Microsoft

Deploying Microsoft Copilot Studio Agents Externally to Serve Customers & Suppliers

Patterns for Public Channel Integration

Microsoft worked with Shane Young, Power Platform MVP, to build out Copilot Studio training for Power Platform makers as part of a paid collaboration. This paper is intended to provide considerations for builders when building; it is not intended as a step-by-step.

Table of contents

Microsoft Copilot Studio enables the creation of sophisticated conversational AI agents. While often used internally, significant value can be unlocked by deploying these agents to external channels such as public websites, social media platforms, and custom applications.

This paper outlines common patterns, technical considerations, and best practices for extending Copilot Studio agents beyond organizational boundaries to engage customers and external users effectively.

We will explore three primary deployment patterns: embedding agents in public websites, integrating with Facebook Messenger, and incorporating agents into custom applications using the Direct Line API. Although we're using Facebook as an example, additional social channels are available through ISVs for Azure channels.

1. Introduction: Extending Copilot Studio Agents to External Audiences

You have utilized Microsoft Copilot Studio to build an effective custom agent – your organization's specialized agent. It's designed to be intelligent, helpful, and capable of addressing specific user needs. However, the potential reach of this agent often extends beyond internal Microsoft 365 users. Consider scenarios where customers visiting your public website, users interacting with your brand on Facebook, or individuals using your custom mobile application could benefit from this AI-driven assistance.

Microsoft Copilot Studio is engineered to support these external-facing deployments. You can configure your agents to operate on various public channels, effectively providing an accessible interface to your AI capabilities for the outside world.

Key motivations for external deployment include:

- **Enhanced User Accessibility:** Engage users within their preferred environments – your public website, social media channels like Facebook Messenger, or dedicated custom applications – rather than requiring them to adopt internal tools.
- **Scalable Support Operations:** Provide immediate, automated assistance and answers on public channels 24/7, reducing the load on human support teams for common inquiries.
- **Seamless Brand Integration:** Present the AI agent as a natural, integrated component of your existing external digital presence, maintaining alignment with and tone of your brand's voice.
- **Expanded Application Scenarios:** Enable use cases such as public-facing FAQ bots, automated order status inquiries, social media customer service agents, in-app virtual assistants, and more.

This paper details three common patterns for deploying Copilot Studio agents externally: website embedding, Facebook Messenger integration, and custom application integration. We will cover the necessary configuration steps, essential technical aspects like authentication and security, and guidance on optimizing the user experience for each channel.

2. Deployment Patterns for External Integration

Let's examine the practical steps and considerations for deploying your Copilot Studio agent across different external channels.

2.1. Deployment Pattern 1: Website Integration via Embedded Chat

Use Case: Integrating an AI support agent directly onto a public-facing website (e.g., corporate site, e-commerce platform, support portal) to provide instant answers to common questions, assist with navigation, or handle simple tasks like order tracking or advanced tasks like help filing a credit application or purchasing a product. This offers immediate customer support within the browsing experience. Contextual examples can often be found in community discussions and official blogs, such as posts detailing Copilot Studio capabilities.

Deployment Steps:

1. **Prepare and Publish the Agent:** Ensure your agent is thoroughly tested and published within Copilot Studio. Publishing activates the agent and makes it available for channel connections. Refer to Microsoft Learn – "Key concepts – Publish and deploy your agent" for foundational steps.
2. **Configure Authentication:**
    - For most public website scenarios requiring broad access, select **No authentication** in the agent's security settings. This allows any visitor to interact without needing to sign in, maximizing accessibility. However, **be aware of the security risks of this approach! Selecting the No authentication option allows anyone who has the link to chat and interact with your bot or agent.** It is recommended you apply authentication, especially if you are using your bot or agent within your organization or for specific users, along with other security and governance controls. Additionally, agent Actions that require authentication won't be available if you choose this option.
    - If specific user context is required later in the conversation (e.g., retrieving order details), the **Authenticate manually** option can be implemented within specific topics, but **No authentication** provides the simplest initial setup for public access (Refer to Microsoft Learn – "Key concepts – Publish and deploy your agent" for more details on authentication for web channels).
3. **Obtain the Embed Code:** Navigate to the **Channels** section in Copilot Studio, select **Custom Website** (or similar naming), and copy the provided <iframe> code snippet. This snippet contains the necessary configuration to render the chat widget. Instructions are available at Microsoft Learn – "Publish an agent to a live or demo website".
4. **Integrate into Website HTML:** Provide the copied <iframe> snippet to your web development team or embed it directly into the HTML source of the desired web pages (often placed in a shared footer element for site-wide availability). Publish the website changes to make the chat widget visible (live website publishing guide).
5. **Customize Appearance (Optional):** Enhance the user experience by customizing the welcome message, incorporating branding elements like a logo, or suggesting initial conversation starters directly within the Copilot Studio channel settings. See customization options under custom welcome messages. You can also customize the chatpane using PowerCAT Tools. (See Power CAT Tools for general information.)
6. **Validate the Deployment:** Access the live website, locate the chat icon, and initiate a conversation to verify the agent responds correctly. Updates made and republished within Copilot Studio typically reflect automatically on the website without needing to modify the embed code.
7. **Authentication & Security Considerations:**
    - **Open Access:** The **No authentication** setting facilitates easy public access but means the agent operates without specific user identity context, limiting its ability to perform personalized, secure actions (limitations without auth).
    - **Secret Management (If Applicable):** If using more advanced integration methods that might involve secrets (like the Direct Line pattern discussed later), ensure these secrets are *never* exposed in client-side code (website HTML/JavaScript). Secrets must be managed securely on the server-side. Refer to best practices in Microsoft Learn – "Configure web and Direct Line channel security".
    - **Content Moderation:** Consider implementing content filtering or moderation strategies if user input might contain sensitive information. Copilot Studio offers features, and organizations should

align with governance policies (this is more dependent on your agent's initial setup, and details can be found on this Microsoft Blog post on Copilot governance and controls).

8. **User Interaction Design:**
    - **Visibility:** Ensure the chat widget is easily discoverable using a clear, persistent icon or button, often placed in a standard location like the bottom-right corner.
    - **Greeting:** Provide a clear and welcoming initial message, potentially suggesting common queries or capabilities.
    - **Conciseness:** Design agent responses to be clear and concise, respecting the user's time and attention span in a web context. (Again – this is defined in your agent setup and is not necessarily specific to externally published agents.)
    - **Accessibility:** Adhere to accessibility standards (e.g., WCAG) to ensure the chat interface is usable by everyone, including those relying on assistive technologies (Microsoft Accessibility Guidelines).

Note: The <iframe> embed method is generally the most straightforward approach for website integration, as it leverages the pre-built chat interface provided by Microsoft.

2.2. Deployment Pattern 2: Facebook Messenger Integration

Use Case: Deploy a Copilot Studio agent to a Facebook Business Page to handle customer inquiries, provide information, or offer support directly within the Facebook Messenger platform. This allows engagement with users on a widely used social media channel.

Architecture: Copilot Studio provides a dedicated connector for Facebook Messenger, simplifying the integration. Communication flows from the user in Messenger through Facebook's platform to Copilot Studio and back. No intermediary Azure Bot Service relay is typically required for this specific channel. Be aware that conversation data transits through Facebook's infrastructure (Microsoft Learn – Publish an agent to Facebook).

Prerequisites:

- An active Facebook Business Page.
- A Facebook Developer Account (developer.facebook.com) with administrative access to the Page.
- A Copilot Studio agent, created and published.

Deployment Steps:

1. **Configure a Facebook Application:**
    - Navigate to Facebook for Developers and create a new application or select an existing one.
    - Record the **App ID** and **App Secret** from the app's Basic Settings. Secure the App Secret (Retrieve Facebook app information).
    - Enable "**Allow API Access to App Settings**" under Advanced Settings (Enable API access).
    - Add the **Messenger** product to your application (Add Messenger to your app).
    - In Messenger Settings > Access Tokens, link your Facebook Business Page using "**Add or Remove Pages**". Grant the necessary permission, typically "**pages_messaging**" (manage and access Page conversations) (Configure Facebook pages).
    - Generate a **Page Access Token** for the linked Page. Securely store the **Page ID** and the generated **Page Access Token** (Get tokens).

2. **Connect Copilot Studio to the Facebook App (Configure the Facebook channel):**
    - Within your Copilot Studio agent, navigate to **Channels** and select the **Facebook tile**.
    - Enter the collected **App ID, App Secret, Page ID, and Page Access Token** into the configuration panel.
    - Upon successful connection, Copilot Studio will provide a **Callback URL** and a **Verify Token**. Record these values.
3. **Configure Facebook Webhooks (Connect your Facebook app to Copilot Studio):**
    - Return to your Facebook app's Messenger Settings > Webhooks section. Click "**Add Callback URL**"
    - Paste the **Callback URL** and **Verify Token** obtained from Copilot Studio and verify the connection
    - Subscribe to the webhook to relevant Page events. Click "**Add Subscriptions**" for your Page. Select at minimum **messages** and **messaging_postbacks**. Consider others like messaging_options if needed.
    - **Initial Testing**: While the Facebook app is in Development mode, only designated admins, developers, or testers can interact. Send a test message to your Page via Messenger to confirm connectivity.
4. **Submit for Facebook App Review and Go Live:**
    - **Policy Compliance**: Ensure you have accessible URLs for your Privacy Policy and Terms of Service. Add these to your Facebook app's Basic Settings. Adherence to Facebook Platform Policies is mandatory (Facebook for Developers – Messenger Platform Policies & App Review Guidelines).
    - **App Review Submission**: Submit your application for review via the Facebook Developer portal (App Review section), specifically requesting the necessary permissions (e.g., pages_messaging) (Submit for Facebook Review). Be prepared to describe the agent's functionality.
    - **Approval Process**: Address any feedback provided by Facebook reviewers. Compliance with policies, such as messaging windows, is critical (Details on Facebook app review).
    - **Activate Public Access**: Once approved, switch your Facebook app's status to Live in the developer dashboard (Facebook - moving from development to live). Ensure your Facebook Page is also published. The agent should now be accessible to the public via Messenger.
    - **Monitoring**: Utilize Copilot Studio analytics and Facebook Page Insights to monitor interaction patterns and performance.

Authentication & Security Considerations:

- **Public Access Requirement:** Set Copilot Studio authentication to **"No authentication"**. Users interact via their Facebook identity. If previously configured for Microsoft Entra ID, this setting *must* be changed in Settings > Security > Authentication, followed by republishing the agent.
- **Manual Authentication (Advanced):** For scenarios requiring user verification or access to protected resources, Copilot Studio topics can implement manual OAuth flows, potentially leveraging Facebook Login (Authenticating manually with OAuth flows). This adds significant complexity and is not recommended unless you are already highly proficient in the setup process.
- **Credential Protection:** Safeguard the **Facebook App Secret** and **Page Access Token**. These credentials should never be exposed publicly. Utilize an existing KMS, such as Azure Key Vault, if you have one available.
- **Data Transit:** Acknowledge that conversation data passes through Facebook's systems. Ensure compliance with your organization's data privacy policies.
- **Platform Policies:** Strictly adhere to all applicable Facebook Platform Policies (Facebook for Developers – Messenger Platform Policies & App Review Guidelines).

User Interaction Design (Messenger Specifics):

- **Text Formatting:** Messenger primarily supports plain text. Avoid reliance on Markdown for emphasis. Use simple language, short paragraphs, and emojis judiciously (Facebook Markdown support).
- **Message Length:** Break down longer responses into multiple, smaller messages for better readability on mobile devices.
- **Quick Replies:** Leverage Quick Replies within Copilot Studio topics to present users with tappable response options (Microsoft Learn – Copilot Studio Quick Replies and Adaptive Cards). These render as buttons in Messenger but have limitations (e.g., number of replies, disappear after use) (Microsoft Learn – Facebook channel integration tips).
- **Adaptive Cards:** Be aware that Adaptive Card support in Messenger is limited. Cards may render partially (e.g., showing only an image) or lack interactivity (Microsoft Learn – Copilot Studio Quick Replies and Adaptive Cards - *Note: Link discusses general card support*). Always provide essential information and actions in text format; do not rely solely on cards.
- **Images:** Use mobile-optimized images and provide descriptive text context.
- **Responsiveness:** Ensure timely responses. Typing indicators are usually handled automatically. Consider providing status messages ("Please wait while I check...") for potentially long-running operations.
- **Native Messenger Features:** Complement the agent by configuring Facebook Page features like a **Persistent Menu** or **Ice Breakers**.
- **Escalation Path:** Design a clear mechanism for users to request human assistance (e.g., recognizing phrases like "talk to an agent"). This is part of your Escalation topic setup within the agent itself.

Recommendation: Thoroughly test the agent's conversational flows directly within the Facebook Messenger mobile application to accurately assess the end-user experience, as it differs significantly from the Copilot Studio test canvas.

2.3. Deployment Pattern 3: Custom Application Integration via Direct Line API

Use Case: Embedding a Copilot Studio agent directly within the user interface of a custom-built mobile application (iOS, Android) or web application (e.g., customer portal, internal line-of-business app). This pattern provides the most seamless and deeply integrated user experience (Microsoft Learn – "Publish an agent to mobile or custom apps").

Deployment Steps: This pattern requires custom development effort, primarily focused on building the chat UI and managing communication via the Direct Line API.

1. **Select Integration Approach:**
    - **Embedded Web Chat:** If the custom application can host web content (e.g., using a WebView component in mobile), the <iframe> embed snippet (similar to Pattern 1, but using the snippet from the "Mobile or custom app" channel) can be used. This leverages the Microsoft-provided chat UI.
    - **Direct Line API (Focus of this Pattern):** For complete control over the UI and user experience, integrate using the Bot Framework Direct Line API. This involves building a custom chat client within your application (same link above).
2. **Publish Agent and Obtain Token Endpoint:** Publish the agent in Copilot Studio. Navigate to **Channels** > **Mobile or custom app**. Copy the **Token Endpoint** URL. This URL is used by your *backend service* to request temporary access tokens for the agent (Get your agent's token parameters).
3. **Implement Secure Token Generation (Server-Side):** This is a critical security step. The Direct Line secret associated with your agent must *never* be exposed to the client application (mobile app or web frontend).
    - The client application requests a chat token from *your* secure backend API.

- Your backend service uses the **Token Endpoint** URL and the securely stored **Direct Line secret** (e.g., retrieved from Azure Key Vault) to make a server-to-server request to the Microsoft Bot Framework token service.
- The token service returns a short-lived **Direct Line token** and a **conversationId**.
- Your backend service returns *only* the temporary **token** and **conversationId** to the client application. Adhere strictly to the token exchange pattern outlined in Azure bot service security guidelines.

4. **Establish Connection from Client Application:** The client application now uses the obtained token and conversation ID to connect.
    - **Native Mobile App:** Utilize appropriate libraries (e.g., Bot Framework Direct Line SDKs if available/suitable for the platform, or implement direct HTTPS calls) to connect to the Direct Line service endpoint using the token. Send user messages (activities) via HTTP POST and receive agent responses via a persistent connection (e.g., WebSockets) or polling. Consult the Bot Framework documentation, "Direct Line API".
    - **Custom Web App:** The recommended approach is to use the official BotFramework-WebChat JavaScript library. Initialize this component with the fetched token and conversation ID. It provides a customizable chat UI and handles the complexities of sending/receiving messages over Direct Line (Microsoft Bot Framework Web Chat documentation).
5. **Develop the Custom Chat User Interface:** Design and build the chat interface elements (message bubbles, input area, etc.) to match the look and feel of your custom application. Ensure your UI can render various activity types sent by the agent, such as text, images, buttons (suggested actions), and potentially Adaptive Cards. Refer to Bot Framework docs, "Adaptive Cards in custom channels" for card rendering guidance.
6. **Manage Conversation State and Token Refresh:** Maintain the conversationId for the duration of the chat session. Since Direct Line tokens expire, the client application must implement logic to request a new token from your backend service (using the same conversationId or initiating a new one) before the current token expires or when reconnection is needed. Define the desired session persistence behavior (e.g., retain history across app launches vs. ephemeral sessions). See details on session lifecycle.
7. **Perform End-to-End Testing:** Validate the entire flow: client requests token -> backend fetches token -> client receives token -> client connects and exchanges messages with the agent. Test edge cases like token expiration, network interruptions, and error handling.

Authentication & Security Considerations:

- **Direct Line Secret Protection: Crucially important:** The agent's Direct Line secret must reside *only* on your secure backend infrastructure. Never embed it in client-side code (mobile app binaries, web app JavaScript). The token exchange pattern is mandatory for secure implementation.
- **User Identity Propagation:** Your custom application likely has its own user authentication system. You can optionally pass the authenticated user's ID (appropriately anonymized or pseudonymized if necessary) to the agent when initiating the Direct Line conversation (e.g., within the initial token request or as part of the first activity). This allows the agent to potentially personalize responses or access user-specific data via backend integrations triggered by the agent. However, the agent itself should not be relied upon for primary application authentication. If agent-level authentication actions are needed, explore the manual authentication approach.

User Interaction Design:

- **Seamless Integration:** Strive for a chat UI that perfectly aligns with the host application's design system (colors, fonts, controls).
- **Contextual Introduction:** Clearly introduce the agent and its capabilities within the app context.

- **Mobile Optimization:** Prioritize brevity and clarity in messages for smaller mobile screens. Use UI elements like multiple message bubbles effectively.
- **Privacy Considerations:** Be transparent about data handling, especially regarding chat history persistence. For applications handling sensitive data, consider ephemeral chat sessions.
- **Escalation Mechanism:** Provide a clear and accessible option for users to escalate to human support or find alternative help channels if the agent cannot resolve their issue. Explore patterns for bridging AI and human assistance.

Note: The Direct Line API pattern requires the most significant development investment but offers maximum flexibility and control, enabling a truly native and integrated agent experience within your custom application. Secure handling of the Direct Line secret via the token exchange pattern is non-negotiable.

3. Implementing Agent Handoff from Copilot Studio

3.1 Introduction to Handoff Scenarios

Microsoft Copilot Studio allows organizations to seamlessly hand off a conversation from an AI agent to a human agent on various customer engagement platforms. This can be especially useful when publishing your agent to an external channel. Two common scenarios are:

1. Transferring the chat to a generic engagement hub (for example, LivePerson, Genesys, Zendesk) via a Bot Framework–compatible interface.
2. Integrating with ServiceNow to either open a live agent chat or create a ServiceNow incident with context from the Copilot conversation.

In both cases, careful configuration is needed to trigger the handoff, maintain context and user experience, and uphold privacy and fallback considerations.

3.2 Pattern: Handoff to Generic Engagement Hubs (e.g., LivePerson, Genesys)

Triggering and Configuring Handoff in Copilot Studio:

- In Copilot Studio, you configure escalation through the Escalate system topic and **"Transfer to agent"** nodes within your conversation flows. Handoff can be triggered implicitly (for example, if the bot can't determine user intent or the user explicitly types "I need a human") or explicitly (when a specific topic decides that a human is needed).
- By default, Copilot Studio includes an Escalate topic that catches phrases like "talk to an agent." You can edit this topic to include a **Transfer conversation** node that initiates the handoff to a live agent. See Microsoft's documentation on "Escalate topic" and "Transfer to agent" nodes in Copilot Studio.
- In explicit handoff scenarios, you add the **Transfer** node where human help is required and optionally include a private note for the agent summarizing the user's situation. When the Transfer node is reached, Copilot Studio packages the conversation context and raises a special "handoff" event so the external system knows to bring in a human agent (related concepts often leverage the Direct Line channel: Generic handoff configuration).

User Experience Considerations:

- The user should remain in a single continuous conversation (web chat widget, mobile chat, or social channel) without switching apps.
- When the Transfer node fires, the bot can display a message like "Connecting you to a live agent…," and then the human agent's responses appear in the same interface.
- In a custom web chat, you may need to handle the handoff.initiate event in your client code to switch the chat interface from "bot mode" to "live agent mode."
- If you are using a third-party chat widget (e.g., Genesys web chat), ensure it supports switching from bot to human agent seamlessly. (Consult specific vendor documentation, e.g., Genesys documentation at Genesys widget or similar.)

Conversation Context Transfer:

- Copilot Studio automatically packages the conversation transcript and context variables (like the last triggered topic or the user's name) in the handoff.initiate payload.
- You should ensure your adapter or engagement hub uses these details to give the human agent immediate context (for example, display the last few messages, a summary, or "notes" from the bot). This prevents the user from repeating themselves.
- If your hub has a standard agent console, map the transcript to that console's message log or "contact history" field.

Privacy and Fallback Best Practices:

- **User Consent:** Let the user know you're transferring them to a human and that their chat will be shared.
- **Minimal Data:** Redact sensitive info if not necessary.
- **Fallback Plan:** If no agent is available, your adapter should send a message back to the user indicating off-hours or queue length. You could then create a ticket automatically or provide an alternative contact method.
- **Routing:** If your hub supports skill-based routing, you can route the user to a specialized agent group based on the bot's "topic" or conversation tags.

3.3 Pattern: Integration with ServiceNow

With ServiceNow being so prevalent for incident management, integrating with ServiceNow during a handoff is a common use case. Microsoft Copilot Studio can easily integrate with ServiceNow, enabling direct live chat escalation or automatic ticket creation. If your organization relies on ServiceNow for IT or customer support, you can unify Copilot's AI-driven interactions with your ServiceNow workflows. Full details on how to implement handoffs can be found here (ServiceNow handoffs), but we'll touch on some of the high level concepts below.

3.3.1 ServiceNow Live Agent Handoff via Virtual Agent

Handoff Trigger and Configuration:

- Similar to generic hubs, you add a **Transfer** node or rely on the Escalate topic in Copilot Studio.
- The difference is that ServiceNow usually acts as a "primary" Virtual Agent environment, with Copilot configured as an external bot via the **Bot Interconnect** feature (available in recent ServiceNow releases). You'll set up:
    1. Azure Bot / Direct Line for your Copilot agent.
    2. ServiceNow Virtual Agent with "Bot Interconnect" turned on.

3. A bridge service (often an Azure Function) that relays messages from ServiceNow to Copilot (and back).
4. A handoff event that tells ServiceNow to switch the conversation from the external bot to a human agent in the ServiceNow Agent Workspace. (Refer to ServiceNow developer resources ServiceNow developer resources and Direct Line concepts.)

Technical Requirements:

- ServiceNow Admin Access to enable Virtual Agent, Agent Chat, and Bot Interconnect.
- Copilot Studio Agent published with Direct Line enabled.
- Middleware to handle messages, typically coded in C# or Node.js, deployed on Azure.
- ServiceNow Scripts or "Custom Chat Integration" to handle user utterances, pass them to the Copilot, and detect a handoff.initiate event.
- When handoff.initiate is received from the Copilot side, ServiceNow's Virtual Agent flow instructs Agent Chat to take over, ensuring the user sees a smooth transition from bot to human in the same ServiceNow chat window.

User Experience:

- Within the ServiceNow portal or chat widget, the user interacts with an AI that's secretly powered by Microsoft Copilot Studio.
- When escalation occurs, the Virtual Agent posts a "Connecting you to a support agent..." message, then the human agent logs in to ServiceNow's Agent Workspace and takes over.
- The user doesn't have to switch UIs or re-authenticate. The entire chat transcript is typically visible in the Agent Workspace, giving the agent context on what the AI and user already discussed (See ServiceNow Virtual Agent documentation).

Context & Privacy:

- The conversation transcript and any relevant variables can be shared with the agent. You can configure the Copilot Transfer node to append a short note summarizing the issue.
- Ensure compliance with your data retention policies—ServiceNow logs the chat by default, so if personal data is present, you might mask or filter it as needed.

3.3.2 Incident Creation (Fallback or Supplement)

If a live agent is unavailable, or if your workflow prefers tickets for certain issues:

1. **Add a Ticket-Creation Action in Copilot Studio:** Use Power Automate or a custom action calling the ServiceNow API (for example, POST /api/now/table/incident). Leverage the official connector for ease of use (ServiceNow connector).
2. **Map Conversation Data:** Set "Short description," "Description," or custom fields from the user's answers.
3. **Confirm with the User:** "An incident has been created (INC12345). Our support will reach out soon!"
4. **Optionally log the transcript** in the "Work notes" or "Additional comments" field. You can decide how much detail to include, balancing context vs. privacy.

3.3.3 Best Practices for ServiceNow Integration

- **Set User Expectations:** Let them know your typical response time.
- **Enable Status Checks:** The Copilot agent can query the ServiceNow incident status if the user follows up (e.g., "What's the status of INC12345?").
- **Combine Approaches:** Some organizations attempt a live chat first, and if no agent accepts, automatically create an incident. The user is informed so they're not left waiting.

3.4 Summary of Handoff Patterns

- **Generic Engagement Hub:** A flexible approach for organizations using platforms like LivePerson or Genesys. Requires a custom adapter to pass messages/events between Copilot Studio (via Direct Line) and the hub's agent console.
- **ServiceNow:** Ideal if you already rely on ServiceNow for ITSM, letting Copilot do the first wave of support and either escalate to a ServiceNow live agent or file an incident. Involves enabling Virtual Agent, Bot Interconnect, and potentially a bridging service on Azure to link Copilot to ServiceNow.
- **More options are available!** You have lots of flexibility with Copilot Studio to handoff your agents, especially with the generic engagement hub approach.

In both scenarios, the user is offered a seamless, unified conversation flow—AI first, then human if needed—ensuring customers get quick answers but can always talk to a real person when the complexity demands it. Properly implemented handoffs reduce user frustration, shorten support times, and improve overall satisfaction with your support operations.

4. Monitor and Continuously Improve

Once your agent is implemented and interacting your customers, it's important to monitor the agent's interactions and continuously improve your customers' experience. Copilot Studio provides a wealth of information to assist you in the **Analytics** pane. You should regularly review the agent's analytics to identify opportunities for improvement. Analytics include:

- **Overview:** including total sessions, engagement, and satisfaction score summaries. By default the Analytics show the last 7 days, but you can customize the analysis window's date range.
- **Effectiveness:** allows you to track the outcomes of customer engagements with the agent. This includes how many interactions were resolved, escalated, abandoned, or unengaged. You can use this to determine the agent's overall success at handling engagements.
- **Knowledge source use:** to see exactly which knowledge sources your users are interacting with and error rates. Copilot Studio will even recommend supplemental knowledge sources for common inquiries that are going unanswered.
- **Action use and success rate:** to examine which actions are being called most often by your agent and how often those actions are successful.
- **Satisfaction:** after each interaction, your agent requests Customer Satisfaction (CSAT) information. CSAT scores are exposed in the Analytics section, and you can use them to evaluate and improve your customers' experience with the agent.

5. Conclusion: Activating Your Agent for External Engagement

Developing a capable agent in Microsoft Copilot Studio is the foundational step. Activating its potential requires deploying it to the channels where external users naturally interact – whether that is your public website, prominent social media platforms like Facebook, or within your bespoke applications.

- The **Website Embed** pattern offers a straightforward approach for public-facing information delivery using the standard <iframe> method.
- **Facebook Messenger Integration** leverages a dedicated connector to bring conversational AI into the social media landscape.
- **Custom Application Integration** via the Direct Line API provides the highest degree of control and seamlessness, requiring careful implementation of security patterns and custom UI development.
- **Leverage Handoffs** where it makes sense, just like you do with your agents on internal channels.

Each pattern presents unique implementation steps and considerations, particularly concerning authentication strategies, security best practices (especially secret management), and channel-specific user experience design. By strategically choosing and implementing the appropriate deployment pattern(s), organizations can effectively extend the reach and impact of their Copilot Studio agents, delivering value directly to customers and external stakeholders.