# Amplify Use Case with Copilot Studio

NOVA MAX Demo use cases with Copilot Studio

## Introduction

**Microsoft Copilot Studio** is a low-code platform for building AI-powered agents that can autonomously handle tasks and conversing with users, making it ideal for creating IT support bots [[1]](#footnote-1). We focus on designing four top-priority AI agents using Copilot Studio: **HARDY, INFO, KEYNOVA, and SWIFTREQ**. These agents target high-impact, frequent IT scenarios:

* **HARDY (Hardware Diagnostic Agent):** Monitors device hardware health and autonomously detects, diagnoses, and heals issues.
* **INFO (Knowledge Base Agent):** Provides instant answers by retrieving and summarizing relevant knowledge articles based on user queries or ticket context.
* **KEYNOVA (Password Reset Agent):** Securely verifies user identity and performs password resets/account unlocks, slashing the load of password-related tickets. – <Potential to get voice enabled, and device integration>
* **SWIFTREQ (Request Fulfillment Agent):** Automates repetitive IT service requests (software installs, access requests, etc.) with speed and precision, end-to-end.

Building these agents in Copilot Studio will leverage **conversational AI plus automation** to improve IT support efficiency and user experience.

**Microsoft Copilot Studio Advantages:** Copilot Studio provides a guided environment to create these agents with minimal code. It supports **two agent types** – **conversational agents** (which interact via chat/voice with users) and **autonomous agents** (which run in the background triggered by events) [[2]](#footnote-2). We will use **conversational agents for INFO, KEYNOVA, and SWIFTREQ** (since they engage with user requests), and an **autonomous agent for HARDY** (monitoring devices and acting on triggers). Copilot Studio integrates deeply with the Microsoft ecosystem and beyond: we can connect to Microsoft 365 services, external ITSM like ServiceNow, and hundreds of other systems via **Power Platform connectors** 1. This means our agents can not only chat and reason with AI, but also **take real actions** – resetting a password in Azure AD, creating a ServiceNow ticket, running a diagnostic script, etc., all within one solution.

**Design Approach:** For each agent, we will:

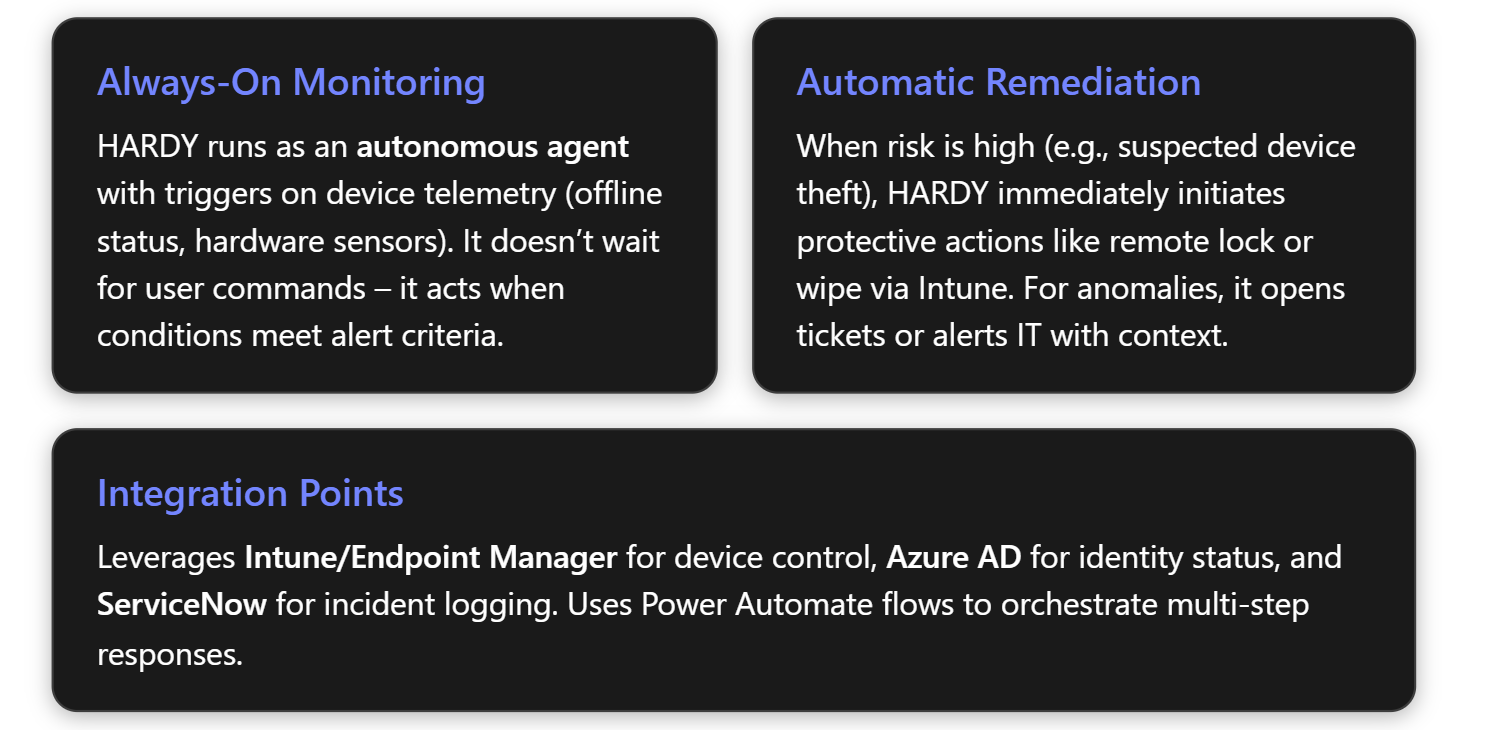
* **Define its core purpose and priority features** to implement first.
* **Map out triggers and conversation flows** (e.g. user says "reset my password", or device goes offline event).
* **Configure necessary data/knowledge sources** (e.g. knowledge base articles for INFO) and **actions/flows** (e.g. Intune wipe action for HARDY, Power Automate flows for SWIFTREQ).
* **Use Copilot Studio’s agent builder** to create the agent:
  + Set up the agent’s instructions (persona, scope, and how it should use tools/knowledge).
  + Add conversational topics and triggers, or autonomous triggers as appropriate.
  + Integrate the agent with **Copilot Studio Tools** (APIs, flows, etc.) for executing tasks.
* **Test the agent** thoroughly with sample scenarios and refine.
* **Deploy the NOVA MAX multi-agent** (e.g., in Teams for user-facing bots, or as a background service for autonomous tasks).
* **Monitor performance and iterate** – using built-in analytics and user feedback to improve accuracy and expand capabilities.

## HARDY – Hardware Diagnostic Agent (Autonomous)

**Purpose:** HARDY is an autonomous agent that keeps watch over endpoint devices (PCs, laptops, mobile devices) and **proactively handles hardware-related issues**. It uses telemetry and device management data to detect when a device might be missing, misused, or failing, and then takes appropriate action or alerts the right people. The goal is to minimize device downtime and prevent minor issues from becoming major incidents.

**Key Functions to Prioritize:**

* **Missing Device Detection:** Identify when a device goes “missing” (for example, if a device hasn’t been online for a specified period or shows an anomalous last known location). This could indicate loss or theft.
* **Hardware Anomaly Detection:** Monitor for patterns like frequent crashes, overheating, disk errors, battery degradation, etc. that suggest hardware problems. (We assume integration with telemetry sources like Intune endpoint analytics or a tool like SysTrack for these signals.)
* **Autonomous Remediation Actions:** Based on the risk level:
  + If a device is suspected lost/stolen, automatically **lock or wipe** it via Intune’s device management API for security [[3]](#footnote-3), and possibly disable it in Azure AD.
  + If hardware issues are detected (e.g., failing drive), create a ServiceNow **incident with diagnostic info** for IT to replace the part, and/or notify the user and IT team via Teams.
  + If minor anomaly (non-critical), perhaps just notify IT or log it for trend analysis (e.g., record it in a Dataverse table for devices).
* **Logging & Learning:** Keep a record of detected anomalies and actions taken. Over time, adjust thresholds (what is “offline beyond threshold,” etc.) by learning normal patterns (e.g., a device offline every weekend may be normal for certain users).



**Design & Implementation Steps in Copilot Studio:**

1. **Agent Type – Autonomous:** In Copilot Studio, create a **New Agent** and choose an **autonomous agent** (designed to run by triggers, not user chat) 4. Name it "HARDY – Hardware Diagnostic Agent" and give it a description (e.g., “Monitors device health and initiates automated remediation for hardware issues”). This sets HARDY to operate independently based on events we’ll define.
2. **Define Triggers:** Determine what events will trigger HARDY’s logic. We might not have out-of-the-box triggers for “device offline” in Copilot Studio, so we use Power Platform integration:
   * **Power Automate Trigger:** Create a flow that runs on a schedule or subscribes to device status changes. For example, a daily scheduled flow could query Intune (via Graph API) for devices that haven’t checked in for > X days, or devices reporting a compliance state “Not Compliant” with reason “no check-in”. For location anomalies, if Intune captures last known location or if Azure AD sign-in logs show unusual geography, that could be another data point (this might require a custom check).
   * **Custom Trigger Event:** Store results of that check in Dataverse or post them as a trigger. Copilot Studio can use a Dataverse table trigger or a custom Power Fx expression trigger. Alternatively, if using a Monitoring tool that can call a webhook, configure it to call a **Power Automate HTTP trigger** that then calls the Copilot agent.
   * In Copilot Studio’s agent builder, add a **Trigger** that connects to the output of the above flow. For example, the flow could invoke the agent with a payload: { "DeviceID": ..., "AlertType": "MissingDevice", "LastSeen": "2025-07-20" } when criteria are met.
   * Similarly, set up triggers for hardware anomalies: perhaps another flow that listens for alerts from a telemetry system (like a SysTrack alert for high temperature or SMART failure warning) and then calls the agent with details (e.g., "AlertType": "HardwareFailure", "Issue": "DiskSMART", "DeviceID": ...).
3. **Configure Tools/Actions:** HARDY will need to perform actions in response:
   * **Intune Device Actions:** Use the **Microsoft Graph Intune connector** (or a custom action) to remotely lock or wipe a device. In Copilot Studio, go to the **Actions/Tools** section and add a new Action. If a built-in connector exists (e.g., an Intune connector or we can call Graph via an HTTP request action), configure it. For example, the action “WipeDevice” may call the endpoint POST /deviceManagement/managedDevices/{id}/wipe. Similarly a “LockDevice” or “RetireDevice” action could be configured. Provide required parameters (DeviceID, etc.) which will come from trigger data.
   * **ServiceNow Incident Creation:** Add a tool for ServiceNow. If using the built-in IT Helpdesk template as reference, note it already can create tickets 5. In our custom scenario, define an action using the **ServiceNow connector** (via IntegrationHub or Power Automate flow) to create an incident. We’ll need to map fields like short description (“HARDY: Device flagged as missing/issue”), details (include last seen info or error context), assignment group (perhaps IT support team).
   * **Notification (Teams/Email):** Add an action to send a Teams message or email. For Teams, we can use the Microsoft Teams connector action “Post message to user/channel”. For email, use Office 365 Outlook connector “Send email”. We might notify the device’s assigned user (“We noticed your device hasn’t been online, please confirm if it’s in your possession.”) and/or notify an IT admin group.
   * Optionally, add an action to update a tracking database (Dataverse table) logging the event and action.

- `/LockLostDevice(DeviceID: Trigger.DeviceID)` to explicitly invoke that action when certain conditions are met in the narrative instructions\.

- We can include guardrails like “only wipe if critical, otherwise just lock” as part of the instructions\. Since instructions are in natural language backed by the agent’s reasoning, we ensure to be explicit: \*“HARDY should only wipe a device if explicitly instructed in the incident playbook \(for example, confirmed stolen\); otherwise prefer remote lock\.”\*

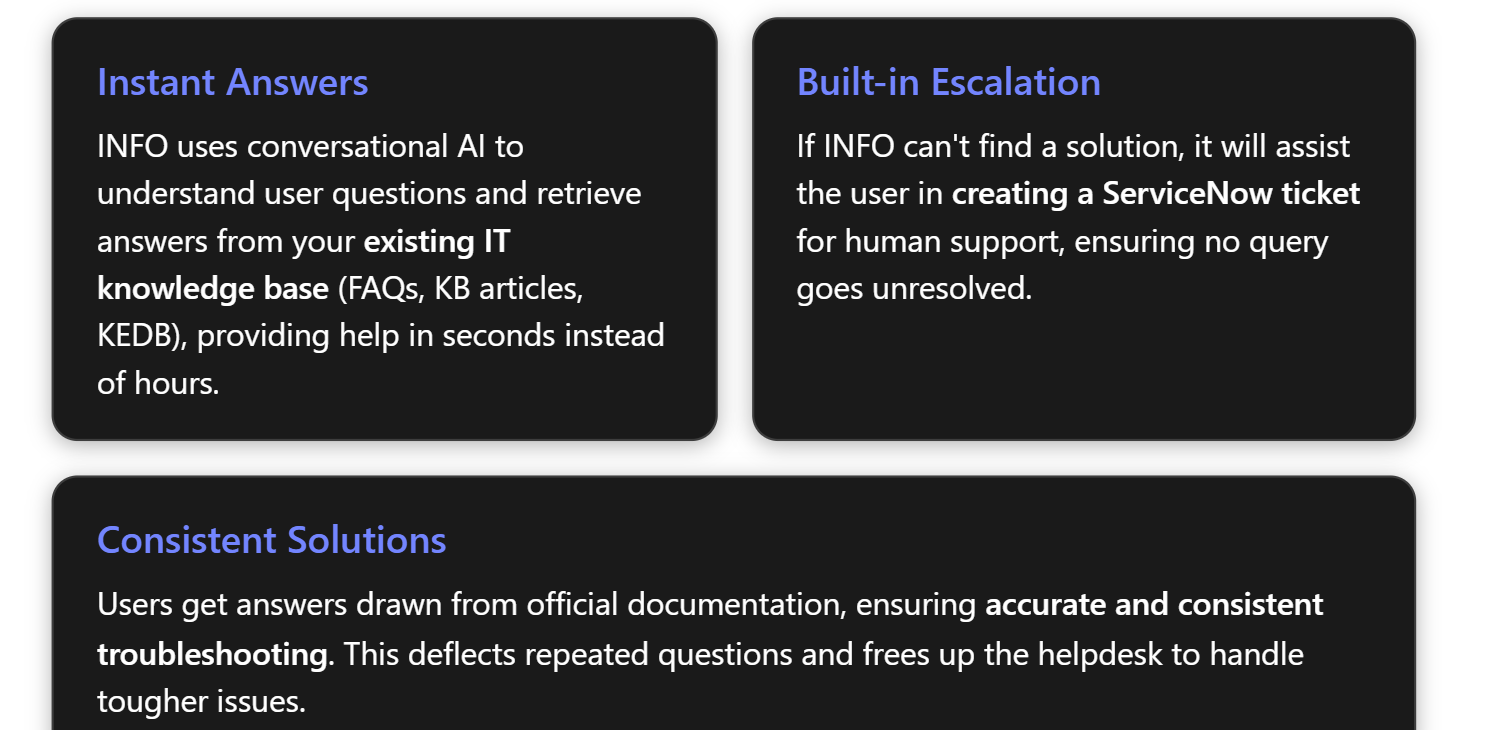
- Also instruct HARDY on data handling: e\.g\., \*“When notifying, do not include sensitive device data to unauthorized persons\.”\* \(This is a security consideration to explicitly mention if needed\.\)

## INFO – Knowledge Base Informational Agent (Conversational)

**Purpose:** INFO is a conversational AI agent that acts as an **intelligent knowledge base assistant**. When a user (or even a support agent) asks a question or when INFO detects a relevant context (like a ticket description), it fetches answers from the organization’s knowledge repositories (FAQs, internal wikis, Known Error Database) and provides a concise, helpful summary. Essentially, INFO ensures that the collective knowledge of the IT team and past solutions is instantly available to help resolve queries **without waiting for a human**.

**Key Functions to Prioritize:**

* **Natural Language Q\&A:** Allow users to ask questions in plain language (“Why is my VPN disconnecting frequently?”) and have INFO understand intent and find the answer.
* **Knowledge Retrieval:** Integrate INFO with the company’s existing **knowledge bases**. Commonly, IT knowledge might reside in ServiceNow Knowledge articles, SharePoint sites, Confluence pages, etc. INFO should search these sources for relevant content 5. It should use semantic search or ML (so it can find answers even if the wording differs).
* **Summarization & Answer Generation:** Take the content of one or multiple relevant articles and generate a **summarized answer** for the user. The answer should be in easy-to-understand terms, possibly with step-by-step instructions if it’s a how-to. It should reference the source (so the user knows it’s based on an official article).
* **Fallback and Ticketing Integration:** If INFO cannot find a good answer (or if the user says the answer didn’t help), INFO should gracefully escalate – for instance, offer to create a ServiceNow ticket or connect to a human agent. The IT Helpdesk template already includes this logic: the agent can help create a ServiceNow ticket when it doesn’t have an answer 5.
* **Ticket Context Sensing:** When used by support staff or integrated in the helpdesk workflow, INFO could auto-suggest solutions. For example, when an IT agent opens a ticket, INFO could read the issue description and immediately suggest relevant KB articles or resolutions (this could be an internal usage of the same agent’s capability, possibly via Teams or the ticketing interface).
* **Continuous Learning:** Log which queries were answered well and which weren’t. Use feedback (thumbs up/down from users or case resolution info) to identify content gaps. Then update the knowledge sources or train the agent to improve next time.



**Design & Implementation Steps in Copilot Studio:**

1. **Agent Type – Conversational:** Create a new **conversational agent** in Copilot Studio named "INFO – Knowledge Base Agent". This agent will interact via chat (for instance, deployed in Microsoft Teams or on an intranet page) with users. Set the persona in the description to something like: “A helpful AI assistant that answers IT questions using our internal knowledge articles, and escalates if it doesn’t know the answer.”
2. **Connect Knowledge Sources:** The key to INFO is having access to the knowledge base:
   * In Copilot Studio’s **Knowledge** configuration, add sources. For example, connect to **ServiceNow** if the organization’s IT KB articles are there. (The IT Helpdesk template suggests integration with ServiceNow KB during installation 5 – we need credentials and instance URL to connect.)
   * If there are knowledge articles in SharePoint or files (PDFs, docs), use the **Add knowledge source** feature to ingest those. Copilot Studio might allow adding SharePoint sites or uploading documents to an internal index.
   * If a Known Error Database (KEDB) exists (perhaps entries of past problems and fixes), ensure those entries are accessible – either via the SN KB or by exporting them into a readable source for the agent.
   * We should configure the **descriptions** of these knowledge sources appropriately (as Copilot Studio uses those to decide when to use them) 6. For example, label one source "IT Knowledge Base Articles – contains solutions for common technical issues".
3. **Define Intents/Topics:** Since this is mainly an open Q\&A agent, we might rely on the generative AI to handle a wide range of questions. However, we can define some structured **topics** in Copilot Studio for known categories:
   * **Troubleshooting** topic: to handle when user describes a problem (“My email won’t sync” or “I have a VPN error”). This topic would use the generative knowledge search. Essentially, under this topic, we configure the agent to search the connected KB and form an answer.
   * **Procedure/How-To** topic: for requests like “How do I install printer” or “How do I access my payslips” (if IT also covers some HR IT systems). These might fetch specific knowledge articles or have static answers if not found.
   * **Ticket Status** topic: If integrated with SN, allow questions like “What’s the status of ticket INC12345?” The IT Helpdesk agent template does this 5. We can add a tool using the ServiceNow connector to query ticket status by ID, and an intent trigger when user says “status” or provides an incident number. This is a nice-to-have that leverages integration.
   * **Fallback/Escalation** topic: A catch-all when the agent is unsure. It can apologize and offer to create a ticket. Possibly trigger this when confidence in answer is low or after a negative feedback. The template mentions this scenario, and even suggests an engagement with a human support rep if configured 5.

Copilot Studio’s **NLP** capabilities will help route user queries to the right topic/intent 1. We should add some example trigger phrases for each topic to train the model. For instance, under Troubleshooting: “not working”, “error”, “issue with”, etc., under How-To: “how do I”, “procedure”, “way to”, etc.

1. **Add Tools/Actions (if needed):**
   * INFO’s primary “action” is retrieving knowledge, which Copilot Studio might handle natively if knowledge sources are added. However, we will also add an action to **create a ServiceNow ticket** (for escalation). Use the ServiceNow connector to set up a *“CreateIncident”* action. This was likely pre-configured in the IT Helpdesk template as well. It should capture the user’s issue description, perhaps the conversation thread, and create an incident. Ensure to include the user’s info (name, email) in the ticket fields.
   * If doing ticket status queries, add a *“GetIncidentStatus”* action similarly.
   * If we want to allow a human handoff (say to a live agent on Teams or an engagement hub), that’s a more advanced integration – often it requires a live chat takeover. This might be outside Copilot Studio directly, but mention it as possible extension (the template suggests using an *engagement hub* for escalation to a person 5).
2. **Write Agent Instructions:** In the instructions (which guide how the AI organizes its actions and responses), include guidance like:
   * “When the user asks a question about IT or technical issues, search the knowledge base sources for relevant answers. Provide a concise explanation or solution. If steps are involved, list them clearly. Use a friendly, professional tone.
   * Only use information from the approved knowledge sources to answer (do not invent an answer).
   * If the question is not answerable from the knowledge base or not related to IT support, politely say you cannot help with that.
   * If you cannot find an answer, suggest creating a support ticket. If the user confirms, call the CreateIncident action and then give the user the ticket number and next steps.
   * If the user asks for the status of a ticket or says they already have a ticket, use the GetIncidentStatus action to retrieve it.
   * Always ensure to summarize in your own words rather than quoting large text, but you can quote key details if necessary (for example, error codes or exact setting names).”
   * Also instruct the agent to clarify if needed: e.g., if a query is ambiguous (“My application is not working”), the agent might ask which application or ask for an error message.
   * Persona-wise: specify that INFO should be *helpful and factual*, not overly verbose. We likely want answers that are a few sentences or a short list of steps, plus an offer for more help if needed.

These instructions align the agent with using its knowledge sources and when to escalate. Since Copilot Studio can orchestrate knowledge retrieval in generative mode, the key is telling it what to do if knowledge is lacking.

1. **Test with Sample Queries:**
   * Query examples:
     + “I can't connect to VPN on Wi-Fi.” – The agent should find if there’s a known article (for instance, maybe there's an article “VPN drops on wireless – how to fix”). Test if it returns a solution (perhaps: “We have a known solution: try enabling the ‘reconnect automatically’ setting, and ensure you have the latest Wi-Fi driver.”). If none is found, check if our knowledge base is comprehensive – maybe we add a Q\&A entry for that case.
     + “How do I reset my email password?” – This might overlap with KEYNOVA’s domain. INFO might find an article instructing to go to a portal, etc. Alternatively, if we prefer such requests be handled by KEYNOVA, we could configure INFO to recognize “password reset” queries and transfer to KEYNOVA. (This is interesting: Copilot Studio might allow one agent to call another agent’s action. Or simpler, we let INFO answer with “I can help you reset it securely – please provide some info” and actually that triggers KEYNOVA’s flow behind the scenes. Cross-agent coordination is advanced; maybe out-of-scope for now. We ensure INFO’s answer aligns with what KEYNOVA would do.)
     + “What’s the status of ticket INC1001?” – Ensure it calls the GetIncidentStatus and returns something like, “Your ticket INC1001 is in progress, assigned to the Network team, last updated 2 hours ago.”
     + “I need help, nothing is working!” – This broad query might not match a specific article. See if INFO defaults to a gentle clarification or directly to escalation. We might adjust: maybe respond with a clarifying question: “I’m sorry to hear that. Could you tell me which system or application isn’t working? I’ll try to assist.”

**Challenges & Mitigations:**

* *Coverage:* The agent is only as good as the knowledge it has. We must ensure our KB is comprehensive and up-to-date. Mitigation: do a content audit, prioritize top 50 common questions/issues and verify we have good answers for them.
* *False Confidence:* AI sometimes might present an answer that sounds confident but is slightly off. By grounding INFO strictly to provided sources and instructing it to escalate if unsure, we reduce hallucination. We’ll also test critical answers manually. Users should be reminded that if something seems off, they can always ask for a human or say it didn’t help (triggering escalation).
* *Access Control:* Some knowledge might be for IT eyes only (like confidential incident post-mortems). We should ensure the sources we connect are those safe to share with the general employee base. If needed, we can configure multiple instances of INFO or use Copilot’s built-in identity context to restrict certain content. For now, we assume the knowledge base is for general IT solutions that anyone can see.

## KEYNOVA – Password Reset Agent (Conversational)

**Purpose:** KEYNOVA is a specialized conversational agent focused on one of IT’s most frequent and time-consuming tasks: **password resets and account unlocks**. Its mission is to securely handle identity verification and credential resets autonomously, so users can regain access to their accounts within seconds, without involving the helpdesk. This agent directly tackles the high volume of password-related tickets, freeing IT staff and improving user productivity (no more waiting on hold just to reset a password).

**Key Functions to Prioritize:**

* Voice enabled Password reset request and redirect to <<external portal>>
* **Audit**

**Design & Implementation Steps in Copilot Studio:**

1. **Agent Type – Conversational:** Create a new conversational agent “KEYNOVA – Password Reset Assistant”. Describe it as: “Helps employees securely reset their passwords or unlock accounts through an automated process with identity verification.” This sets the stage that it will primarily be invoked by user requests.
2. **Intents & Trigger Phrases:** Define a topic or intent for password reset requests. Likely triggers include phrases like “forgot my password”, “reset my password”, “unlock my account”, “can’t log in”. We’ll configure these as trigger phrases in Copilot Studio’s Topics section. A single topic “Password Reset/Unlock” can cover both cases by conversation:
   * If the account is locked vs password forgotten might be distinguished by user input or by trying one method then the other. But we can handle both similarly: the fix is resetting credentials in both cases, essentially.
   * Also consider: if this agent is used via chat when the user is already logged in, they presumably still know their current password (or maybe not if account locked). But often, a user with a forgotten password can’t log in at all (so how do they access the chatbot? Possibly via a different device or a chat not requiring AD login, e.g., a web widget after identity verification, or by interacting with a voice agent like GENIE on phone). However, we assume at least some interface is accessible, maybe the user is on their phone or a kiosk chat where the agent first asks, “Please enter your employee ID or email to identify yourself.”
   * We might implement conversation differently depending on context (if the user is chatting from an authenticated session vs not). For now, assume they can access a chat interface where they input identifying info.

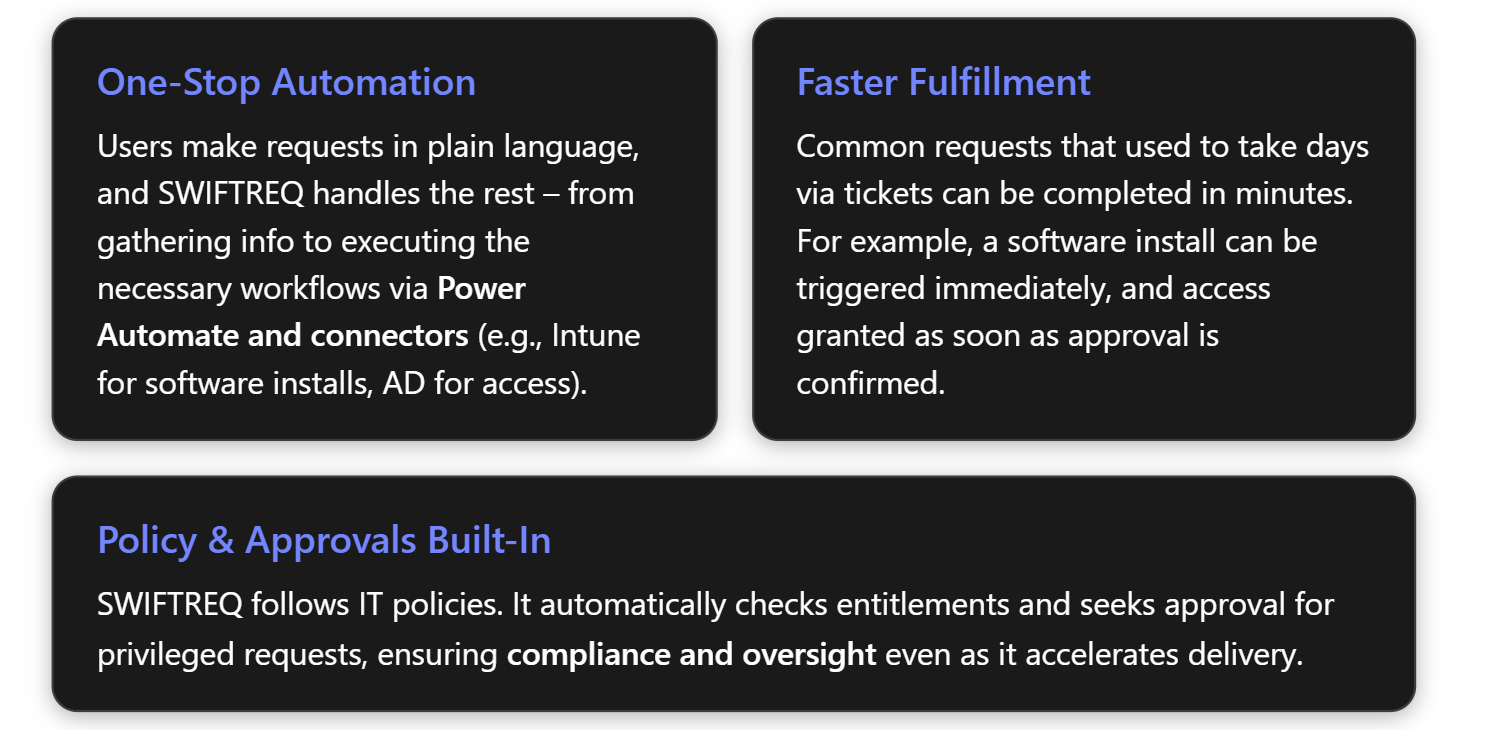
**Example Scenario:** An employee, Sarah, forgets her domain password on Monday morning and is locked out. Instead of calling IT and waiting, she grabs her smartphone, opens the company’s IT support chat (with KEYNOVA). She types “I forgot my password.” KEYNOVA asks her to verify – she receives a text code on her phone and enters it correctly. Immediately, KEYNOVA resets her password and tells her that a temporary password has been emailed to her personal account. She uses that to log in and then sets a new password. The entire process took 2 minutes and she’s back to work. Meanwhile, a ServiceNow incident was logged automatically noting that a self-service reset was done for audit. Multiply this by hundreds of incidents a month, and we see how KEYNOVA drastically reduces downtime and IT workload.

## SWIFTREQ – Request Fulfillment Agent (Conversational)

**Purpose:** SWIFTREQ is a conversational agent designed to **handle IT service requests** with speed and accuracy. Where KEYNOVA tackled a specific use case, SWIFTREQ is more general: it automates tasks like software installation requests, access requests, or other routine IT service catalog items. When a user needs something (access to a shared drive, a new software, a new hardware peripheral, etc.), instead of filling out a form and waiting days, they can ask SWIFTREQ and have the process completed (or at least initiated) in minutes. This agent aims to expedite fulfillment of common requests and ensure they are done correctly without human error.

**Key Functions to Prioritize:**

* **Software Installation Requests:** E.g., "I need Visio installed on my machine." SWIFTREQ should be able to create or trigger the deployment of that software if the user is entitled to it. This might involve checking license availability, then pushing the software via Intune or another deployment tool.
* **Access Requests:** E.g., "Please give me access to the Marketing file share" or "I need to be added to the Project X Teams channel." The agent should gather necessary info (which share or system, and reason if needed), then either directly grant access (if pre-approved) or kick off an approval workflow (manager approval) and then grant access.
* **New Hardware/Peripheral Requests:** For items like requesting a new keyboard, a laptop upgrade, etc., SWIFTREQ could auto-create a ServiceNow request or order if within policy. Perhaps not immediate fulfillment (since physical delivery is needed), but it automates the request logging and approval steps.
* **Service Ticket Creation for Complex Requests:** If a request is unusual or cannot be fully automated, SWIFTREQ should smoothly create a ticket with all details gathered, handing it to IT to follow up. The user gets confirmation that it’s in process.
* **Multi-step Workflow Orchestration:** Many requests have multiple steps (e.g., new hire onboarding: need to set up account, assign laptop, grant app access). While that’s a complex one, SWIFTREQ could eventually chain actions to do multi-step tasks in sequence. Initially, prioritize simpler single-step tasks.
* **Integration with Approvals:** Ensure that if certain requests need management approval or budget approval, the agent integrates with an approval system (Power Automate Approvals or ServiceNow approvals). The agent can either wait for approval (if the chat can be async) or inform the user that it’s pending approval and will notify once done.



**Design & Implementation Steps in Copilot Studio:**

1. **Agent Type – Conversational:** Create the agent "SWIFTREQ – Request Fulfillment Bot" as a conversational agent. It will interact with users through chat (likely in the same IT support Teams channel or a portal as INFO and KEYNOVA). In description, clarify: “Handles IT service requests (software, access, etc.) by automating workflows and forms – fulfilling requests quickly or routing to IT with details.”
2. **Identify Top Use-Case Topics:** It's wise to start with 1-2 request types before expanding. Based on pain points, let’s start with:
   * **Software Installation** and **Access to Shared Resource** (like a shared folder or application access).

We’ll develop these as separate conversation topics in Copilot Studio:

- \*\*Software Request\*\* topic:

- Trigger phrases: “install \\\[software\]”, “need \\\[software\]”, “get me \\\[Visio\]”, etc\.

- Dialog: Ask on which device \(if user has multiple or if needed\), confirm the version if applicable, and confirm license eligibility\.

- Action: Use Intune or software deployment connector to install\. Microsoft Intune has Graph APIs to assign a software package to a device or user\. Alternatively, if not using Intune, perhaps trigger an SCCM deployment or send a request to an IT technician\.

- If license required \(like Adobe or something limited\), either check a licensing system or just proceed if within policy and later true\-up\. If license pool is trackable, we could integrate with a licensing DB to subtract one or ensure availability\.

- Provide feedback: “Alright, I have initiated the installation of Visio on your machine\. It should install in the next 15 minutes\. I’ll let you know when it’s done\.”

- Also possibly log a behind\-the\-scenes change record if needed for compliance\.

- \*\*Access Request\*\* topic:

- Trigger: “access to X”, “permission to Y”, “rights to folder Z”, etc\.

- Dialog: Identify the resource and permission level if relevant\. For example, user says “I need access to the Finance SharePoint site\.” The agent should confirm: “Access to Finance SharePoint – are you a member of finance or is this for a project?” \(Maybe not needed, could just proceed to request\.\)

- Check policy: maybe certain accesses require manager approval by default\. The agent can say, “Company policy requires manager approval for Finance data access\. Shall I send an approval request to your manager?” If yes, it can use Power Automate Approvals: send manager an actionable message \(via Teams or email\)\.

- Action: If approved \(the manager clicks approve, which triggers flow back\), then add the user to the appropriate AD group or SharePoint group\. Use connectors: e\.g\., Azure AD connector “Add user to group” if the resource is controlled by group membership\. Or SharePoint connector to grant access directly\.

- Notify: “You now have access to the Finance SharePoint site\. Please try accessing it; you might need to log in again\.”

- If manager denies or times out, inform user politely it wasn’t approved \(and maybe create a ticket for record with that outcome\)\.

- \*\*General/Other Requests\*\* topic:

- We can have a fallback if the request doesn’t match a known automated path\. Like if someone says “I need a new laptop battery”, which we didn’t automate, the agent can capture details and create a ServiceNow request \(which will then be handled by IT\)\.

- This ensures SWIFTREQ still helps by gathering info \(“For a new battery, I’ll need your laptop model and asset tag\.” – user provides – “Thanks, I have submitted a request to IT\. You’ll receive an update soon\.”\)\.

- Use ServiceNow connector to create a request ticket with category “Hardware request” and the info gathered\.

- We should implement the structured ones first \(software, access\) and use the general fallback for others\. Over time, we can add more structured flows as they become common\.

* **Integrations/Tools Required:**
  + **Intune / Software Deployment:** Use **Graph API or Intune connector** to assign software. For example, if the organization uses Intune Win32 apps or Microsoft Store for Business, etc., we might have an app ID for Visio that we can assign to the user/device. Alternatively, we can trigger a **Power Automate flow** that uses an SCCM connector or even an RPA script to start the deployment. For now, assume Intune: we add an action “DeploySoftware(softwareName, deviceID)” that calls a flow or a script to deploy.
  + **Active Directory / Azure AD:** For access, an action “AddUserToGroup(group, user)” using Azure AD connector. Or if it’s a SharePoint resource, use the SharePoint connector action to grant site access.
  + **ServiceNow:** For logging any requests or for the fallback scenario, use SN connector’s create record (similar to incident but likely a “Request” record type or an incident with category request).
  + **Approvals:** Leverage **Power Automate Approvals** or **Adaptive Cards via Teams** for manager approval. We can implement this by a flow: agent calls “RequestApproval(manager, requestDetails)” which triggers an approval. The challenge is that approvals are asynchronous – the agent might not wait in chat for minutes/hours. Two ways:
    - Synchronous approach: If the manager is available to approve quickly (say via Teams, we ping them immediately, and if they click approve in 1-2 minutes, we can keep the user waiting a bit with a message “waiting for approval…”). Copilot Studio might allow a waiting mechanism for a short period.
    - Asynchronous: Tell user “Your request has been sent for approval. I’ll notify you once it’s approved and processed.” Then the agent or another mechanism can message the user when done (maybe via proactive message in Teams or email).
    - Possibly use the Engagement hub or maintain context. For now, design assuming fast approval for demo purposes, but mention asynchronous follow-up is possible.
  + The **Power Platform connectors** library has connectors for all these (Azure AD, Intune, ServiceNow, etc.) 1, which we will use inside Copilot Studio.
* **Dialog Design for Topics:**
  + Each structured topic will have a guided conversation:
    - For software: ask software name (if not already clear or to confirm version), possibly check entitlement (could call a small action that checks if user’s department or license count is available), then confirm installation.
    - For access: ask the resource name (maybe offer suggestions if we have a known list, but that’s advanced), ask reason if needed (optional but could log), handle approval if needed.
  + Use Copilot Studio’s conditions to handle approval outcome: e.g., a branch in the flow after calling approval: one path if approved, one if denied.
  + For each, after fulfillment action, confirm to the user.
  + If any step fails (e.g., Intune returns error or group not found), fall back to “I couldn’t complete this automatically, I will have IT look into it.” and create a ticket.
  + The conversation should feel like a natural chat, even though behind scenes we may be calling multiple steps. We might have to insert a slight delay or typing indicator especially if something takes a bit (like kicking off software install may not have immediate confirmation – we could choose to either trust it’s queued and say “It’s on the way” immediately, or actually poll for completion. For simplicity, we’ll not wait for completion, just assume it will happen.)
  + Instruct the agent to keep the user informed: if an approval is needed, clearly state it; if waiting on something, maybe say “processing your request…”.
* **Agent Instructions (Overarching):** Provide guidelines like:
  + “When the user requests something, determine the category (software, access, hardware, other) and follow the corresponding procedure.
  + Use the available Tools to fulfill the request whenever possible. Ensure any necessary approvals are obtained first.
  + If a request is outside your scope or fails, politely inform the user and create a ticket for human follow-up.
  + Speak in a helpful, can-do tone, but set expectations if a request will take some time (e.g., installation in progress or waiting for approval).
  + Adhere to policies: do not grant access or install software that violates policy (for example, if user requests admin rights or unapproved software, the bot should not do it — instead escalate to IT with a note).
  + Confirm completion or next steps so the user is confident their request is being handled.”
  + These instructions will help the AI choose the right action and know when not to comply (like the admin rights example: we can include “If user requests elevated admin privileges, do not auto-grant; escalate that request.” We likely will catch that under “other requests” which go to IT with justification).

**Examples of expansion:** In future iterations, SWIFTREQ could handle things like onboarding a new employee (covering multiple actions), scheduling a loaner device, or even non-IT requests (ordering office supplies) if integrated with other systems. The Power Platform connectors (over 900 available) enable integration with services like SAP (for procurement), HR systems, etc., making the scope virtually limitless 1. The “art of the possible” here is that any repetitive process that can be flow-charted and has system APIs can be handed over to the agent. It's truly creating a digital worker that takes on service desk tasks.

1. [Building AI Agents with Copilot Studio: Step-by-Step Implementation Guide](https://www.itmagination.com/blog/ai-agents-copilot-studio-implementation-guide) [↑](#footnote-ref-1)
2. <https://adoption.microsoft.com/files/copilot-studio/Autonomous-agents-with-Microsoft-Copilot-Studio.pdf> [↑](#footnote-ref-2)
3. [IT Helpdesk - Microsoft Copilot Studio | Microsoft Learn](https://learn.microsoft.com/en-us/microsoft-copilot-studio/template-it-helpdesk) [↑](#footnote-ref-3)