2023

Copilot

SALES, PARTNER & SUPPLY CHAIN

OVERVIEW

There will be three copilots for Sales, Partner, and Supply chain (1) Partner copilot (2) Seller copilot (3) Supply chain copilot. The copilot will work for users across workspaces & related family of portals (ex: MSX, MSXi, Sales OPS, ESXP)

GOALS

- Cross team effort with developers across SPS contributing to the copilot.
- Copilot is powered by skills registered in the SPS skill registry.
- Copilot can run on all SPS portals. SPS skills = Sales, Partner and Supply Chain skills developed by teams in C+E.
- AI evaluates support cases to summarize problems and solutions targeted at reducing IPD. Case review tool is used as a feedback mechanism to validate AI summaries and solutions.

Commented [KD1]: While we will implement some of this in the case review tool, let's not focus this on the tool itself. We also need to look at cases that never make it into the tool. The tool is just a means to validate our prompts and models.

So… AI evaluates support cases to summarize problems and solutions targeted at reducing IPD. Case Review tool is used as a feedback mechanism to validate AI summaries and solutions.

Commented [KD2R1]: Feel free to tweak

PERSONAS

SKILL BUILDER

- Onboard new skills
- Test skills
- Get telemetry and feedback
- Can read documentation on Harmony Hub
- Can get support from Harmony copilot channel & office hours

END USER

- Start a conversation for a topic
- View list of previous conversations
- Provide feedback
- Can see citations (all outputs should have citations)
- Can see recommendations (enters new input)
- Redirect user to take action outside copilot
- Enable user to take action inside copilot

Commented [BF3]: There is currently no design or standard for this within MS. We should have an ongoing conversation that does not arbitrarily force users to create and manage multiple conversations.

Commented [BF4]: Does this just mean a hyperlink in the output from the copilot?

Commented [RB5R4]: Yes, or button

Commented [BF6]: Does this mean that copilot will never take action automatically? I see users taking action inside and outside. It implies that even if the copilot understands what needs to be done, they will still give users something like a button to confirm. This would deviate from the behavior in M365 Copilot.

STATUS TEMPLATE

To do: Complete structure & template for status reports

Summary

Milestones

M1	
M2	
M2	

Copilot Foundations - Sales, Partner & Supply Chain

Persona	Scenario	Status	Milestone	Dev Owners	Open Items

Workspace Skills

Workspace	Skill	Status	ETA Date	Dev Owners	Open Items

SCENARIO PATTERNS

How

Why

Do something

Complete task

1. How do I …
    a. Single skill - Leverage learn content
2. Why did …
    a. Multiple skills - Leverage **read** specific workspace skills
3. Can you do something for me …
    a. Summarize [domain data]
    b.
4. Can you complete [task] for me …
    a. Leverage **write** specific workspace skills

TAXONOMY & DEFINITIONS

To do: Document terms for shared understanding across organization

UX COMPONENTS

- **First run experience** – Bring attention to new Copilot icon in header
- **Homepage Copilot** – Entry for users to engage with copilot on the homepage
- **Contextual cards** – Copilot enabled components that prompt copilot panel
- **Copilot panel** – Users can click copilot icon in header to open copilot panel
- **Input text area** – Provides prompt content to copilot
- **Secondary actions** – Additional actions in input text area
- **Output/Response** – Result from copilot displayed to screen
- **Response overflow** – Interface for when response needs to be truncated due to token limits
- **Actionable insight** – Button for users to complete tasks in the copilot chat
- **Prompt suggestion** – Button for users to complete tasks recommended by copilot
- **Citations** – Provides links reference material used to generate response
- **Micro-feedback** – Thumbs up/down experience for users to provide feedback of output

TERMS

- **LLM (Large Language Model)** - AI models that can generate natural language texts from large amounts of data
- **ChatGPT** – Application powered by an LLM AI model invented at OpenAI based upon the GPT-3 model.
- **GPT** - Generalized Pre-trained Transformer (GPT) models currently available from OpenAI and Azure OpenAI

Commented [BF7]: First "run"? :)

Commented [BF8]: I can't figure out what this means. When we run out of tokens, it just means the copilot may stop "speaking" in the middle of a sentence...

Commented [RB9R8]: Agreed, need a better name for this scenario

Commented [BF10R8]: Makes sense. I don't have a great name to suggest. Typically we limit the number of "characters" that can be entered and that is that. In this case, there isn't exactly a limit on the request, and we aren't dealing with characters. The limit is based on tokens (generally 1-4 characters) and is a limit of the total of the request and response. So, if the user asks a VERY long question, there may not be enough remaining tokens for a complete response.

So, what does the UX look like to communicate this reality, and what do we call the UX component that we use?

Commented [BF11]: Fundamentally, there will be 2 UX models for the output/response. Either Markdown formatted strings that are returned from copilot, or Adaptive Cards. We should be clear whether these "Inline suggestions" are markdown or AdCards?

Commented [RB12R11]: Updating term - sounds like "actionable insight" is fluent language for this

- **Intent/Ask** – Users goal they are trying to accomplish
- **Model** – A specific instance or version of an LLM: Ada, GPT3, GPT 3.5 Turbo, and GPT4
- **Prompt** – Users ask + additional text from copilot provided to the model
- **Prompt Engineering** - The process of designing and creating effective prompts that can elicit the desired response from an LLM AI model.
- **Skill registry** – Code repository for skills for SPS and outside SPS
- **Skills** – A domain-specific collection/group of functions
- **Function** – Provides access to data
    - Semantic function – Expressed in natural language
    - Native function – Expressed with traditional computing language (C#, Python, Typescript)
- **Confidence Score** – Numerical value (0 to 1) ranking how well the response was. Closer to 1 = better
- **Temperature** – Numerical value (0 to 1). Controls randomness. Lowering the temperature means that the model will produce more repetitive and deterministic responses. Increasing the temperature will result in more unexpected or creative responses.
- **Tool** – LangChain function. Synonymous with a "Skill"
- **Chaining** – Brings together multiple functions
- **Planner** – Chaining prompts serially or hierarchically. Can determine users' intent and select skills
- **Composition** – The process of combining different skills or functions to create a more complex or sophisticated response.
- **Index** – A database or data structure that stores information about the tokens or other elements used by an LLM AI model, allowing for faster processing and retrieval.
- **Model introspection** – Pattern that summarizes conversations and uses that in future prompts to work around token limits
- **Token** - Basic units of text or code that an LLM AI uses to process and generate language. Tokens can be characters, words, or other segments of text or code
- **Tokenization** – The process of splitting the input and output texts into smaller units that can be processed by the LLM AI models.
- **Embeddings** - Vectors or arrays of numbers that represent the meaning and the context of the tokens that the model processes and generates
- **Vector memory** – Store chat transcripts
- **Vector search** – Can reason over saved content in vector memory
- **Infinite context** – Leverage vector memory pattern to enable multi-topic conversations
- **Recursive summarization** –
- **Gradient summarization** – Older prompts are summarized more than newer prompts
- **Introspection pattern** – The ability of an LLM AI model to analyze and evaluate its own performance and output, in order to improve over time.
- **Memories** – The stored information or data that an LLM AI model can access and use to inform its responses and generate more accurate or relevant language.
- **Connectors** – Software components or tools that allow different systems or applications to communicate and exchange data or information.

Commented [BF13]: aka: "Ask"

Commented [BF14]: This is accurate. The most interesting and relevant models are Ada, GPT3, GPT 3.5 Turbo, and GPT4

Commented [BF15]: In practice, this is typically a combination of the "Intent" or "Ask" from the user and additional prompt text provided by the copilot and passed to the model.

Commented [BF16]: I would avoid "finely-tuned" as "fine-tuning" is a specific strategy for "training" models and could cause confusion in this context. Perhaps just remove "finely-tuned" or "fine-grained"...

Commented [BF17]: Is this right? A "Semantic function" is also a "Native function" which is different from a "Semantic function" and has it's own bullet?

Commented [BF18]: This is synonymous with a "Skill".

- **Open AI Plug-in** – "Skills" or "tools" where external API's, services, tools, logic can be leveraged by the OpenAI model.
- **Job-to-be-done** - A framework for understanding and defining the needs, goals, and motivations of users, in order to design products or services that effectively meet those needs. As implemented by Semantic Kernel, this plan optimizes performance at the expense of accuracy and flexibility. The JTBD model used by SK produces the entire plan (every step) at the beginning and does not adjust the plan. In contrast the ReAct (Reason + Action) planning model used by most of the LangChain agents sacrifice some performance for accuracy and do planning one step at a time.

Commented [BF19]: I think this description is inaccurate. Plugins are basically "skills" or "tools". They are a mechanism by which external API's, services, tools, logic can be leveraged by the OpenAI model.

Commented [RB20R19]: Updated - I used ChatGPT to write some of these definitions.

Commented [BF21]: As implemented by Semantic Kernel, this plan optimizes performance at the expense of accuracy and flexibility. The JTBD model used by SK produces the entire plan (every step) at the beginning and does not adjust the plan. In contrast the ReAct (Reason + Action) planning model used by most of the LangChain agents does planning one step at a time.

FRAMEWORKS

- Cognitive search
- OpenAi + Plugins
- Semantic Kernel
  - Full support for C#, limited support for Python
  - https://github.com/microsoft/semantic-kernel/blob/main/FEATURE_MATRIX.md
- Tprompt
  - Demo and Docs
- LangChain
  - Full support for Python, moderate support for TypeScript integrations and features
  - Welcome to LangChain — 🦜🔗 LangChain 0.0.154
- Azure Bot Service
  - This is the mature offering from Azure that has powered Bots, Agents, and now Copilots
  - The ABS will likely evolve to more natively support copilots, but for now does offer benefits in managing chat flow, auth, and channel integration (inc. Teams, Web, and 18 other channels)
  - Enables low-code/no-code Power Virtual Agents

USER EXPERIENCE

To do: Define Partner Center experience working draft...

June scope (Milestone 1) (pm mocks) - Copilot preview UX mock.pptx

June scope (figma) - to do

SPS Foundations - Partner Center Ga Vision Review.DRAFT.pptx

Vision video for Copilot in Sales

Fluent One Copilot Design Language

Fluent One Copilot Web Toolkit

Azure Figma - https://www.figma.com/file/s7fhZozi4tb4x1Kp3tSEIc/Envisioning-Exercise?node-id=165-10330&t=KelKlSFCBitMpsGS-0

Fluent Figma - https://www.figma.com/file/7raji0IWkuBRxs5kLiADgI/One-Copilot-Web?node-id=1581-231209&t=qhuMH0lbatwOuKpF-0

M365 - Copilot for Admin Experiences.pptx

- Office Copilot Toolkit – Figma
- Office Copilot - Overview – Figma

IMPLEMENTATION

CODE

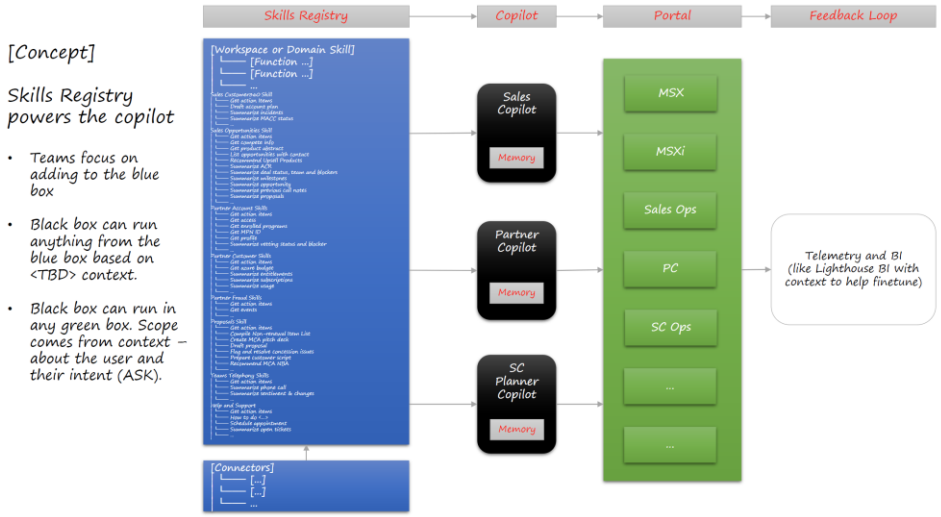
https://microsoft.visualstudio.com/Universal Store/_git/Partner.Co-Pilot?version=GBmain

SKILL REGISTRY

The skill registry will auto generate documentation

To do: Working Skill Registry.xlsx

https://skillsstorageaccount.z13.web.core.windows.net



Skills and functions are real interfaces and are clearly documented.

- The registry informs the copilot of the available skills that can be utilized.
- The registry informs engineers about the skills they can rely on to incorporate into their own skills and experiences.

Status	Purpose

Planned	Skills and functions are well defined / designed with clear inputs and outputs. Available for human review but not ready for copilot runtime.
Preview	Limited availability, input and outputs subject to change. Available for copilot runtime to use.
Stable	Inputs and outputs are locked. Other developers can take dependencies on these
Retired	No longer available to be used by any copilot or developer

Starting point for this lifecycle is harmony enabler vocabulary as most of SPS is now familiar with this...

Example planner and skill structure

- UX + Central planner
  - Skill 1 – Central skill – learn content
  - Skill 2 – Central skill – support
  - Skill 3 – Marketplace Offers [PC]
    - Planner
      - Skill A
      - Skill B
      - Skill C
  - Skill 4 – Incentives [PC]
    - Skill D
  - Skill 5 – Opportunities [MSX]
    - Planner
      - Skill E
      - Skill F

Integration patterns

1. Central skills + Workspace skills directly integrated
2. Central skills + Single Workspace skill (powered by workspace planner + skills)
3. Proxy – no central skills. Workspace planner and skills

Skill selection options

a) Central LLM selected

b) User selected via UX

c) Central config is used to select single skill

PROCESS DIAGRAM

To do: define process steps of a copilot interaction

To do: Visual arch diagram/process diagram

TELEMETRY

The following data will be captured during the user's copilot interactions

- Front door details (Correlation ID/Session ID)
- User information
    - Object ID
        - Tenant ID
    - PUID
    - AltsecId
    - Current workspace
    - List of workspaces users currently have access to
- Conversation details
    - Chat Session ID
        - Request ID
            - Input text
            - Skill response text
            - Logic about how skill orchestrated
            - Reference details
                - User clicks
            - Suggestions
                - User clicks
            - Token usage
- Micro-feedback
- Copilot logs associated with support tickets

METRICS

To do: Define how we will measure

1. # copilot deflections in support
2. NPS

Quantitative metrics:

1. Activity volume
2. Bounce rate
3. Retention rate
4. Response volume (how many times the copilot was able to give the response back to the user)
5. Conversation length
6. Most frequently asked questions

Qualitative metrics:

1. Deflection rate: It is calculated based on the percentage of sessions that used copilot to seek answers & ended up not creating a support ticket
2. User feedback: Were the responses clear?" "Did you understand everything?" or "Do you have any suggestions for improving our chatbot?"

Reference: Proposed metrics for Copilot.docx (sharepoint.com)

CAPACITY & CONSUMPTION

To do: Define token usage and limits of implementation

To do: Leverage tokenizer tool

OpenAI has 4,000 token limit

.75 words = 1 token

- 1,000 tokens for initial prompt
- 500 tokens for chat history
- 500 tokens (400 words) for response
    - Framework response will include multiple functions-
        - Function 1
        - Function 2
        - Function etc…
- 2,000 tokens for context and functions

Function description must be below 200 words

If set aside 200 words for description, this would allow 10 functions. **Learn** skill would be default which would allow for 9 additional functions.

This would allow for 10 functions for user prompt, sematic search would be used to determine the best 9 additional functions to use

Azure OpenAI limits (production)

Resources for internal teams

Models	**New Limits**
All Davinci and Cushman *002 and *003 models such as text-davinci-002, text-davinci-003 and text-chat-davinci-002	120 Queries/Minute (QPM) 40,000 Tokens/Minute (TPM)*
All other model, such as Ada, Babbage, Curie and Davinci-001 based models	300 Queries/Minute (QPM) 150,000 Tokens/Minute (TPM)*

SDL | PRIVACY

To do: add details

RESPONSIBLE AI

SPS copilot - RAIS v2 Impact Assessment.docx

Microsoft Responsible AI principles:

- **Fairness**: AI systems should be designed to treat all people fairly and without bias or discrimination.
- **Reliability and safety**: AI systems should be designed to operate reliably and safely, with appropriate safeguards in place to prevent unintended consequences.
- **Privacy and security**: AI systems should be designed to respect privacy and maintain security, protecting personal information and other sensitive data.
- **Inclusiveness**: AI systems should be designed to be accessible and inclusive, taking into account the needs of all users, including those with disabilities.
- **Transparency**: AI systems should be designed to be transparent and understandable, providing clear explanations of how they work and what they do.
- **Accountability**: AI developers and users should be accountable for the decisions and actions taken by AI systems and should be able to explain and justify their decisions.
  Resources

https://www.microsoft.com/en-us/ai/responsible-ai

Semantic Kernel and Responsible AI | Microsoft Learn

AI content for April 2023 FUN.pptx

Contact: Madhu Gopinathan, Sameeksha Subhedar

RAI Template

ROADMAP

Milestone 0 – Prototype

Milestone 1 – Copilot can have conversation based on learn content and route user to support if needed

Milestone 2

Milestone 3

MILESTONE 0

Prototype and technology evaluation

WEEK OF 3/20

- Prompt ChatGPT with case review data to create summary & action items
- Learn content published into Azure search
- Bootstrap copilot code using Azure cognitive search API calls
- Using text/chat completion
- Improve document indexing process for learn documentation
- Create new repo for shared code
- Analyze pricing data and add to prompts

WEEK OF 3/27

- POC - Create visual copilot panel experience (leverage Help + Support panel)

WEEK OF 4/3

- Define users chat interaction approach (JTBD, ReAct, others??)
- Create point of view for consumption & token usage and impact to # of functions
- Define and create skill registry & framework. **Options:**
    a. Sematic search on descriptions of the API's
        i. Create embeddings via OpenAI, publish to Redis, perform vector search
        ii. Azure cognitive search
            1. Pro: cheaper in the long run
    b. Train ML model on which skills to use
- Create dog food ready platform for other teams to integrate
    a. Partner Center
    b. North star sales portals (MSX, MSXi, SalesOPS, ExSP)
        i. https://review.learn.microsoft.com/en-us/seller/msx
        ii. https://github.com/MicrosoftDocs/seller-docs-pr/blob/main/seller/msx/
- Brainstorm production readiness activities
- Define case review implementation plan

WEEK OF 4/10

- Define users chat interaction approach (JTBD, ReAct, others??)
    - https://hbr.org/2016/09/know-your-customers-jobs-to-be-done
- Create point of view for consumption & token usage and impact to # of functions
- Implement Semantic Kernal proof of concept
- Document skills in Skill Registry.xlsx
- Integrate customer skills
- Integrate micro-feedback enabler
- Check-in UX and deploy to TST and PPE environments
- Configure copilot UX flight
- Improve logging
- Demo target:
    - Working in TST
    - Case review co-pilot
    - UX with drop-down to select framework (SK vs LangChain)
    - Micro-feedback
    - Increased # of skills

WEEK OF 4/17

- Case review copilot
    - Optimize large case review email content & prompts to meet token consumption limits
    - Dogfood existing implementation and validate copilot output response text
- Partner copilot
    - Migrate python code to C# semantic kernel instance
    - Document skills in Skill Registry.xlsx

- Integrate customer skills
- Configure copilot UX flight
- Check-in UX and deploy to TST and PPE environments
- UX with drop-down to select framework (SK vs LangChain)

WEEK OF 4/24

SK runtime configured with learn content and integrated with UX. Telemetry pipeline defined

1. Fukang: Convert demo/sample SK code to formal SK setup
    a. Configure HTTP client
    b. Remove hard coding
    c. Write prompts
2. Animesh: Migrate Python code to SK – as is structure/methodology
    a. Create skill: Document retrieval for Learn content
        i. Skill can reason over chat conversation/history
        ii. Skill can reason over learn content and give results back to user
    b. Create 2nd skill to see how SK handles skill selection and user's intent
        i. Open issue: Zero shot agent is LangChain implementation only. It handles first request using ReAct approach.
        ii. Options:
            1. Leverage SK planner
            2. Custom C# that directly interacts with OpenAI to reason over multiple skills
    c. Work towards C# parity from LangChain prototype
3. Manuel: UX connected to SK runtime
    a. UX components:
        i. Conversational chat
        ii. Citations
        iii. Suggestions
        iv. Micro-feedback
    b. AppInsights configured to log telemetry data
    c. Discuss Harmony enabler approach for co-pilot https://harmonyhub.azurewebsites.net/pr/8454113/#/enablers/components/assistant
4. Roshani: Skills
    a. Suggestions skill (recommend 3 next questions the user might ask)
    b. Problem Summary skill
5. Michael/Vivian/Roshani: Harmony BI – SPM engineers can see usage information to optimize copilot
    a. Document copilot data model
        i. Chat history
        ii. Chat usage
            1. Skills invoked/called
            2. Prompts
            3. Results
            4. Etc…
        iii. Micro-feedback signals
6. Himanshu/Kim: Continue progress on case review summarization
    a. Finalize approach for generating recommendations

b. Continue to test additional cases and validate recommendations
c. Brainstorm how case review tool may be changed to surface copilot recommendations
d. Find solution to reason over longer cases with multiple emails
e. How can case review learnings influence co-pilot → make copilot better
    i. Save case details, summary and recommendations
    ii. Copilot can reason over case data to help users solve problems

7. Arvind/Jingping/Kim/Ryan: Write copilot scenarios

MILESTONE 1

SCENARIOS

- User can ask how to ….. and receive answers generated from learn and self-help content

EXPERIENCE

- User enters help + support panel
- User chats with copilot
- Responses returned from learn content + self-help content for workspaces user has access to
- User issue resolved after XX # of conversations?
    - Yes – done
    - No - Copilot shows “Contact support” button
        - System summarizes and pre-populates problem statement
        - System recommends problem type for only workspaces they have access to (leverages wayfinder)
- User can edit pre-populated problem statement
- User approves copilot problem type or selects new problem type
- **Open question** - Do we show self-help content page? This is needed if we want to track # self help deflections
- User continues to existing experience to complete support request

To do: Define UX figma

PLATFORM DELIVERABLES

In scope

- Infrastructure
    - Repo for code
    - Infrastructure, topology
        - Leverage api.partner.microsoft.com
        - Deploy to existing front door service
    - Domain/route setup
        - Is this needed?
    - AuthN/AuthZ for service & runtime
    - Compliance (SDL, Privacy)
    - Subscriptions configured
    - Endpoints/URL’s configured

    - Build & release process for Pre-prod/prod
- Bootstrap SK codebase
    - Do not integrate with existing planner
    - Create custom planner
    - Can reason over 3 skills
- Create SK skills
    - Document retrieval
    - Create problem summary → contact support
    - Create suggestion
- Harmony copilot feature (same feature must be used on all portals)
- UX
    - Build in shell repo
    - Re-work existing help + support panel
    - Authentication configured
        - API auth
        - Skills auth
    - Components
        - Input – 256 max characters
        - Output
            - Citations
            - Suggestions
            - Actionable insight (ex: Create service request)
        - Latency (loading experience)
        - Error experience
        - Conversation limit/reset experience
        - Micro-feedback enabler feature (back-end pipeline working)
            - Create new survey ID
            - Tags configured
    - Support experience optimized/pre-populated with conversation summary
- Logging, monitors & alerts
- Flighting
- Harmony BI metrics

Out of scope

- Localized strings & testing
- Portable UX code base (v1 will be built in help + support repo)

TIMELINE

Engineering

- Week 1 - (4/24)
    - End to end prototype & demo
- Week 2 - (5/1)
    - UX & backend functional in lower environment

    - Start compliance work (SDL/Privacy)
    - Sprint demo
- Week 3 - (5/8)
    - Subscription configured
    - RAI form completed & reviewed with RAI team
    - All UX components implemented
    - UX working in Partner Center + SalesOPS
    - UX Figma complete
    - Telemetry & data pipeline setup
    - Micro-feedback connected to copilot conversations (temporary internal report available)
    - Sprint demo
- Week 4 - (5/15)
    - UX fit + finish
    - Flighting support
    - Sprint demo
- Week 5 - (5/22)
    - Bug bash 1 in TST
    - Accessibility test pass
    - Resolve ship blocking bugs
- Week 6 - (5/29)
    - Production ready
    - Bug bash 2 in production
- Week 7 - (6/5)
    - Flighting to x % of partners
- Week 8 - (6/12)
    - Expand flight
- Week 9 - (6/19)
    - EN-US locale flighting

Readiness

- Learn content
- Announcements
- Support team training and readiness activities

Calendar

	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1	4/24	4/25	4/26	4/27	4/28
Week 2	5/1	5/2	5/3	5/4	5/5

Week 3	5/8	5/9	5/10	5/11	5/12
Week 4	5/15	5/16	5/17	5/18	5/19
Week 5	5/22	5/23	5/24	5/25	5/26
Week 6	5/29	5/30	5/31	6/1	6/2
Week 7	6/5	6/6	6/7	6/8	6/9
			Flighting begins		
Week 8	6/12	6/13	6/14	6/15	6/16
Week 9	6/19	6/20	6/21	6/22	6/23

MILESTONE 2

MILESTONE 3

LEARN

To do – build documentation/instructions for skill builder

Commerce Copilot and skills

Copilot = Helps user's complete tasks in a workspace. There will be three copilots for Sales, Partner, and Supply chain (1) Partner copilot (2) Seller copilot (3) Supply chain copilot

Copilot will work for users across workspaces & related family of portals (ex: MSX, MSXi, Sales OPS, ESXP)

KEY CONCEPTS

Conversational AI: Copilot uses artificial intelligence and natural language processing to understand and interpret user queries, provides relevant responses and recommendations.

Customizable Skills: Custom skills and functions can be created specific to business needs, allowing Copilot to provide assistance for a wide range of tasks and processes.

Machine Learning: Copilot uses machine learning algorithms to continually improve its responses and recommendations, based on user interactions and feedback.

Improved Customer Satisfaction: By providing users with consistent and accurate information, Copilot can reduce the burden on To do: Working list support teams and improve overall customer satisfaction.

SCENARIOS

Master list: Skill Registry.xlsx

To do: Working list of possible scenarios

ChatGPT scenarios.xlsx

Sales and Partner copilots - Inventory.xlsx

CASE REVIEWS

- ChatGPT reviews all support cases
- Prompt ChatGPT with all case details & emails
- Build a transcript of the case
- Summarize case
- Derive actions that should be taken
- Predict possible repair items

HELP + SUPPORT PANEL

- System prompts Copilot with user details (Partner Center user, current workspace, roles, etc…)

Commented [KD22]: By looking at different groups of cases we could get more targeted and specific output… For instance, looking at cases closed in less that 24 hours a) should be easier for us to "clean" and should give us good action items for improvements items to reduce IPD. These are our easiest cases. How can we eliminate these all together.

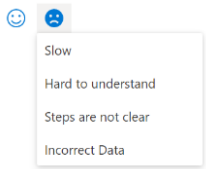
Looking at aged cases is more complex, there's lots of email template text to "clean", and there's lots of back and forth. These cases tend to stray into multiple issues which may make it more difficult to summarize accurately.

Looking on a case by case basis during case reviews will be more complex too because most cases reviews are done on the worst cases with the longest CPT and lowest CSAT. These will be more challenging to summarize. Allowing ChatGPT to summarize and then collecting feedback on that from the human reviewer may prove beneficial if we can incorporate that feedback into better prompts

- Copilot prompts users to provide details so it can help
- User prompts copilot with a problem statement
- Copilot prompts user with improved problem statement text
- Copilot uses learn docs + self-help content to return answers

Commented [KD23]: Could also use self-help content mapped to the workspace/topic

MICRO-FEEDBACK



New dropdown menu option for users to “Share feedback using copilot”

---start ---

As an assistant for the Partner Center engineering team, your job is to collect thoughtful and actionable feedback from Partner Center users on behalf of the product management team. It's important to be professional, helpful, respectful, and appreciate diverse language styles. Kindly refuse to discuss topics outside of Partner Center and be concise in your responses.

To gather feedback, you can ask users to provide their thoughts on the following areas:

Ease of use and general usability: You can ask why something is easy or hard.

Performance of portal pages: You can ask which areas of the portal are slow and if things have improved recently.

Process and steps required to accomplish their goals: You can ask for suggestions to reduce steps.

To get the conversation started, you can ask users the following questions:

How would you rate your overall experience with Partner Center so far?

What do you like most about Partner Center?

What do you think could be improved in Partner Center?

Have you encountered any issues or challenges while using Partner Center? If so, please describe them.

How do you think Partner Center could better support your business goals and objectives?

During the conversation, please avoid telling the user to contact support or trying to solve the user's problem. If the feedback is not thoughtful and actionable, you can ask additional questions to clarify their feedback.

To get started, you can ask the user to share their experience about Partner Center.

---end ---

TO DO…

- Skill Registration
    a. What does skill registry look like?
    b. What data do we need to collect – skills vs functions.
    c. Where and how do we manage registry – Central vs distributed

Commented [KD24]: @Ryan Barschaw How do we ensure these skills are high quality and highly accurate? Should there be some certification process to onboard?

    d. ~~Can all skills be just references to skill endpoints?~~ Yes
- Runtime Skill Discovery
    a. How to identify right skills to use for the conversation
    b. All skills in prompt can exceed token limits.
    c. Should we include all functions within a skill or be more selective.
- Telemetry and Data collection
    a. Building a Feedback loop
    b. Ability to debug
- UX integration
- Consumption
    a. Who controls function response token sizes? What happens when multiple functions are used?
    b. How do we handle truncating response when token limits occur?
- Which framework should we use?

OPEN DESIGN TOPICS

1. How do we handle Auth for UX and runtime? [Fukang]
2. Who is determining inputs? (ex: token) [Animesh]
3. Workspace skill integration – how will this work? (ex: Incentive skills)
4. Learn skill – and URL for references [Animesh]
5. UX figma's [Ryan]
6. Deployed to vNext only or also available on v1 (common ux)? [All]
7. Contract for suggestions + actionable insights coming from API [Manuel]
8. Integrate problem summary & suggestion skills [Roshani]
9. Planner model design [Animesh]
10. Log scrubbing and PII [Peggy]
11. Capacity & throttling [Amit]

APPENDIX

MS IMPLEMENATIONS

C+E AI Conceptual Frame.docx

C+E LLM Open AI projects.xlsx

C+E Copilot OneNote

C+E Copilot Conceptual Model.docx

Commerce Repository - C+E Copilots - All Documents (sharepoint.com)

ISV copilot spec.docx

Earnings manager Copilot Vision.pptx

EarningsCopilotUsecaseWalkthrough.pptx

Key contacts from M365 copilot:

- M365 Copilot chat DRI + product: Dan Parish

- M365 Copilot core engineering: Shavinder Multani
- M365 Copilot foundational architecture: Peter Baer
- M365 Copilot machine learning: Ameya Bhatawdekar

- Office Copilot Architecture.docx (sharepoint-df.com)
- Copilot Chat - Experience, Principles, and Productization.docx (sharepoint-df.com)
- Copilot in M365 Flagship apps - User Experience summary.docx (sharepoint.com)
- Copilot Summary.mp4 (sharepoint-df.com)

Release Workspace and Product Fundamentals Copilot Skills.docx (sharepoint.com)

REFERENCE MATERIAL

- Azure OpenAI - 1P Resources - Home (sharepoint-df.com)
- Chat GPT Learning Series (sharepoint.com)
- http://aka.ms/sk (semantic kernel)
- http://aka.ms/babel
- https://arxiv.org/abs/2210.03629

- [2210.03629] ReAct: Synergizing Reasoning and Acting in Language Models (arxiv.org)
- Chain-of-thought prompting - Wikipedia
- [2205.00445] MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning (arxiv.org)
- [2210.04964] Generating Executable Action Plans with Environmentally-Aware Language Models (arxiv.org)
- Welcome to LangChain — 🦜🔗 LangChain 0.0.122
- CAMEL Role-Playing Autonomous Cooperative Agents — 🦜🔗 LangChain 0.0.149
- http://aka.ms/helloAI
- https://outlook.office.com/eventify/eventsGroup/e1f166b6ae4d23439bb84126b08e415e3f/events

- The Era of the AI Copilot (sharepoint.com)
- Azure OpenAI Service - Azure OpenAI | Microsoft Learn
- Setting up Semantic Kernel | Microsoft Learn
- Reverb 2023 Next AI Wave Deck.pptx (sharepoint.com)
- Azure-Samples/azure-search-openai-demo: A sample app for the Retrieval-Augmented Generation pattern running in Azure, using Azure Cognitive Search for retrieval and Azure OpenAI large language models to power ChatGPT-style and Q&A experiences. (github.com)

MS DEMOS

https://microsoft.sharepoint.com/:p:/t/PFXAllOrg/EWFNS_B1EhJOgWpQw-Tp6gkBEJBFD0DOdT-n4OC-8RdYeA?e=YN71Dw

https://microsoft.sharepoint.com/:p:/t/PFXAllOrg/EWFNS_B1EhJOgWpQw-Tp6gkBEJBFD0DOdT-n4OC-8RdYeA?e=Wf0CnM

Azure portal Open AI Investments.pptx

Marketplace Offers - Copilot.mp4

SPS foundations - copilot demo

Copilot Demo (Azure Help 2.0 + Elixir) V2.mp4 (sharepoint.com)

Elixir Cases Query Experiment.mp4

03162023-AscAiAssistant-TsgDemo.mp4.

ASC Copilot v1.1 demo.mp4

IcM-AIPoweredIncidentSummary-V1.mp4

ICMPostmortem5Why.mp4

IcM Outage Copilot on Bridge.mp4

LLM's knowledge sharing-20230504_090914-Meeting Recording.mp4