



COMMUNITY SUMMIT

Dynamics 365
Business
Central (NAV)

October 11–15, 2026
Gaylord Opryland Resort



The largest independent Microsoft AI & Business Applications User Conference on the planet.



| Building Custom AI Agents in Business Central: From Design Experience to Production AL Code



@ 2026 Dynamic Communities

| Renato Fajdiga



- Renato Fajdiga, Derecjun, Director of Business Products
- From Zagreb, Croatia.
- More than 10 years in Dynamics world in various roles.
- Business Apps lead.
- Microsoft MVP and MCT for Dynamics 365 Business Central.
- Speaker at various conferences and co-founder of BizSource User Group Croatia.



@ 2026 Dynamic Communities

| Milan Milinčević



- 7+ years in Dynamics world in various roles.
- Leading BC development @ Enora
- Microsoft MVP for Dynamics 365 Business Central.
- Speaker at various conferences
- Co-founder of BizSource User Group Croatia.



@ 2026 Dynamic Communities

Agenda



01

Agent Design Experience

Design, test, and iterate on agents inside Business Central

02

MCP + Copilot Studio

Connect external agents to BC data via Model Context Protocol

03

Microsoft Foundry & more

Full model choice, multi-agent orchestration, enterprise governance

Agenda



Following the agent development lifecycle

- 1 Why agents** Agents vs. copilots, and when an agent is the right tool
- 2 Design** Agent Design Experience: identity, instructions, profile, permissions, tasks + demo
- 3 Build** From exported definition to AL extension with the Agent SDK + demo
- 4 Ship** Testing non-deterministic behavior, permission scoping, preview limitations
- 5 Beyond BC** MCP, Copilot Studio and Azure AI Foundry - if time allows

Agents ≠ copilots



What makes an agent different from Copilot chat

Copilot

Conversation, not execution

- ✗ Responds to prompts
- ✗ Stateless conversation
- ✗ Manual data entry remains

Agent

A colleague, not an interface

- ✓ Performs work autonomously
- ✓ Dedicated BC user & session
- ✓ Navigates pages, fills fields, runs actions
- ✓ Asks humans when stuck

Agent or automation?



Not every process needs a reasoning model

Traditional automation

AL code · Power Automate · job queue

- Rules are fixed and fully known
- Input is structured (fields, APIs)
- High volume, zero tolerance for variance
- Every run must give the same result

An agent

Agent Design Experience · Agent SDK

- Input is unstructured (emails, PDFs, free text)
- Steps need judgement within clear boundaries
- Process varies from case to case
- A human review step is acceptable

Rule of thumb: if you can write the IF-THEN, write the code. Agents cost Copilot credits on every step.

The agent development lifecycle



From an idea to an extension

NO-CODE consultants, product owners

PRO-CODE developers



The prototype is the specification: instructions, profile and permissions travel from the sandbox into the extension.

The largest independent Microsoft AI & Business Applications User Conference on the planet.



PART 1 • DESIGN

| Playground



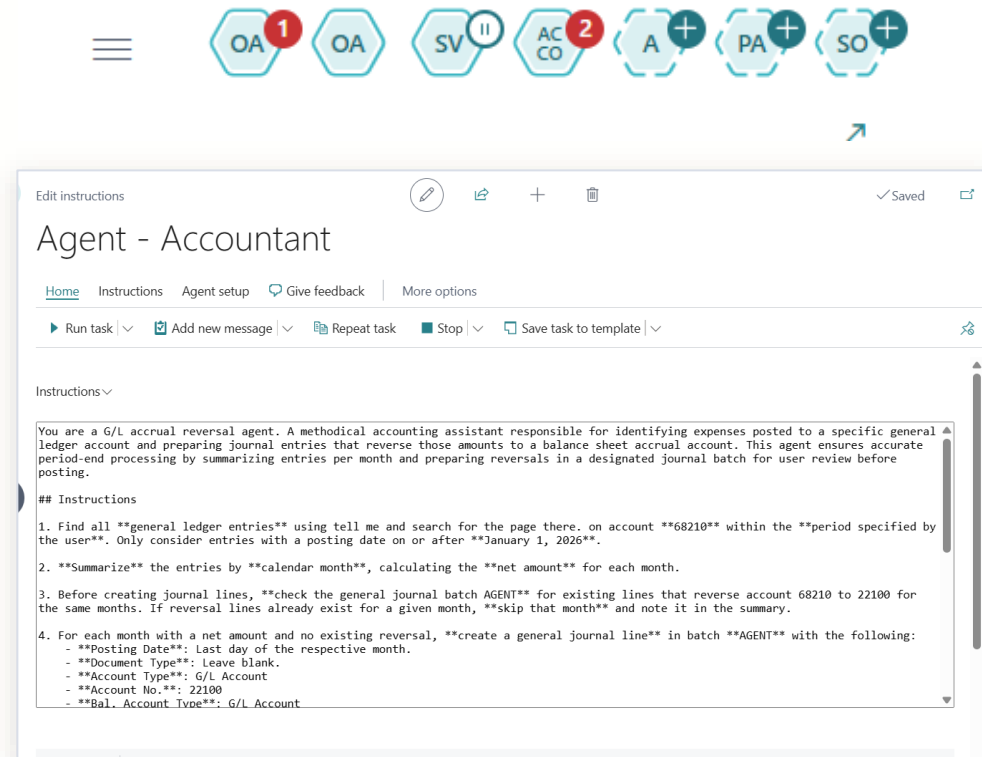
@ 2026 Dynamic Communities

The Agent Playground



A prototyping tool, not a deployment platform

- Write, test and refine agent instructions in a sandbox
- Agents are users in BC
- Test with different profiles and permissions
- Review the AI credit costs
- Must be converted to AL before production use
- Instructions, tasks, messages, ...



Anatomy of an agent



Five things you define in the Agent Design Experience

1

Identity

Name, display name, initials and description. Users see it in the role center.

2

Instructions

Natural language: responsibilities, guidelines and step-by-step tasks.

3

Profile

Which pages, fields and actions the agent sees. Its role center is the starting point.

4

Permissions

Permission sets on the agent user. Least privilege, like any new employee.

5

Tasks

How work arrives: a task plus a message - in the designer typed manually.

Export → definition is exported to XML format.

Writing instructions that work



Responsibility, guidelines, steps

1

Responsibility

What the agent is accountable for - one job.

2

Guidelines

Rules that apply to every task, e.g. always request a review before posting.

3

Instructions

Ordered steps with sub-steps. Name real pages, fields and actions.

Start rough, then let Copilot restructure it. Less is more.

****RESPONSIBILITY****: Check customer credit
on new sales orders

****GUIDELINES****:

- ALWAYS request a review before
blocking an order

****INSTRUCTIONS****:

1. Open the sales order. Memorize the
customer no. and order amount.
2. Available credit = Credit Limit
- Balance.
3. If available credit < order amount:
 - a. ****DO NOT**** release the order
 - b. Ask for assistance with the figures
4. Add a comment line with the result.

Keywords the agent understands



And what breaks instructions

Ask for assistance	stop and hand over to a user
Request a review	pause for confirmation, then continue
Reply / Write an email	outgoing message, always reviewed
Memorize	keep a value for later steps
Set field	enter a value in a field
Use lookup	pick a value from a lookup
Invoke action	run an action, e.g. Release

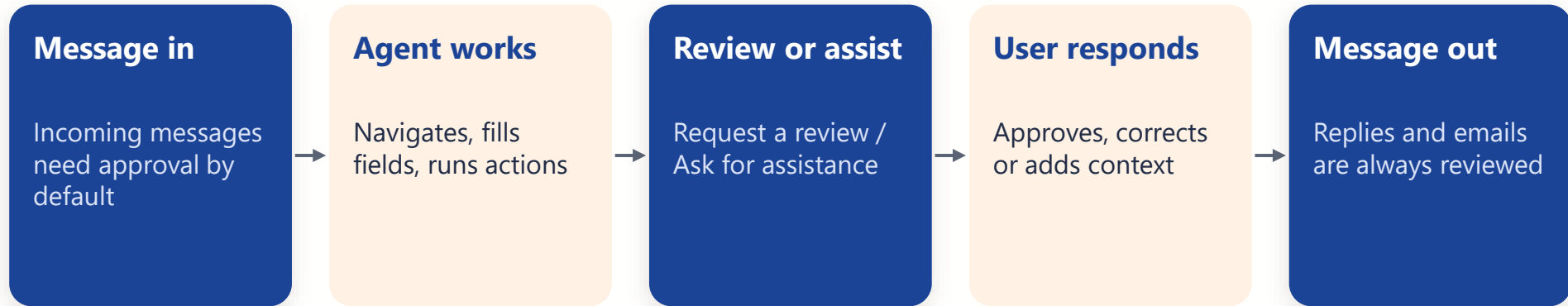
What breaks agents

- Vague or ambiguous steps
- Too many instructions
- Contradicting rules
- Missing business context
- Naming internal tool names instead of the outcome
- Text that looks like a jailbreak to safety filters

Human in the loop



Built into the runtime, not bolted on



Dark steps = a human decides

From 28.1: `SetRequiresReview(false)` skips approval of incoming messages - only for trusted, already validated sources. A Warning annotation from `AnalyzeAgentTaskMessage` still forces a review.

The largest independent Microsoft AI & Business Applications User Conference on the planet.



LIVE DEMO

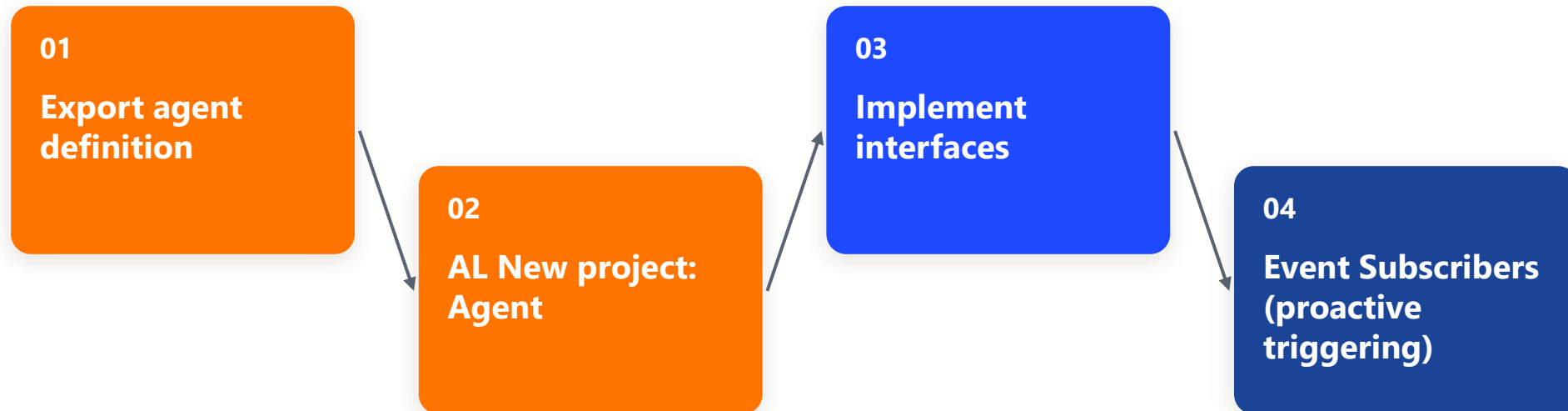
Demo: Building agents in BC



@ 2026 Dynamic Communities

Converting to AL code

Three interfaces for production



Three interfaces, one enum



The Agent SDK contract

```
enumextension 50101 "My Agent Provider"
    extends "Agent Metadata Provider"
{
    value(50101; "My Agent")
    {
        Implementation =
            IAgentFactory = MyAgentFactory,
            IAgentMetadata = MyAgentMetadata,
            IAgentTaskExecution = MyAgentTaskExec;
    }
}
```

- Register the agent type by extending the enum
- Start from AL: New Project > Agent template

IAgentFactory

How instances are created

GetDefaultProfile · GetDefaultAccessControls ·
ShowCanCreateAgent · GetFirstTimeSetupPageId

IAgentMetadata

How an instance presents itself

Initials · setup and summary (KPI) pages · GetAgentAnnotations

IAgentTaskExecution

How tasks are processed

AnalyzeAgentTaskMessage · user intervention suggestions ·
page context

Triggering tasks from AL

No more typing tasks by hand



```
[EventSubscriber(ObjectType::Codeunit,
  Codeunit::"Sales-Post",
  'OnAfterSalesInvHeaderInsert', '', false, false)]
local procedure OnAfterInvoicePost(
  var SalesInvHeader: Record "Sales Invoice Header")
begin
  MessageBuilder.Initialize('System',
    'Review invoice ' + SalesInvHeader."No.");

  AgentTask := TaskBuilder
    .Initialize(AgentUserSecurityId, 'Review invoice')
    .AddTaskMessage(MessageBuilder)
    .Create();
end;
```

Page action

"Send to agent" on a document

Business event

Posting, release, new record

Email / external

SetExternalId to track the thread

Attachments

AddAttachment for PDFs and images

The largest independent Microsoft AI & Business Applications User Conference on the planet.



LIVE DEMO

Demo: Converting to AL code



@ 2026 Dynamic Communities

Testing non-deterministic behavior



Same input, different path. Test for outcomes.

Accuracy

- Verifiable acceptance criteria per scenario
- Many variations of the same case, incl. typos
- Check outcome, not the exact path

Regression

- Every failure from development becomes a test case
- Re-run after each instruction change
- Keep instruction versions in source control

Red teaming

- Ask for data it shouldn't see
- Ask for invalid actions (price 0, then sell)
- Check outgoing messages for data leaks

Cost & performance

- Track steps and credits per task
- Test on realistic data volumes
- Fewer steps = cheaper and more accurate

Preview limitations and your timeline



Check the status on the day you plan a go-live



- Designing agents: sandbox only
- English is the supported language
- Business Central online only
- No built-in triggers in the designer - email, events, schedules need AL

- Copilot credits per step - set up billing first
- Agents can't manage users or other agents
- No access to sensitive pages (user card, permission assignment)
- Prerelease docs - APIs can still change

BC agents live inside BC



That's both the strength and the limit

No conversational UI

Users interact through the task pane, not a chat

Only one AI model

You use whatever BC ships with, no choice

No external channel reach

Can't put it in Teams, a website, or M365 Copilot

No multi-system orchestration

It only works with BC data and pages

The largest independent Microsoft AI & Business
Applications User Conference on the planet.



PART 3 · BEYOND BC

| BC MCP

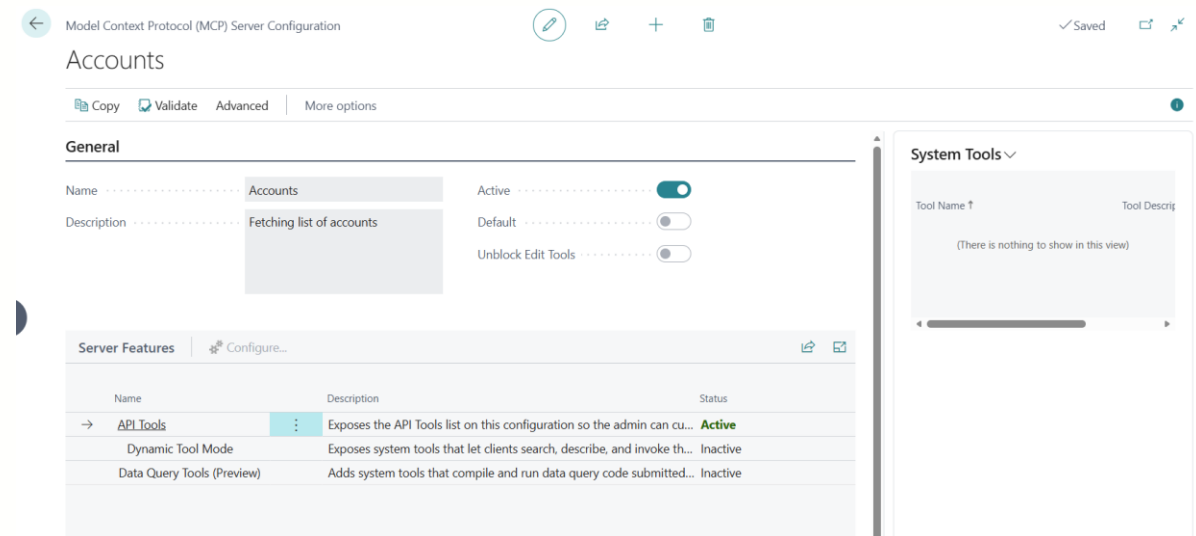


@ 2026 Dynamic Communities

Model Context Protocol (MCP) Server



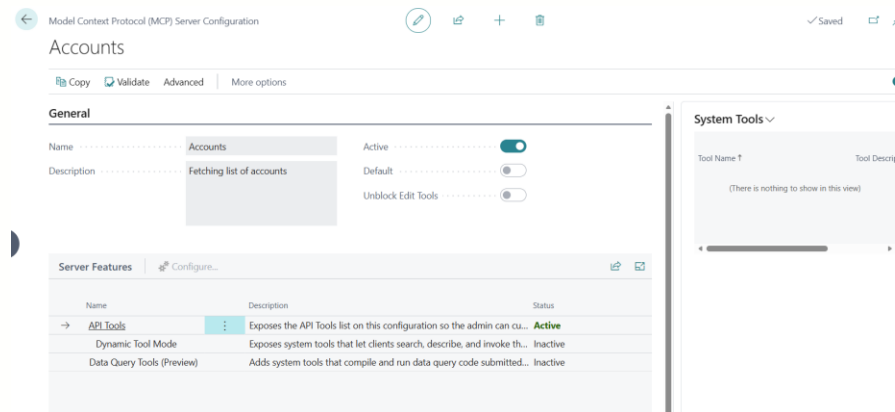
- Exposes existing API pages as tools any AI agent can call
- Authentication through connecting user credentials
- Existing security model applies
- Focused on Microsoft ecosystem (Copilot Studio)
- Possible to use with external systems (i.e. Claude Desktop, VS Code, ...)



BC's native MCP server



THREE SERVER FEATURES – TURN ON WHAT YOU NEED



API Tools

One tool per allowed operation on an API page or query: List_reports, Create_reportRunRequest, run_reportRunRequests...

Dynamic Tool Mode

Three generic tools (search / describe / invoke) instead of one per object – works around client tool limits, e.g. 70 in Copilot Studio

Data Query Tools (preview)

The agent writes an AL query, BC compiles and validates it, rows come back. Read-only, no deployment

Terminology trap: Dynamic Tool Mode is about tool discovery; Data Query Tools are about ad-hoc queries. Two different things.

The largest independent Microsoft AI & Business
Applications User Conference on the planet.



PART 3 · BEYOND BC

| Copilot Studio



@ 2026 Dynamic Communities

What is Copilot Studio?

In plain words, for those who haven't opened it yet



Copilot Studio is where you build an agent without writing code.

You tell it in plain language what the agent is for, give it documents or data to read, connect the systems it may use, and test it in a chat window. When it works, you publish it to Teams, Microsoft 365 Copilot or a website.

It is part of the Microsoft Power Platform, needs its own license, and the agent always acts as the person who is talking to it.

THREE THINGS YOU DO IN IT

Describe

Write what the agent should do, how it should talk, and what it must refuse. That text is the agent's job description.

Connect

Add knowledge (files, SharePoint, websites) and tools (connectors, MCP servers, workflows). Business Central is one of those tools.

Publish

Test in the chat pane, then make it available in Teams, Microsoft 365 Copilot or a website – where people already work.

Why it matters for BC people: the agents from Part 1 live inside the BC client. Copilot Studio is how you put a BC-aware agent in front of people who never open BC.

Copilot Studio <-> BC



Both are tool types on the Build tab

Business Central Connector

Power Platform connector, predefined actions

- Actions such as Find records (V3), Create record (V3), Update record (V3) added as tools
- Fixed inputs per tool: environment, company, API category, table
- Runs as the connected user, respecting BC permissions
- Easy, less flexible; you can edit tool descriptions

USE IT FOR

Standard integration and automation; simple CRUD

Business Central MCP Server

Standard and custom API pages and queries as tools

- Tools follow the MCP configuration: List_ / Create_ / Modify_ / Delete_ and bound actions
- The agent coordinates several steps itself: retrieve, analyze, act
- Custom APIs from your AL extensions work the same way
- Tool list updates automatically when the server changes

USE IT FOR

AI agents only – multi-step orchestration over BC

Use both when it helps: if an API's description is too cryptic for the model to interpret through MCP, add that one API via the connector and write an AI-friendly description.

The new Copilot Studio



One studio, three harnesses – and you pick one per agent

GitHub Copilot harness

THE NEW EXPERIENCE

- Takes a goal, breaks it into steps, calls connectors, MCP and connected agents, recovers when a step fails
- Skills, memory, native Word / Excel / PowerPoint / PDF output
- Production-ready preview; usage billed in Copilot Credits

Standard harness

THE CLASSIC EXPERIENCE

- You define topics, prompts and paths; predictable, rule-based behavior
- Agent flows, the mature feature set, your existing agents
- Fully supported alongside the new one

Copilot chat harness

EXTEND MICROSOFT 365 COPILOT

- Connects enterprise knowledge to Copilot Chat so employees get grounded answers where they already work
- Internal publishing only
- Consumption-based or included in Microsoft 365 Copilot licenses

You choose the harness when you create the agent and there is no conversion between them. Since July 2026 every new agent also gets a Microsoft Entra Agent ID automatically.

Anatomy of a new agent



Everything on one Build tab – instructions first, no topic canvas

BUILD TAB – WHAT YOU CONFIGURE

- 1 **Instructions.** Identity, tone, scope and behavior in natural language – the primary control, no topics.
- 2 **Knowledge + memory.** Connected data sources; Memory keeps details across interactions.
- 3 **Tools.** Connectors, MCP servers and workflows – how the agent acts. BC enters here.
- 4 **Skills.** Reusable Markdown instructions (SKILL.md), loaded on demand; import existing GitHub Copilot or Claude Code skills.
- 5 **Model + connected agents.** Pick the model; delegate specialized tasks to other agents.

Same idea as Part 1: describe the job, connect what the agent needs, set the boundaries; the enhanced orchestration decides when to use knowledge and when to call a tool.

FOUR TABS, ONE AGENT

Build identity, knowledge, tools, skills, model, memory

Preview chat with the agent; activity trace shows reasoning steps and tool calls

Evaluate test sets that measure agent quality

Monitor production health, sessions, reactions, credit usage

The largest independent Microsoft AI & Business Applications User Conference on the planet.



LIVE DEMO

Demo: Copilot Studio + BC MCP



@ 2026 Dynamic Communities

The largest independent Microsoft AI & Business Applications User Conference on the planet.



| When Copilot Studio is not enough...



@ 2026 Dynamic Communities

Why Microsoft Foundry?

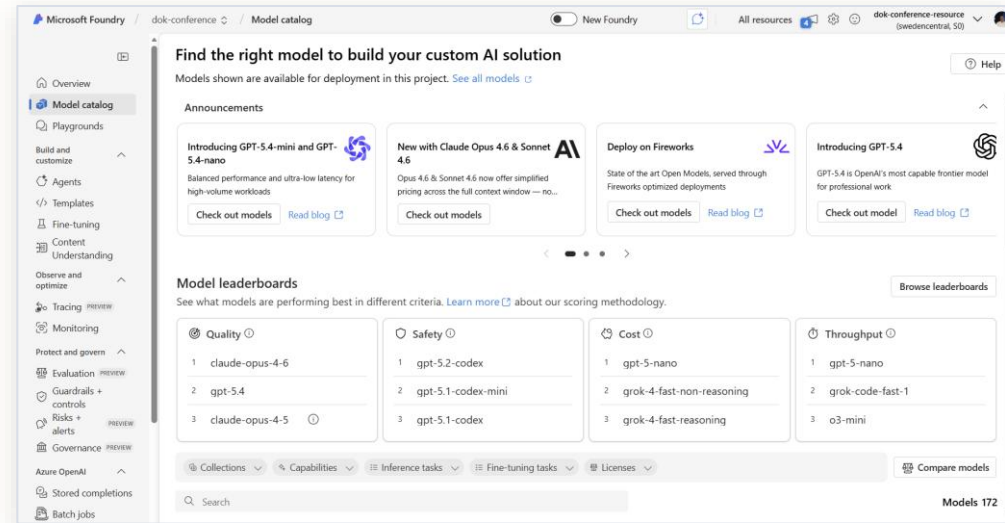


Copilot Studio constraints

- Limited model choice
- 70 tools per agent (even with Dynamic Tool Mode)
- Single-agent only, no multi-agent orchestration

Foundry gives you

- Full model freedom
- Multi-agent workflows with visual orchestration
- Enterprise governance: Entra ID, RBAC, private networking
- End-to-end observability and tracing
- CI/CD pipelines for agent deployment



Foundry Agent Service



Three agent types: from no-code to full custom containers

01

Prompt Agents

No code, fastest to ship

- Instructions + model + tools
- Portal or SDK
- Service handles hosting

02

Workflow Agents

Orchestrate, don't script

- Visual multi-agent builder
- Multi-step processes
- Approval chains & collaboration

03

Hosted Agents

Your code, our runtime

- Agent Framework, LangGraph, any framework
- Managed containers
- Built-in scaling

The largest independent Microsoft AI & Business Applications User Conference on the planet.



| Key takeaways



@ 2026 Dynamic Communities

Key takeaways



1

You have three paths, pick the right one for the project

Playground for prototyping, Copilot Studio for cross-platform chat, Foundry for full enterprise control

2

MCP is the new integration layer for BC

Configure your API pages once, any agent from "any" platform can connect

3

Agents are here, use Agents

Key takeaways



What to do on Monday

- 1 Prototype in the sandbox, ship in AL**

The Design Experience is for learning fast; the extension is what you deploy.
- 2 Treat instructions as code**

Short, structured, versioned in Git, re-tested after every change.
- 3 Least privilege + custom profile**

The agent can only use what it sees and what it is allowed to touch.
- 4 Test outcomes, not paths**

Acceptance criteria, variations, regression cases, red teaming, credits per task.
- 5 Use an agent where judgement is needed**

If you can write the IF-THEN, automate it the traditional way.

Resources



Everything we showed is documented

Microsoft Learn

learn.microsoft.com/dynamics365/business-central/dev-itpro/ai/

AI development toolkit overview

[ai-development-toolkit-overview](#)

Write effective instructions

[ai-development-toolkit-instructions](#)

Best practices for designing an agent

[ai-development-toolkit-best-practices](#)

Coding agents in AL

[ai-agent-sdk-overview](#)

Convert an agent to AL code

[ai-agent-sdk-convert-agent](#)

GitHub

Sales Validation Agent sample

[microsoft/BCTech](#) › [samples/BCAgents](#)

Agent SDK source (System App)

[microsoft/BCApps](#) › [System Application/App/Agent](#)

Instruction patterns

[microsoft/alguidelines](#)

| Q & A



 renato@fajdiga.info | renato@derecjun.com

 milan.milincevic@enora.no

@ 2026 Dynamic Communities

The largest independent Microsoft AI & Business
Applications User Conference on the planet.



**Thank you for
attending! 🙏**



@ 2026 Dynamic Communities