Master Variables, Multi-Agent Architectures, and Channel Deployment

Learn how to manage conversation state with variables, build specialized child agents for modular solutions, and deploy your agent across multiple channels to reach users where they work.

🧭 Lab Details

Level	Persona	Duration	Purpose
200	Maker	30 minutes	After completing this lab, participants will be able to create and manage both topic-level and global variables to maintain conversation state, create child agents with specialized knowledge and configure parent agent orchestration, and deploy agents to web and Microsoft Teams channels with appropriate security settings.

📚 Table of Contents

🤔 Why This Matters

Agents need memory, specialization, and accessibility to deliver real business value.

Think of variables, child agents, and channels like organizing a company: - **Without variables**: Your agent forgets information between conversation steps, forcing users to repeat themselves - like a team with no shared notes - **Without child agents**: One agent tries to be an expert in everything, leading to confused responses - like a single employee covering every department - **Without channels**: Your perfectly built agent sits behind a locked door - users can't reach it where they work

With all three: Your agent remembers context, delegates to specialists when needed, and meets users in Teams, on the web, and beyond - like a well-organized company with shared knowledge, clear expertise, and open doors.

Common challenges solved by this lab: - "My agent forgets what users told it earlier in the conversation" - "My agent is overwhelmed trying to be an expert in too many areas" - "My agent is ready but I don't know how to make it available to users" - "I need my agent in Teams, on our website, and in our mobile app"

In 30 minutes, you'll transform your agent into a sophisticated, memory-enabled, multi-agent system deployed where users need it.

🌐 Introduction

Variables, child agents, and channels are the keys to building enterprise-grade agent solutions. Variables act as your agent's memory - storing user information and conversation context. Child agents enable modular architecture where specialized sub-agents handle specific domains. Channels determine where and how users interact with your agent - whether through Teams, web widgets, or custom applications.

Real-world example: An IT support agent collects a user's employee ID and department at the start of the conversation, storing them in variables for personalized responses throughout the interaction. When the user asks about password policies, the parent agent delegates to a specialized "Security Policies" child agent with deep authentication knowledge. The entire experience is available in Microsoft Teams (where employees chat daily), on the company intranet (for quick access), and through a mobile app (for field workers). Variables maintain context, child agents provide expertise, and channels ensure accessibility.

This lab teaches you how to architect sophisticated agent solutions that scale with your organization's complexity - combining conversation memory, specialized intelligence, and strategic deployment.

🎓 Core Concepts Overview

Concept	Why it matters
Topic-Level Variables	Store data within a single topic's scope - perfect for collecting and using information during a specific conversation flow without polluting the global namespace
Global Variables	Maintain data across the entire conversation and all topics - essential for user preferences, session information, and context that persists throughout the interaction
Set Variable Node	Modify variable values during conversation flows - this allows dynamic data transformation, calculations, and state management
Child Agents	Specialized sub-agents with focused knowledge and instructions - these enable modular architecture, domain expertise, and scalable agent development
Agent Orchestration	Parent agents route conversations to appropriate child agents based on context - this creates seamless multi-agent experiences that leverage specialized knowledge
Channels	Distribution points where users interact with your agent - each channel (Teams, web, mobile, custom) has unique capabilities, authentication requirements, and user experiences
Channel Security	Authentication and access controls that determine who can interact with your agent and how - essential for protecting sensitive data and ensuring compliance

📄 Documentation and Additional Training Links

- Use variables to store and reuse information (https://learn.microsoft.com/microsoft-copilot-studio/authoring-variables)
- Create and manage child agents (https://learn.microsoft.com/microsoft-copilot-studio/advanced-use-skills)
- Multi-agent architectures in Copilot Studio (https://learn.microsoft.com/microsoft-copilot-studio/advanced-multi-agent)
- Configure channels for your agent (https://learn.microsoft.com/microsoft-copilot-studio/publication-fundamentals-publish-channels)
- Deploy to Microsoft Teams (https://learn.microsoft.com/microsoft-copilot-studio/publication-add-bot-to-microsoft-teams)

✅ Prerequisites

- Completed Build Intelligent Agents with Knowledge Sources, Tools, and Topics lab - specifically the mailing list topic from Use Case #4, which is the foundation for the variables exercise
- Access to Microsoft Copilot Studio with tool and channel configuration permissions
- Access to Microsoft Teams (for Teams channel testing)
- A document for child agent knowledge (e.g., a prompt engineering guide PDF)

🎯 Summary of Targets

In this lab, you'll implement conversation memory with variables, build a multi-agent architecture with child agents, and deploy your agent across channels. By the end of the lab, you will:

- Create and configure topic-level and global variables with appropriate data types
- Use the Set Variable node to transform and concatenate data during conversations
- Create child agents with specialized knowledge and instructions
- Configure parent agent orchestration rules for proper conversation routing
- Deploy your agent to the web channel with security settings
- Deploy your agent to Microsoft Teams
- Test multi-channel deployment to verify consistent functionality

🧩 Use Cases Covered

Step	Use Case	Value added	Effort
1	Work with Variables	Understand how variables maintain conversation context and how to navigate the variables interface	5 min
2	Create and Orchestrate Child Agents	Build specialized sub-agents with focused expertise to create modular, scalable solutions	13 min
3	Deploy Your Agent Across Channels	Make your agent accessible across web, Teams, and other platforms with appropriate security	12 min

🛠️ Instructions by Use Case

🧱 Use Case #1: Work with Variables

Explore the variables interface in Copilot Studio to understand how conversation state is managed through topic-level and global variables.

Use case	Value added	Estimated effort
Work with Variables	Understand how variables maintain conversation context and how to navigate the variables interface	5 minutes

Summary of tasks

In this section, you'll explore existing variables in your mailing list topic, understand variable properties and scope, review how variables are referenced in messages and formulas, and observe variable values during a test conversation.

Scenario: Your mailing list topic (from the previous lab's Use Case #4) already collects user information and stores it in variables. You'll explore how Copilot Studio automatically creates variables, understand the difference between topic-level and global scope, and observe how variables work during a live test conversation.

> ⚠️
> **WARNING:** This use case is exploration only. Do NOT create, modify, or delete any variables or nodes - doing so could break your existing topic. You are here to navigate the interface and understand how variables work.

Objective

Understand variable types, properties, scope, and behavior by exploring the existing variables interface in your agent.

Step-by-step instructions

Open the Mailing List Topic

1. In your Copilot Studio agent, navigate to **Topics** in the left panel.
2. Open the **Join Copilot Studio Mailing List** topic that you created in the previous lab's Use Case #4.

> ℹ️
> **Note:** If you don't have this topic, you can explore variables in any existing topic that has question nodes.

Review Existing Variables

3. Find the **Question** node where the user's email address is collected.
4. Click on the question node to expand its properties panel.
5. Look for the **Save response as** or **Variable** section. Notice that Copilot Studio automatically created a variable to store the email address when the question node was built.
6. Note the variable name (likely something like `emailAddress` or `userEmail`).

> 💡
> **Tip:** Whenever a question node is created, Copilot Studio automatically creates a variable to store the user's response. This variable is scoped to the topic by default.

Explore Variable Properties

7. Click on the variable name to open the variable properties panel.

8. Review the variable configuration — do not change any values:
    - **Name**: The variable identifier used in the topic
    - **Type**: Data type (Text, Number, Boolean, etc.)
    - **Scope**: Topic-level or Global
    - **Description**: Documentation for the variable's purpose
9. Notice that this variable is **Topic-level** by default, meaning it only exists within this topic.

> ⚠️
> **IMPORTANT: Topic-level variables** exist only within a single topic and reset when the topic ends. **Global variables** persist across all topics and the entire conversation. Global variables use a namespace like `Global.emailAddress`. Use global variables sparingly — only when you need to share data across multiple topics.

View All Variables

10. Click **Variables** in the left navigation panel (or look for a Variables section in your topic).
11. Review the **Topic variables** section showing all variables in the current topic.
12. Review the **Global variables** section showing variables available across the entire agent.

> ℹ️
> **Note:** The Variables view gives you a centralized place to see all variables, their types, and their values during testing.

Understand How Variables Are Used

13. Return to your topic canvas and scroll through the nodes. Look for places where variables are referenced:
    - **Message nodes** may use curly braces like `{Topic.emailAddress}` to dynamically insert variable values into text
    - **Set Variable nodes** can transform data using formulas like `"Text " & Topic.variableName` (the `&` operator concatenates strings)
    - **Condition nodes** can branch logic based on variable values
14. Notice the **+** button between nodes. Click it to see the available node options (but do not add any nodes):
    - **Set a variable value**: Modify variable values using text, other variables, or formulas
    - **Ask a question**: Collect user input and automatically store it in a variable
    - **Add a condition**: Branch conversation logic based on variable values

> 💡
> **Tip:** Understanding these node types helps you see how variables flow through a topic — from collection (question nodes) to transformation (set variable nodes) to output (message nodes) to logic (condition nodes).

Observe Variables During Testing

15. In the test panel, start a new conversation.
16. Trigger the mailing list topic by saying:

```
I want to join the mailing list.
```

17. Follow the prompts and provide information when asked (email, name, etc.).
18. While in the test conversation, click **Variables** in the test panel (or look for a Variables icon).
19. Review the current values of all variables in the conversation. Notice how variables populate in real-time as you progress through the conversation.

> 💡
> **Tip:** Monitoring variables during testing is essential for debugging complex conversation flows and understanding data flow. This is one of the most useful debugging tools in Copilot Studio.

🏅 Congratulations! You've completed Use Case 1!

Test your understanding

Key takeaways:

- **Topic Variables Scope Locally** – Topic-level variables exist only within a single topic, preventing namespace pollution and keeping data organized
- **Global Variables Share Context** – Global variables persist across all topics and the entire conversation, enabling cross-topic data sharing
- **Variables Are Auto-Created** – Copilot Studio automatically creates variables when you add question nodes, so variables are already working in your topics
- **The Test Panel Shows Live Values** – Monitoring variables during testing is one of the most powerful debugging tools in Copilot Studio

Lessons learned & troubleshooting tips:

- If a variable isn't available in a message or node, check its scope - topic variables can't be accessed from other topics
- Always give variables descriptive names - `emailAddress` is better than `var1`
- Use the Variables panel during testing to verify data is stored correctly
- Consider variable lifetime - topic variables reset when the topic ends, global variables persist until the conversation ends

Challenge: Apply this to your own use case

- What user information would your agent need to remember across a conversation?
- Which variables should be global vs. topic-level in your agent?
- How could you use the Variables test panel to debug a broken conversation flow?

🔄 Use Case #2: Create and Orchestrate Child Agents

Build specialized child agents with focused knowledge and instructions to create modular, scalable agent architectures.

Use case	Value added	Estimated effort
Create and Orchestrate Child Agents	Build specialized sub-agents with focused expertise to create modular, scalable solutions	13 minutes

Summary of tasks

In this section, you'll learn how to create child agents, configure their specialized knowledge and instructions, set up orchestration rules in the parent agent, and test multi-agent interactions.

Scenario: Your Copilot Studio Assistant needs specialized expertise in prompt engineering frameworks. Instead of overloading the main agent with all knowledge, you'll create a child agent called "CARE Prompt Guidance" that specializes in the CARE framework for prompt writing. This child agent will be automatically invoked when users ask about general prompt guidance.

Objective

Create a specialized child agent and configure the parent agent to orchestrate conversations appropriately.

Step-by-step instructions

Create a Child Agent

1. In your Copilot Studio agent, locate the **Agents** or **Skills** section in the left navigation panel.
2. Click **+ Add agent** or **Create child agent** (terminology may vary).
3. Select **Child agent** as the agent type.
4. Configure the child agent with the following details:

Name:

```
CARE Prompt Guidance
```

Description:

```
This agent provides information on the CARE Prompt guidance.
```

ℹ️
Note: The description helps the parent agent understand when to route conversations to this child agent. Be specific and clear.

5. Click **Create** to initialize the child agent.

Configure Child Agent Instructions

6. Once the child agent is created, navigate to its **Instructions** section.
7. Add the following instructions:

```
This agent should help users with understanding information about the prompt guidance framework and how they can leverage it to make their agents better.
```

💡
Tip: Child agent instructions should be focused and specific to their domain of expertise. Avoid generic instructions - be precise about what this agent knows and does.

8. Click **Save** to apply the instructions.

Add Knowledge Sources to Child Agent

9. In the child agent, navigate to **Knowledge** in the left panel.
10. Click **+ Add knowledge** to add a knowledge source.
11. Select **Upload files** as the knowledge source type.
12. Download the CAREful Prompts Printable Guide (PDF) (https://media.nngroup.com/media/articles/attachments/CAREful_Prompts_-_Printable-2.pdf?_gl=1*1lto3e8*_up*MQ..*_ga*NDc3ODQ3MjYzLjE3NzA2Njc5OTM.*_ga_630S9ESVKM*czE3NzA2Njc5OTIkbzEkZzAkdDE3NzA2Njc5OTIkajYwJGwwJGgw) and upload it to the child agent as a knowledge source.

⚠️
IMPORTANT: Child agents can have their own dedicated knowledge sources. This keeps knowledge organized and prevents one agent from being overloaded with unrelated content.

13. Wait for the knowledge source to be indexed.
14. Click **Save** to finalize the child agent configuration.

Configure Parent Agent Orchestration

15. Return to your **parent agent** (the main Copilot Studio Assistant).
16. Navigate to the parent agent's **Overview** page. This is where you configure the agent's main instructions — not the child agent's instructions.

17. In the parent agent's **Instructions** field on the Overview page, add the following orchestration instructions. Notice the `(replace this text)` placeholder — you'll replace it with a direct reference to the child agent in the next step.

```
Use (replace this text) when asked to provide just general guidance around prompt building. Never use it when asked to analyze a prompt.
```

18. In the parent agent's Instructions field, select the `(replace this text)` placeholder, then type `/` to open the dropdown menu. This lists your agent's available tools, topics, child agents, knowledge sources, and more. Select **CARE Prompt Guidance** from the list to create a direct reference to the child agent, replacing the placeholder text.

> 💡
> **Tip:** Using `/` references in your agent instructions creates explicit links to specific items in your agent configuration. This ensures the agent knows exactly which tool, topic, or child agent you're referring to — rather than relying on plain text name matching.

> ⚠️
> **IMPORTANT:** Orchestration instructions are critical for proper agent routing. Be explicit about WHEN to use each child agent and WHEN NOT to use them. This prevents confusion and ensures the right agent handles each request.

19. Click **Save** to apply the orchestration instructions.

Review Agent Relationships

20. In the parent agent, navigate to the **Agents** or **Skills** panel.
21. Verify that the "CARE Prompt Guidance" child agent appears in the list of available agents.
22. Check that the child agent is **Enabled** (toggle should be on).

> ℹ️
> **Note:** Disabled child agents won't be invoked by the parent agent. Always verify child agents are enabled after creation.

Test the Child Agent

23. In the parent agent's test panel, start a new conversation.
24. Ask a question that should trigger the child agent:

```
How does the CARE prompt guidance help write prompts?
```

25. Observe the agent's response. It should:
    - Recognize that this is a general prompt guidance question
    - Route the conversation to the "CARE Prompt Guidance" child agent
    - Provide an answer grounded in the CARE framework knowledge

26. Look for indicators in the test panel showing which agent responded (some interfaces show “Responded by: CARE Prompt Guidance” or similar).

Test Orchestration Logic

27. Now test the orchestration instructions by asking a question that should NOT use the child agent:

```
Analyze this prompt for improvements: Write a summary of the quarterly report.
```

28. Verify that the parent agent uses the Prompt Analyzer tool (from the previous lab) instead of routing to the child agent.

> 💡
> **Tip:** This demonstrates proper orchestration - the parent agent understands the difference between “general prompt guidance” (child agent) and “analyze a specific prompt” (tool).

Explore Child Agent Capabilities

29. Ask several different questions to test the child agent’s knowledge:
    - “What is the CARE framework?”
    - “How do I write better prompts?”
    - “What does the A stand for in CARE?”
30. Verify that the child agent consistently provides accurate answers from its knowledge source.
31. Return to the child agent’s configuration and review how you could:
    - Add more knowledge sources
    - Refine instructions for better responses
    - Create additional child agents for other domains

🏅 Congratulations! You’ve completed Use Case 2!

Test your understanding

- When should you create a child agent vs. adding more knowledge to the parent agent?
- How do orchestration instructions help the parent agent make routing decisions?
- What happens if orchestration instructions are ambiguous or missing?

Challenge: Apply this to your own use case

- What specialized domains in your organization would benefit from dedicated child agents?
- How would you organize knowledge across parent and child agents for optimal performance?
- What orchestration rules would you write to ensure proper routing in a multi-agent system?

🧱 Use Case #3: Deploy Your Agent Across Channels

Learn how to configure and deploy your agent to multiple channels, understand channel-specific settings, and implement appropriate security controls.

Use case	Value added	Estimated effort
Deploy Your Agent Across Channels	Make your agent accessible across web, Teams, and other platforms with appropriate security	12 minutes

Summary of tasks

In this section, you'll learn how to navigate the Channels interface, configure the web channel with security settings, deploy to Microsoft Teams, and understand channel capabilities and limitations.

Scenario: Your Copilot Studio Assistant is ready for users. You need to make it available on your company website for easy access and in Microsoft Teams where most employees spend their day. You'll configure both channels with appropriate security settings.

Objective

Deploy your agent to web and Teams channels with proper configuration and security.

Step-by-step instructions

Navigate to Channels

1. In your Copilot Studio agent, click **Channels** in the left navigation panel.
2. Review the Channels overview page to see available channel options:
    - **Microsoft Teams**: Native Teams integration
    - **Demo website**: Test website for quick agent testing
    - **Custom website**: Embeddable web widget for your sites
    - **Mobile app**: iOS and Android integration
    - **Custom channel**: Direct Line API for custom applications
    - Additional channels may include Facebook, Slack, etc.

> ℹ️
> **Note:** Available channels depend on your Copilot Studio license and environment settings. Some channels require additional configuration or premium licenses.

3. Notice which channels are already enabled or configured (typically the demo website is enabled by default).

Explore Channel Capabilities

4. Review the description and capabilities of each channel type:
    - **Teams**: Full authentication, rich adaptive cards, deep Microsoft 365 integration
    - **Web**: Customizable appearance, flexible security, easy embedding
    - **Mobile**: Native app experience with push notifications
    - **Custom**: Full API control for advanced integrations

> 💡
> **Tip:** Choose channels based on where your users already work. Don't force users to adopt new tools - bring the agent to their existing environment.

5. Consider the limitations of each channel:
    - Some features (like certain adaptive cards) may not work on all channels
    - Authentication requirements vary by channel
    - Customization options differ across channels

Configure the Demo Website

6. Click on **Demo website** to view its configuration.
7. Review the demo website settings:
    - **Website URL**: The temporary URL where you can test your agent
    - **Status**: Whether the demo site is enabled or disabled
8. If the demo website isn't enabled, click **Enable** or **Turn on** to activate it.
9. Click **Copy** next to the demo website URL, then open it in a new browser tab.
10. Test your agent on the demo website by asking a few questions to verify functionality.

> 💡
> **Tip:** The demo website is perfect for quick testing and sharing with stakeholders before full deployment. Use it to gather feedback before wider rollout.

Configure Custom Website Channel

11. Return to the Channels page and select **Custom website** (or **Website** depending on your interface).
12. Review the custom website configuration options:
    - **Embed code**: HTML/JavaScript code to embed the agent on your site
    - **Style customization**: Colors, fonts, and branding options
    - **Security settings**: Authentication and domain restrictions
13. Click on **Configure** or **Settings** to access detailed configuration options.

Configure Web Channel Security

14. In the custom website settings, locate the **Security** section.
15. Review the security options available:
    - **No authentication**: Anyone can access the agent (use for public websites)
    - **Require authentication**: Users must sign in (use for internal sites)
    - **Allowed domains**: Restrict embedding to specific domains

> ⚠️
> **IMPORTANT:** Security settings are critical for protecting sensitive data and ensuring compliance. For internal agents with access to company data, always require authentication.

16. Configure domain restrictions by adding your website domains to the allowed list:

```
contoso.com
www.contoso.com
intranet.contoso.com
```

17. Review how domain restrictions prevent unauthorized embedding of your agent on external sites.

> ⚠️
> **WARNING:** Without domain restrictions, anyone can embed your agent on any website. Always configure allowed domains for production deployments.

18. Click **Save** to apply the security settings.

Deploy to Microsoft Teams

19. Return to the Channels page and select **Microsoft Teams**.
20. Review the Teams channel configuration options:
    - **App name**: How the agent appears in Teams
    - **App icon**: Visual branding in Teams
    - **Availability**: Who can access the agent
21. Click **Turn on Teams** or **Enable** to activate the Teams channel.
22. Review the deployment options:
    - **Share with team**: Add the agent to a specific Teams team or channel
    - **Install for myself**: Add the agent to your personal Teams chat
    - **Submit for admin approval**: Request organization-wide deployment
23. Click **Download manifest** or **Get agent link** to get the Teams app package.

> ℹ️
> **Note:** For organization-wide deployment, your IT admin must approve and publish the agent to your company's Teams app catalog.

24. Follow the prompts to install the agent in your personal Teams or share it with a team for testing.
25. Open Microsoft Teams and verify that the agent appears in your chat list or team channels.

Test Multi-Channel Deployment

26. Test your agent in Microsoft Teams by sending a few messages.
27. Compare the experience between Teams and the web demo site:
    - Notice how the UI differs
    - Test the same questions on both channels
    - Observe how authentication works on each channel

> 💡
> **Tip:** Always test your agent on each deployed channel. Some features or formatting may work differently across channels.

Review Channel Limitations

28. Return to the Channels page in Copilot Studio.
29. Review the **Channel capabilities** documentation or table showing:
    - Which features work on which channels
    - Known limitations per channel
    - Recommended practices for multi-channel deployment
30. Consider how channel limitations might affect your agent design:
    - Avoid features that don't work on your primary channels
    - Design for the lowest common denominator if deploying everywhere
    - Create channel-specific topics for advanced features

🏅 Congratulations! You've completed Use Case 3!

Test your understanding

Key takeaways:

- **Channels Enable Access** – Deploy to channels where your users already work to maximize adoption and minimize friction

- **Security Settings Matter** – Always configure appropriate authentication and domain restrictions to protect data and ensure compliance
- **Channel Capabilities Vary** – Test thoroughly on each channel and design agents that work within the limitations of your target platforms
- **Demo Sites Accelerate Feedback** – Use demo websites for quick testing and stakeholder review before full production deployment

Lessons learned & troubleshooting tips:

- If your agent doesn't appear in Teams after deployment, check with your IT admin about app approval policies
- Domain restrictions prevent unauthorized embedding - always configure allowed domains for production
- Test authentication flows on each channel to ensure proper security enforcement
- Some rich UI features may not work on all channels - design for compatibility

Challenge: Apply this to your own use case

- Which channels would provide the most value for your users?
- What security settings are appropriate for your agent's data sensitivity?
- How would you roll out your agent - pilot with a team first or organization-wide immediately?

🏆 Summary of learnings

True learning comes from doing, questioning, and reflecting—so let's put your skills to the test.

To maximize the impact of variables, child agents, and channels in Copilot Studio:

- **Variables Provide Memory** – Use topic-level variables for localized data and global variables for shared context. This creates agents that remember user information and maintain conversation continuity.
- **Scope Variables Appropriately** – Don't make everything global. Topic variables keep data organized and prevent namespace pollution while global variables enable cross-topic coordination.
- **Child Agents Enable Specialization** – Build focused child agents with dedicated knowledge and instructions rather than overloading a single agent with all content.
- **Orchestration Instructions Matter** – Clear, explicit orchestration rules in the parent agent ensure proper routing and prevent confusion between child agents.
- **Deploy Where Users Work** – Meet users in their existing environments (Teams, web, mobile) rather than forcing them to adopt new tools. This maximizes adoption and minimizes friction.
- **Security Comes First** – Always configure appropriate authentication and domain restrictions based on data sensitivity. Test security settings thoroughly before production deployment.

Conclusions and recommendations

Variables, child agents, and channels golden rules:

- Use descriptive variable names that clearly indicate their purpose and content
- Choose the appropriate variable scope - default to topic-level unless you need global access
- Create child agents when knowledge domains are distinct and specialized
- Write explicit orchestration instructions that clarify when to use each child agent

- Deploy to channels where users already spend their time - don’t ask users to go somewhere new
- Always configure security settings appropriate for your data sensitivity level
- Test multi-agent interactions and multi-channel deployment thoroughly before production rollout

By following these principles, you’ll build sophisticated agent solutions that scale with your organization’s complexity - combining conversation memory, specialized intelligence, and strategic deployment to deliver accurate, contextual, and valuable user experiences wherever your users need them.
