INTEGRATION EVERYWHERE

Supercharge your AI Agents with Custom Connectors

Eric Shupps

Microsoft MVP

Eric Shupps

Microsoft 365 MVP

@eshupps

sharepointcowboy

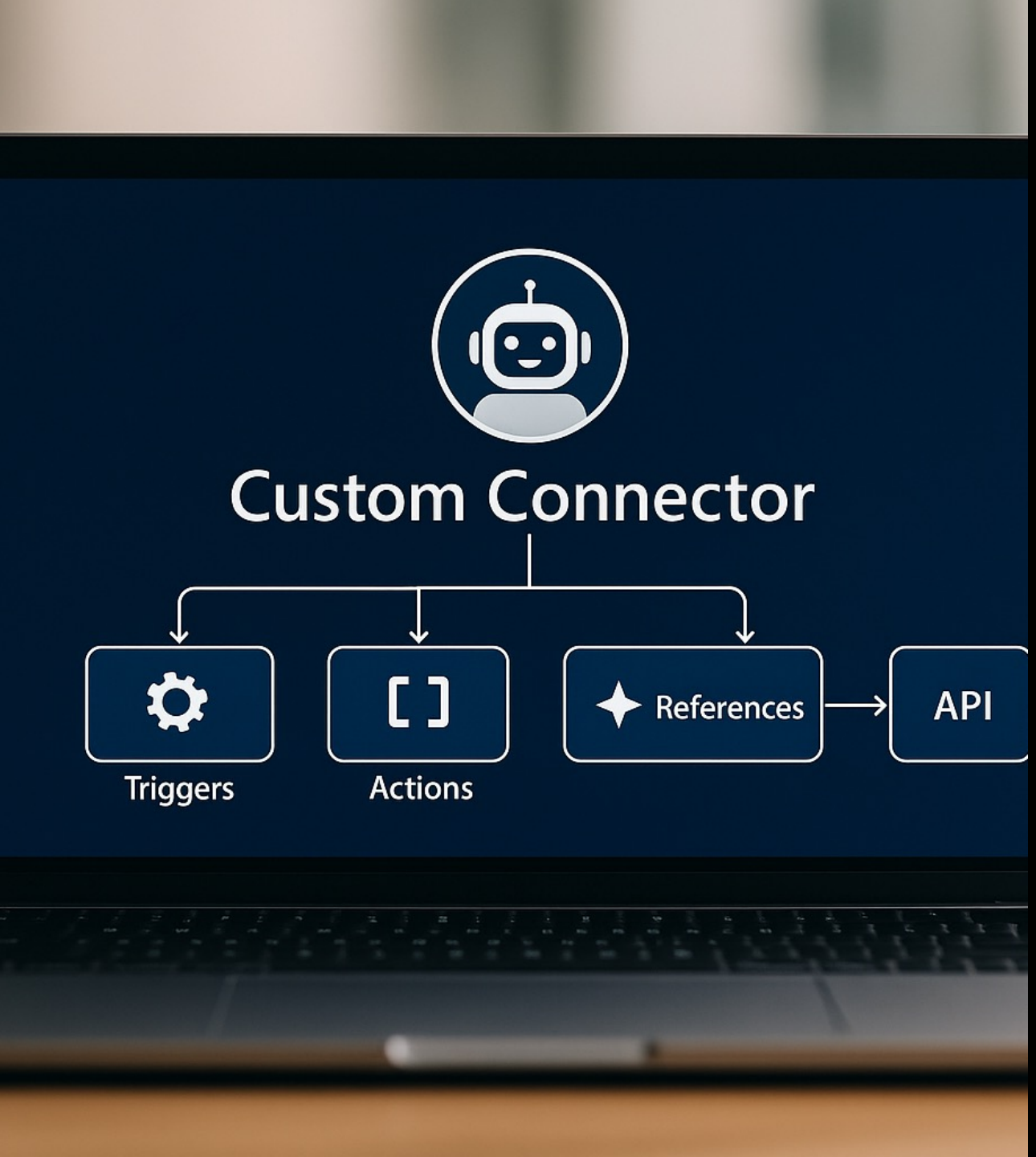
slideshare.net/eshupps

linkedin.com/in/eshupps

github.com/eshupps

Why AI-Ready Connectors?

- Connectors are how we teach Copilot domain-specific and contextual skills.
- Turn APIs into conversational actions.
- Shift from low-code workflows → AI-driven reasoning.
- Integrate with external, legacy, and Line of Business systems directly from an agent
- Leverage existing investments in Power Platform resources

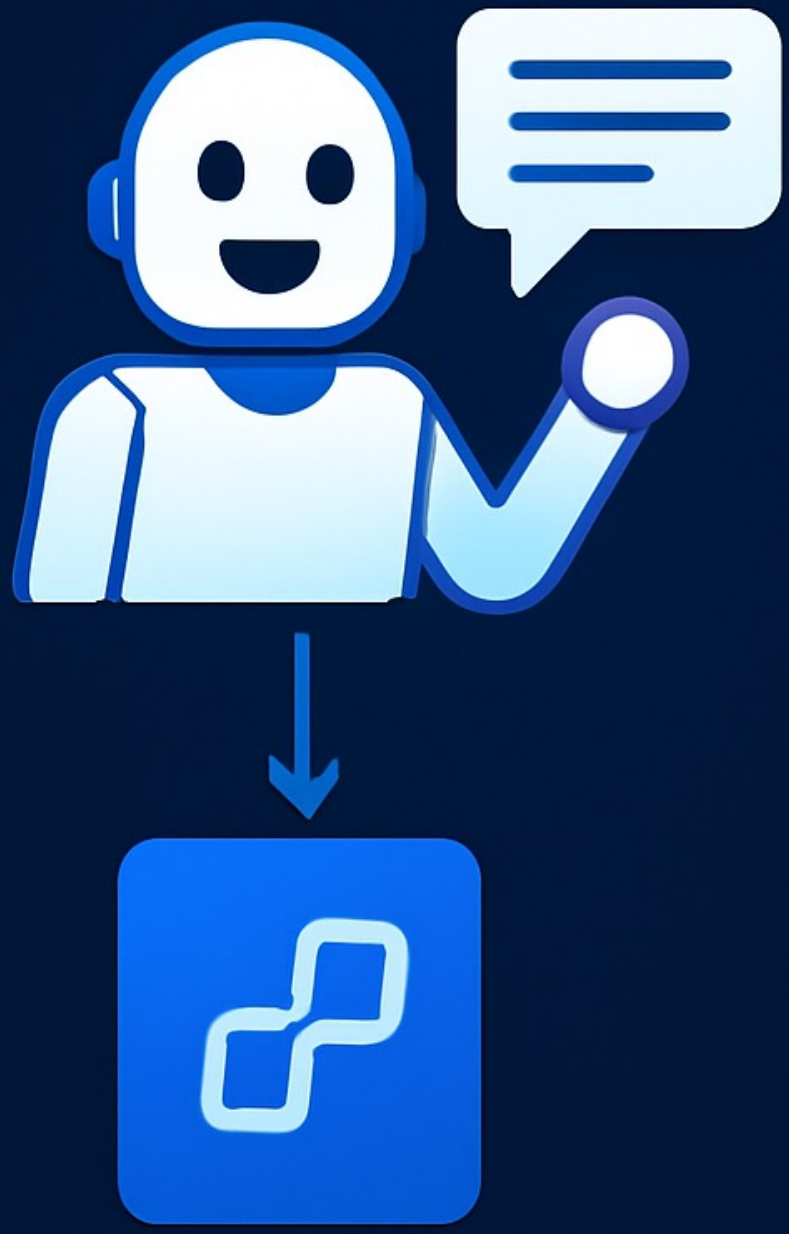


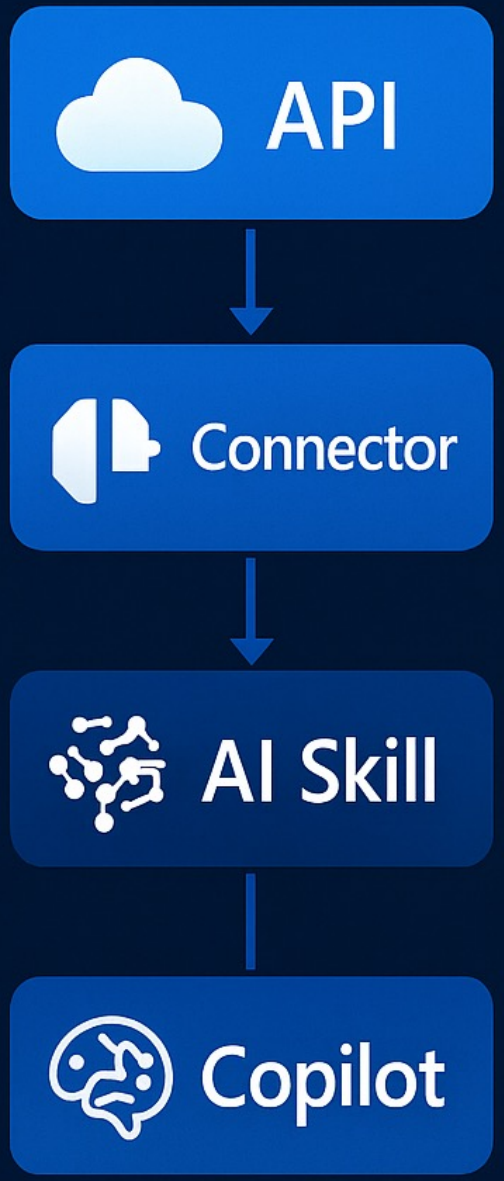
The Connector Framework

- API → The target system.
- Definition → OpenAPI spec (actions, triggers, refs, policies).
- Connector → Wraps the API for Power Platform & Copilot.
- User Interaction → Flows, Apps, and now Conversational AI.

Connector Components

- Information: Title, host, version, security.
- Actions: API operations → exposed as agent skills.
- Triggers: Event cues → surfaced in conversation.
- References: Reusable objects for cleaner specs.
- Policies: Runtime transformations for AI-friendly outputs.
- Code: Programmatic manipulation of inputs and outputs





The AI Shift

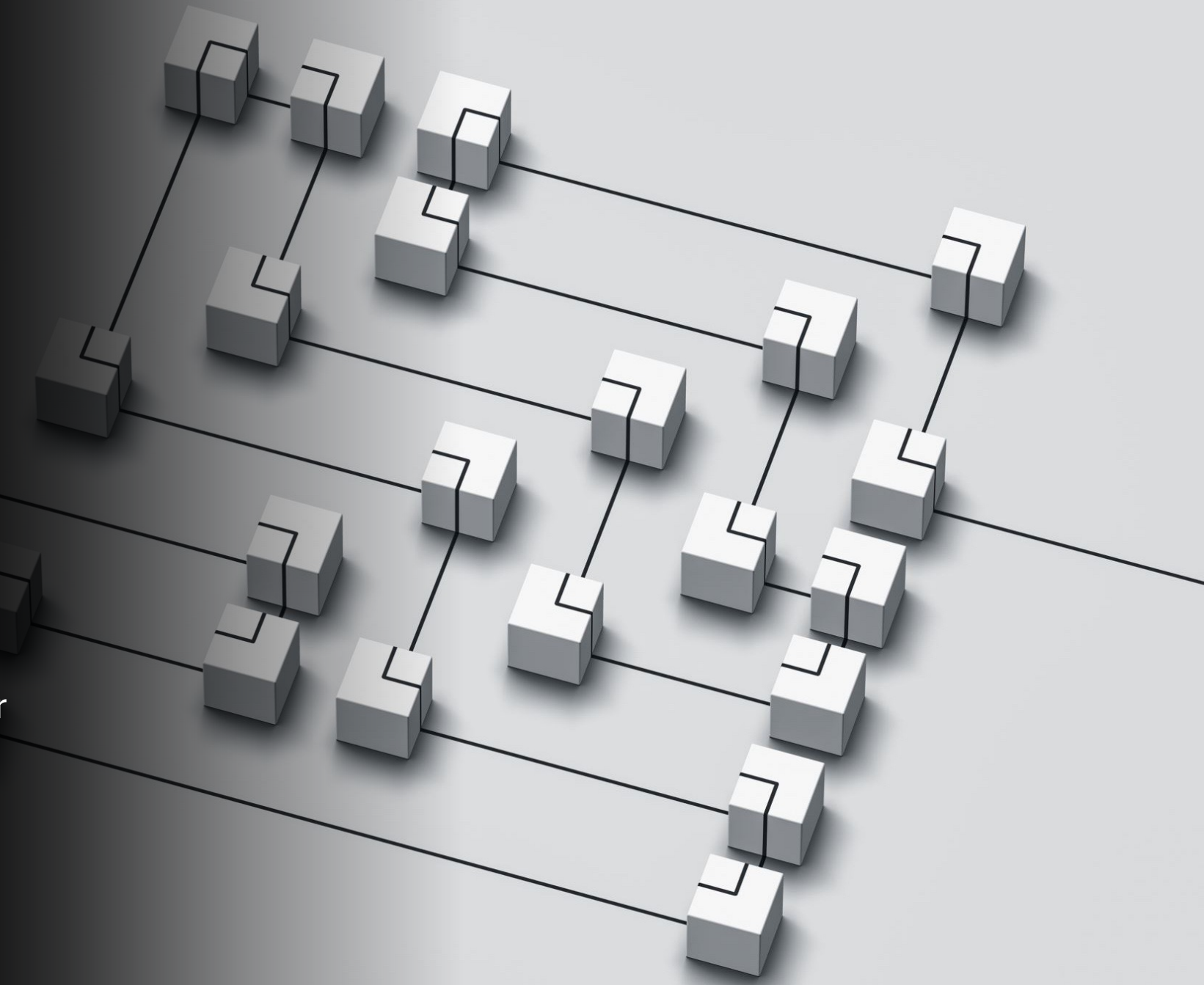
Connectors extend Copilot agent capabilities.

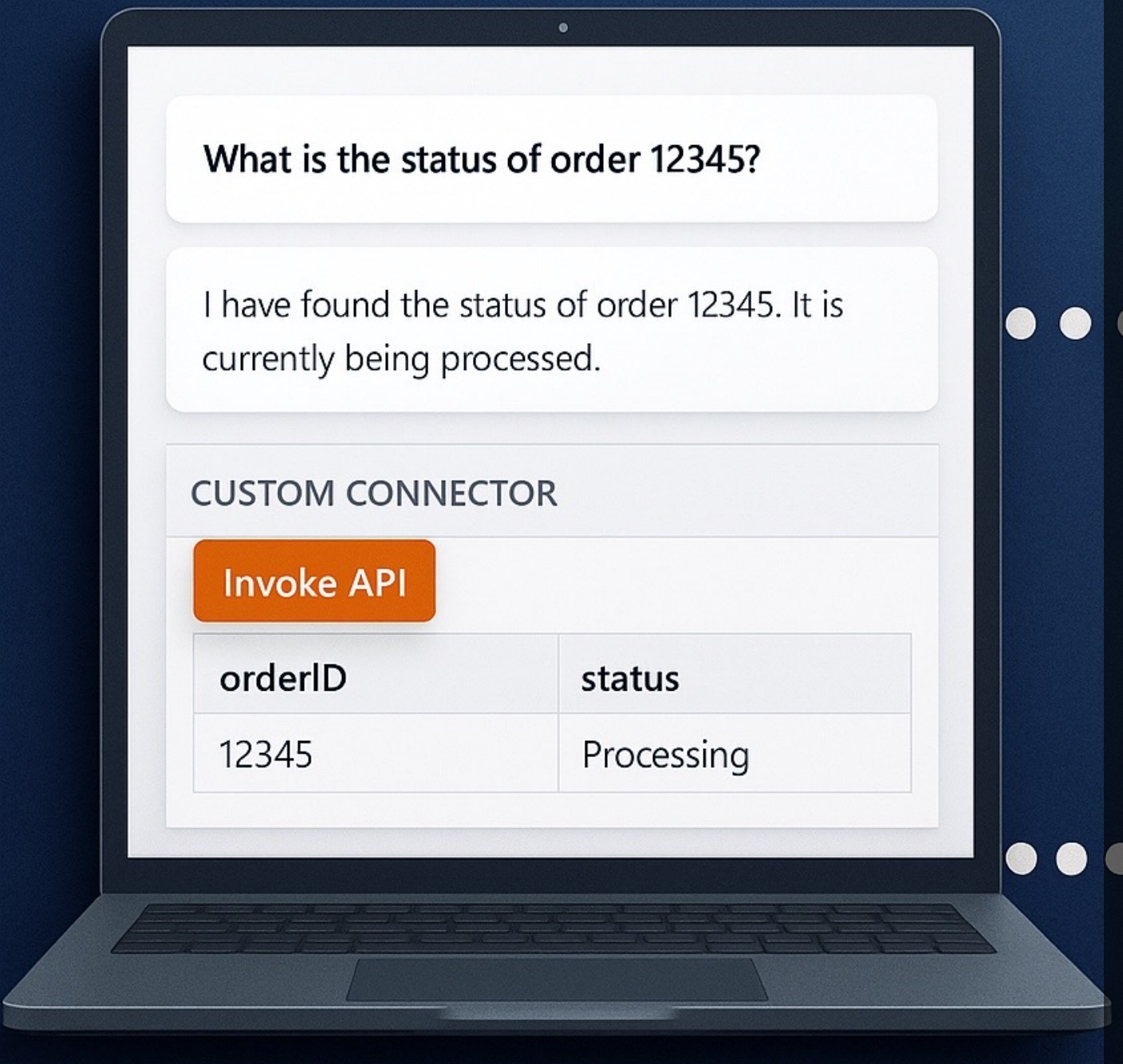
Actions = Paths = conversational intents (exposed as Tools).

Outputs = reasoning inputs.

AI-Enabled Connectors

- x-ms-openai-manifest: adds Copilot readiness.
- Agent actions → appear in Copilot Studio.
- Conversational triggers → enable proactive AI behavior.
- Architecture: API → Connector → Manifest → Copilot Studio → Conversation.

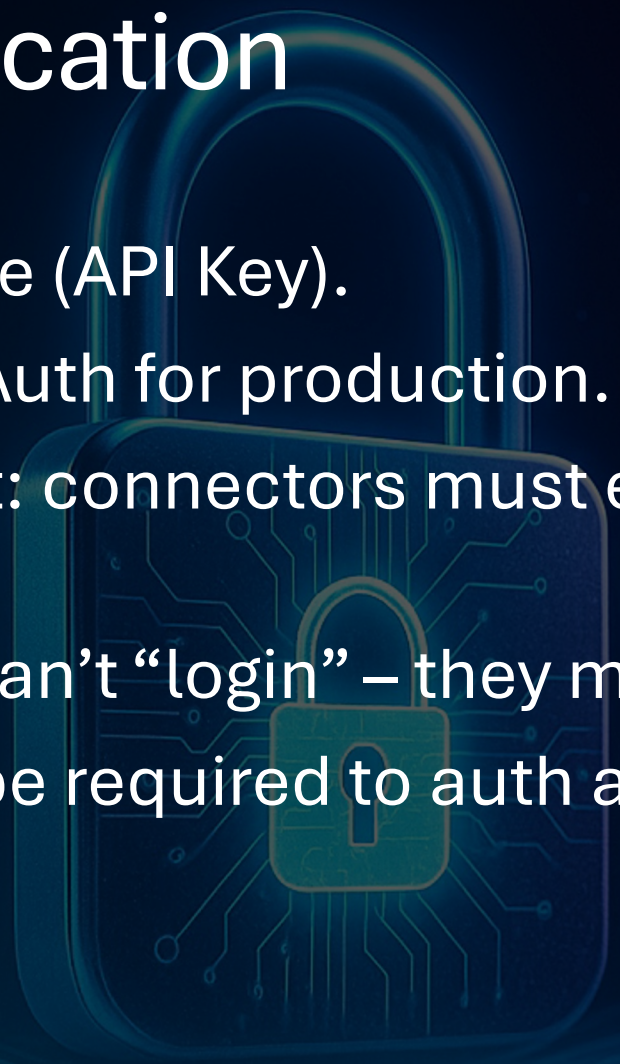




Building AI-Ready Connectors

- Define OpenAPI spec.
- Refine actions (atomic, descriptive).
- Add security config.
- Extend manifest with AI metadata.
- Test in Copilot Studio conversation.

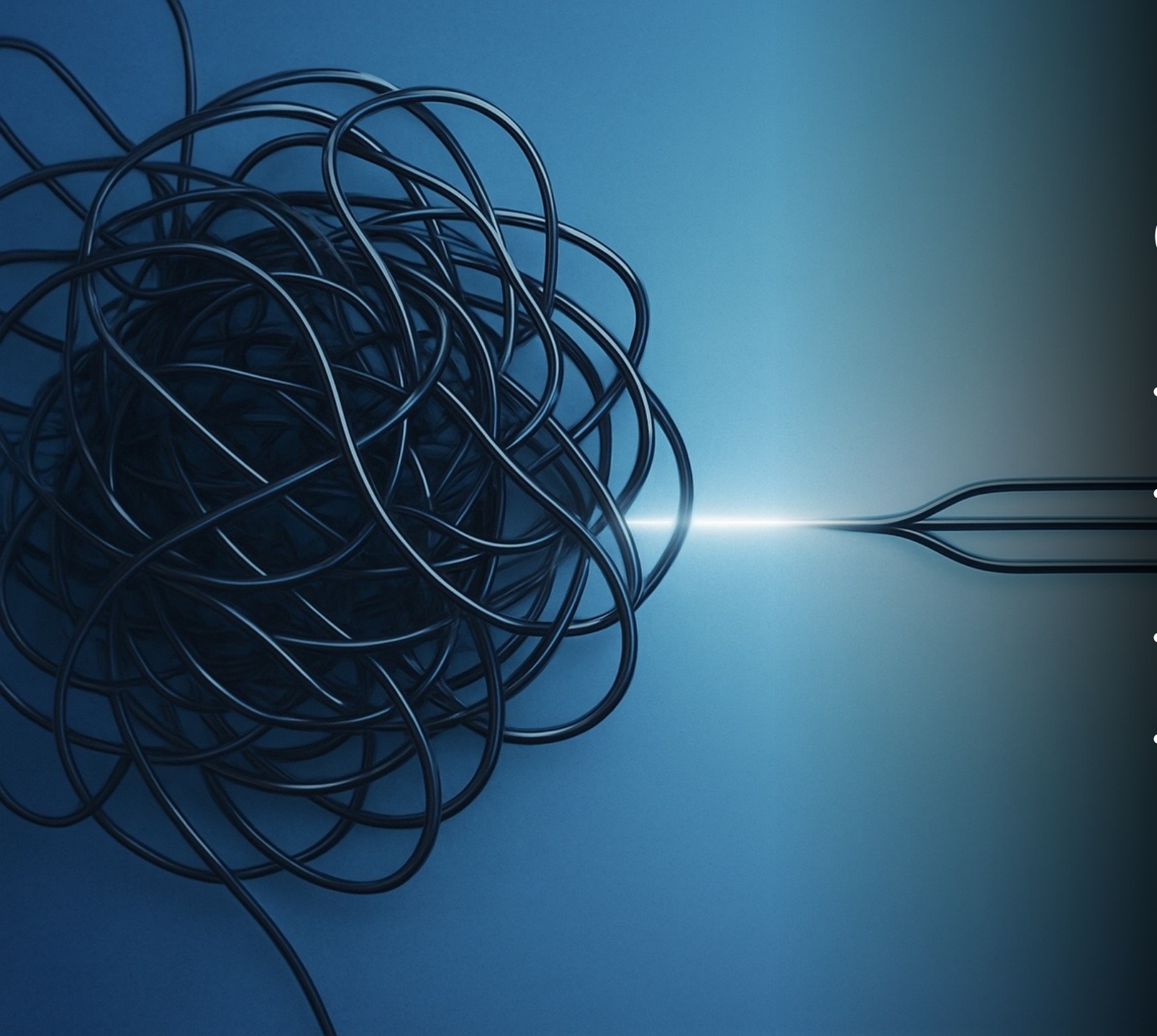
Authentication



- Start simple (API Key).
- Move to OAuth for production.
- For Copilot: connectors must enforce secure authenticated channels.
- AI agents can’t “login” – they must use delegated tokens.
- User may be required to auth as part of conversation flow

DEMO

Building a Custom Connector



Managing Complexity

- References: use #ref to avoid duplication.
- Flattening: break nested objects into simpler responses.
- Modularity: split large APIs into multiple connectors.
- Version control: keep specs manageable, especially for Copilot.

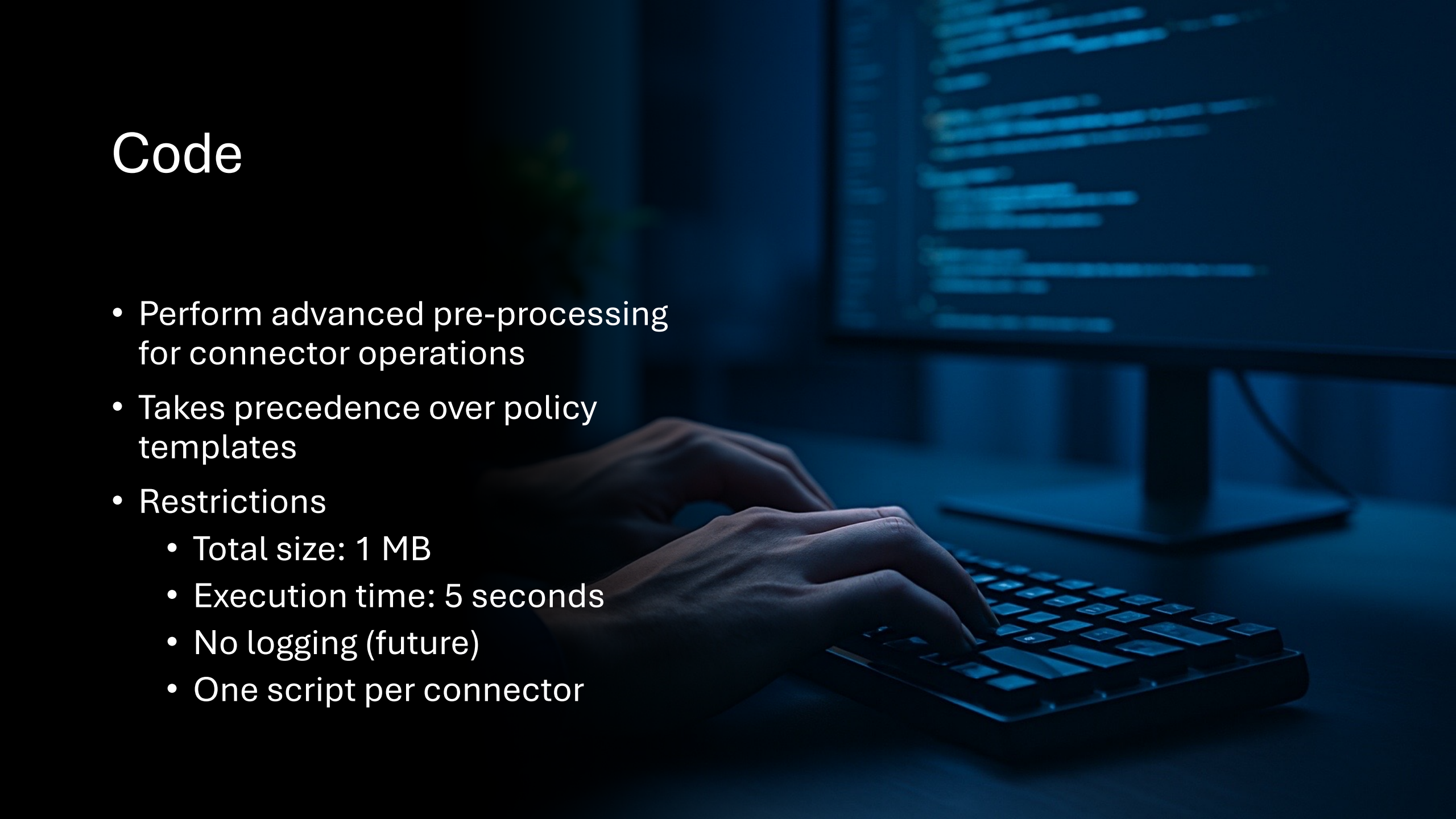
Version Management

- Path-based (/v1, /v2) vs annotation-based.
- x-ms-api-annotation for preview vs production.
- Importance of not breaking conversational models.
- Use only the current version of each path as a tool (Copilot can’t differentiate between versions/deprecation)

Policies

- Modify headers, URLs, methods, objects.
- Transform objects into AI-friendly structures.
- Simplify data for reasoning (AI doesn’t like noise).
- Still hard to debug → test iteratively.

Code



- Perform advanced pre-processing for connector operations
- Takes precedence over policy templates
- Restrictions
  - Total size: 1 MB
  - Execution time: 5 seconds
  - No logging (future)
  - One script per connector

Gotchas

- AI needs small, clearly named actions to reason effectively.
- Overly complex outputs confuse Copilot: Avoid deeply nested JSON or mixed data types.
- Testing differs: In Copilot Studio, you validate conversational behaviors — not just API responses.
- GIGO – The better your definition, the easier it will be for Copilot to “understand” it.
- Conversation context: Copilot may reuse connector outputs in unexpected ways; test with multi-turn dialogs.

Best Practices

- Use clear, natural action names
- Keep outputs short and conversational
- Break APIs into small, atomic skills
- Ensure manifest matches actions
- Test in multi-turn dialogs
- Handle errors with user-friendly messages
- Build with OAuth & security first

Future Proofing

- Build for automation and AI (responses instead of objects).
  - Data-centric objects do not always make for good responses.
- Expect evolving schemas in Copilot Studio.
- Think agent-first: what skills should the AI have?
- Proactively manage tool (action) versioning

References

- eBook: The Ultimate Guide to Power Platform Custom Connectors
  - https://apptigent.com/product/ebook-the-ultimate-guide-to-power-platform-custom-connectors
- Custom connector overview and walkthroughs
  - https://docs.microsoft.com/en-us/connectors/custom-connectors/
- Custom connector OpenAPI extensibility
  - https://docs.microsoft.com/en-us/connectors/custom-connectors/openapi-extensions
- Custom connector connection parameters
  - https://docs.microsoft.com/en-us/connectors/custom-connectors/connection-parameters
- Custom connector policy templates
  - https://docs.microsoft.com/en-us/connectors/custom-connectors/connection-parameters
- Connector Versioning
  - https://docs.microsoft.com/en-us/connectors/custom-connectors/operational-versioning
- Walkthrough
  - https://www.collabmagazine.com/getting-started-with-custom-connectors-for-microsoft-power-automate-and-power-apps/
- Slide Deck
  - https://www.slideshare.net/slideshow/integration-everywhere-supercharge-your-ai-agents-with-custom-connectors-pptx/283922240