouts.

**C.2: B.** Computer Use operates application and website user interfaces when no API exists.

**C.3: A and C.** Solutions with environment variables and connection references make agents portable; managed solutions deployed through pipelines give a repeatable, auditable path.

**C.4: A.** Business terms teach Copilot organisation-specific vocabulary without building a model.

**C.5: A.** Precision is true positives divided by predicted positives; recall (B) is true positives divided by actual positives.

**C.6: B.** Regional environments keep data in region; managed environments, DLP, and personal developer environments provide governance and a clean promotion path.

EXAM DAY

# Exam technique and signal words

Turning knowledge into points

## E.1 Signal words that map to answers

| If the scenario says... | Think first of... |
|---|---|
| **Fastest value, standard process, already licensed** | Prebuilt Copilot features and agents in Microsoft 365 or Dynamics 365 |
| **Employees in Teams/Outlook, Microsoft 365 data, existing permissions** | Extend Microsoft 365 Copilot (declarative agent, connectors) |
| **Makers, connectors, many channels, admin governance** | Copilot Studio |
| **Custom model, specific model control, complex orchestration** | Microsoft Foundry |
| **Runs when an event happens, no user prompt** | Autonomous agent with triggers |
| **Completes actions for a user** | Task agent |
| **Answers questions only** | Prompt and response agent |
| **Agents on different platforms must collaborate** | Agent2Agent (A2A) / connected agents |
| **Reuse tools from an external server in a standard way** | Model Context Protocol (MCP) in Copilot Studio |
| **Automate an app or website with no API** | Computer Use |
| **Predictable, deterministic intents with few variations** | Classic topics with NLU or CLU |
| **Open-ended requests across many topics and tools** | Generative orchestration |
| **No topic matches** | Fallback / conversational boosting system topic |
| **Organisation-specific vocabulary in Copilot answers** | Business terms |
| **Reusable single-purpose generative step** | Prompt (prompt action) in Copilot Studio / AI Builder |
| **Narrow, high-volume, low-latency, possibly on-device** | Small language model |
| **Send each request to the best model for cost and quality** | Model router |
| **Build, buy, or extend** | Buy (prebuilt) when standard; extend when close; build when differentiating |

| If the scenario says... | Think first of... |
|---|---|
| **Cost over the full lifetime** | Total cost of ownership (licences, consumption, build, run, change) |
| **Adoption process for AI** | Cloud Adoption Framework AI scenario |
| **Central enablement and standards** | AI Center of Excellence |
| **Agent resolution, engagement, satisfaction** | Copilot Studio analytics |
| **Detailed custom telemetry and queries** | Application Insights |
| **Model quality before release** | Validation criteria: precision, recall, F1, accuracy, groundedness |
| **Move agents between environments** | Solutions, environment variables, connection references, pipelines |
| **Different settings per environment** | Environment variables / deployment settings file |
| **Restrict connectors makers can combine** | DLP policies |
| **Hidden instructions in documents or emails** | Indirect prompt injection; Prompt Shields; least privilege |
| **Who changed the model or data, and when** | Audit trails: Purview audit, Dataverse auditing, activity logs, versioning |
| **Data must stay in a geography** | Regional environments; Data Zone deployments; data movement settings |

## E.2 Distractor patterns

| Pattern | Example |
|---|---|
| Heaviest platform | Foundry custom agent for a scenario a prebuilt Dynamics 365 agent covers. |
| Wrong agent type | Prompt-and-response agent for an event-driven process. |
| Integration mismatch | Custom connector when there is no API; Computer Use when a stable API exists. |
| ALM shortcuts | Unmanaged solutions in production; building in the default environment. |
| Vanity metrics | Session counts instead of resolution rate and time saved. |
| Prompt-only security | System-prompt instructions as the only defence against injection. |

## E.3 Time management

- About two minutes per standard item; reserve 20 minutes for case studies and review.
- Read the "You need to" sentence first, then the scenario for constraints.
- When two options both work, choose the one that is simplest, governed, and aligned with the existing estate.

APPENDIX A

# One-page cheat sheet

## Architecture decisions at a glance

| Decision | Rule |
|---|---|
| **Platform** | Prebuilt (Microsoft 365, Dynamics 365) → extend Microsoft 365 Copilot → Copilot Studio / Power Platform → Microsoft Foundry. Move right only when a requirement forces it. |
| **Agent type** | Prompt and response (answers), task (actions for a user), autonomous (triggers, acts within guardrails), multi-agent (orchestrator plus child or connected agents). |
| **Interoperability** | MCP for tools and data; A2A for agent-to-agent collaboration across platforms; connectors and agent flows for business systems; Computer Use when there is no API. |
| **Copilot Studio orchestration** | Classic NLU/CLU for predictable intents; generative orchestration for open-ended, multi-tool requests; Fallback / conversational boosting for unmatched input. |
| **Grounding data** | Accurate, relevant, timely, clean, available; organised in Dataverse, SharePoint, Fabric, or connectors with permissions intact. |
| **Custom AI** | Prompt first; RAG for knowledge; fine-tune or custom model for specialised behaviour or labelled prediction; SLMs for narrow, fast, low-cost tasks. |
| **ROI** | Benefits (time saved, revenue, quality, risk reduction) vs TCO (licences, consumption, build, integration, run, change management, governance). |
| **Model router** | Routes each prompt to a suitable model to balance quality and cost. |
| **Strategy** | Cloud Adoption Framework AI adoption; AI Center of Excellence for standards, enablement, and reuse; prompt library with owners, versions, and reviews. |
| **Dynamics 365 Copilot** | Business terms for vocabulary; customisations and connectors for data; Contact Center channels for agents; finance and supply chain agents and in-app help knowledge. |
| **Monitoring** | Copilot Studio analytics (sessions, engagement, resolution, satisfaction, savings); Application Insights for custom telemetry; Foundry tracing and evaluations; feedback and backlog analysis. |
| **Testing** | Test sets and evaluation metrics for agents; precision, recall, F1, accuracy for classifiers; groundedness and relevance for generative answers; end-to-end scenarios across apps; Copilot-assisted test case generation. |
| **ALM** | Solutions (unmanaged in dev, managed downstream), environment variables, connection references, deployment settings, pipelines; Foundry agents via code and CI/CD with evaluation gates; data versioning. |
| **Governance** | Managed environments, environment groups and rules, DLP, sharing limits, agent inventory, Entra identities for agents. |
| **Security** | Least privilege, authentication for agents and actions, Prompt Shields, content filters, sensitivity labels, oversharing controls, private networking for Foundry. |
| **Compliance** | Residency (regional environments, Data Zone), data movement settings, audit trails (Purview, Dataverse auditing, activity logs), responsible AI review. |

APPENDIX B

# Glossary

**Terms used across the chapters**

| | |
|---|---|
| **A2A** | Agent2Agent protocol: an open standard for agents on different platforms to discover and collaborate with each other. |
| **Agent flow** | A deterministic automation in Copilot Studio that agents can call or that can run on triggers. |
| **AI Builder** | Power Platform capability for prebuilt and custom AI models and prompts used in apps and flows. |
| **AI hub** | Power Platform area that brings AI capabilities (prompts, models, document processing) together for makers. |
| **Autonomous agent** | Agent that acts on triggers or schedules without a user prompt, within guardrails. |
| **Business terms** | Organisation-specific vocabulary configured so Copilot in Dynamics 365 understands and uses it consistently. |
| **CLU** | Conversational language understanding: trained intent and entity recognition that Copilot Studio can use. |
| **Computer Use** | Copilot Studio capability for agents to operate websites and desktop applications through their user interface. |
| **Connection reference** | Solution component that points to a connection, allowing different connections per environment. |
| **Declarative agent** | Microsoft 365 Copilot agent defined by instructions, knowledge, and actions, running on Copilot's orchestrator. |
| **DLP policy** | Power Platform data policy that controls which connectors can be used together. |
| **Environment variable** | Solution component holding environment-specific values such as URLs or knowledge source locations. |
| **Generative orchestration** | Copilot Studio mode in which a language model chooses topics, actions, and knowledge to answer a request. |
| **Managed environment** | Power Platform environment with premium governance features such as sharing limits and usage insights. |
| **Managed solution** | Locked solution deployed to downstream environments; unmanaged solutions are used for development. |
| **MCP** | Model Context Protocol: an open standard for exposing tools and data to agents. |
| **Model router** | Foundry capability that routes each request to a suitable underlying model. |
| **Prompt Shields** | Protection that detects jailbreak attempts and indirect prompt injection in documents. |
| **Prompt and response agent** | Agent that answers questions or generates content on request without taking actions. |
| **Task agent** | Agent that performs actions for a user, usually within a conversation. |
| **TCO** | Total cost of ownership across the solution's life. |

APPENDIX C

# Hands-on labs

**Ten labs that cover the outline**

| # | Lab | Steps | Chapter |
|---|---|---|---|
| 1 | Platform decision | Pick one process; document why prebuilt, extend, Copilot Studio, or Foundry fits; estimate ROI and TCO. | 1, 2 |
| 2 | Declarative agent | Create a Microsoft 365 Copilot agent with SharePoint knowledge; compare with a Copilot Studio agent. | 1, 4 |
| 3 | Three agent types | Build prompt-and-response, task, and autonomous (trigger) agents in Copilot Studio. | 3 |
| 4 | Topics and orchestration | Design topics with Fallback; switch between classic and generative orchestration; add a prompt action. | 3 |
| 5 | Extensibility | Add a connector, an agent flow, an MCP server, and a connected agent; try Computer Use on a test site. | 4 |
| 6 | Dynamics 365 Copilot | In a trial, enable Copilot in Sales and Customer Service; configure business terms; review finance and supply chain Copilot features. | 3, 4 |
| 7 | Monitoring | Review Copilot Studio analytics; connect Application Insights and run a query; collect feedback. | 5 |
| 8 | Testing | Create a test set for an agent; define validation criteria for a custom model; generate test cases with Copilot. | 5 |
| 9 | ALM | Put an agent, flow, and connection references in a solution; deploy managed to test with a pipeline and a deployment settings file. | 6 |
| 10 | Governance and security | Apply a DLP policy and managed environment rules; enable Purview audit search for agent activity; configure Prompt Shields in Foundry. | 7 |

APPENDIX D

# Final readiness checklist

**Tick every line before booking**

| ☐ | I can... |
|---|---|
| ☐ | Assess where agents add value and review data for grounding readiness. |
| ☐ | Apply the CAF AI adoption process and design an AI and agent strategy. |
| ☐ | Choose between Microsoft 365 Copilot, Copilot Studio, and Foundry, and design multi-agent solutions with A2A and MCP. |
| ☐ | Decide when to build custom models or SLMs, and define prompt library and prompt engineering guidelines. |
| ☐ | Describe the AI Center of Excellence and multi-app Dynamics 365 designs. |
| ☐ | Build an ROI and TCO analysis and a build, buy, or extend recommendation; explain model router. |
| ☐ | Design Copilot in Dynamics 365 customisations, business terms, connectors, and Contact Center integration. |
| ☐ | Design task, autonomous, and prompt-and-response agents, topics, agent flows, and prompt actions. |
| ☐ | Choose NLU, CLU, or generative orchestration and propose Foundry Tools. |
| ☐ | Design extensibility with custom models, Microsoft 365 agents, MCP, Computer Use, reasoning, and voice. |
| ☐ | Orchestrate prebuilt AI in Dynamics 365, Microsoft 365, and Power Platform. |
| ☐ | Monitor, analyse, and tune agents using analytics and telemetry. |
| ☐ | Define test processes, metrics, and validation criteria. |
| ☐ | Design ALM for data, Copilot Studio, Foundry Agent Service, custom models, and Dynamics 365 AI. |
| ☐ | Design security, governance, model security, residency, access controls, and audit trails; mitigate prompt manipulation. |
| ☐ | Score 85% or better on every chapter's practice questions and the free Practice Assessment. |

APPENDIX E

# Official resources

**Where to go next on Microsoft Learn**

- AB-100 study guide (skills measured)
- Agentic AI Business Solutions Architect Expert certification
- Practice Assessment on AI Skills Navigator
- Microsoft Copilot Studio documentation
- Microsoft Power Platform documentation
- Dynamics 365 documentation
- Microsoft 365 Copilot extensibility
- Microsoft Foundry documentation
- Cloud Adoption Framework: AI adoption
- Exam sandbox