CopilotKit

The Agent Protocol Landscape

Understanding AG-UI, MCP, A2A, UI Specs, and how to build agentic applications

Swipe to next page

Why Agent Protocols Matter

AI Agents are becoming the place where user intent turns into action. But how do they connect and communicate with the rest of the ecosystem?

Protocols

Agent Protocols are the language of interoperability between models, tools, apps and users.

How do these protocols work and help?

- **Interoperability →** agents, apps, and tools speak a shared format.
- **Transparency →** open standards instead of proprietary SDKs.
- **Reusability →** modular, composable building blocks for the ecosystem.

Protocols

AG-UI MCP A2A

Although protocols make agents *compatible*, the application layer makes them *collaborative.*

(more on this later)

The Current Established Protocol Ecosystem

The agentic ecosystem is rapidly organizing around a family of open, complementary protocols, each addressing a distinct layer.

You can connect your application to agents directly via AG-UI, MCP, and A2A.

Adopted Standards

Protocol	Maintainer	Purpose
MCP (Model Context Protocol)	Anthropic / Open Source	Defines structured context/tool access between models and clients.
AG-UI (Agent-User Interaction Protocol)	CopilotKit / Open Source	Connects agentic backends and agentic frontends.
A2A (Agent-to-Agent)	Google / Open Source	Enables secure messaging and coordination between agents from different frameworks.

These protocols are not competitors, but **complements**, forming a common language for agents, apps, and users.

Layers in the Ecosystem

The ecosystem is organizing into layers- each defining how agents interact with users, tools, other agents, and UI.

Layers Table

Layer	Protocol / Gen UI Spec	Purpose
Agent ↔ User Interaction	**AG-UI (Agent-User Interaction Protocol)**	The open, event-based standard that connects agentic backends/frontends, enabling real-time, multimodal, interactive experiences.
Agent ↔ Declarative UI	**MCP UI (Anthropic)** **Open-JSON-UI (OpenAI)**	Declarative, LLM-friendly generative UI specs that define what to render and how to structure agent responses visually.
Agent ↔ Tools & Data	**MCP (Model Context Protocol)**	Open standard (originated by Anthropic) that lets agents securely connect to external systems- tools, workflows, and data sources.
Agent ↔ Agent	**A2A**	Defines how agents coordinate and share work across distributed agentic systems.

Unified Takeaway

- AG-UI connects agentic apps to agentic backends.
- MCP connects agents to tools and data.
- A2A connects agents to other agents.
- MCP-UI, and Open-JSON-UI let agents return UIs.

Agent-User Interaction Protocol

AI agents are moving beyond chatbots and into products.

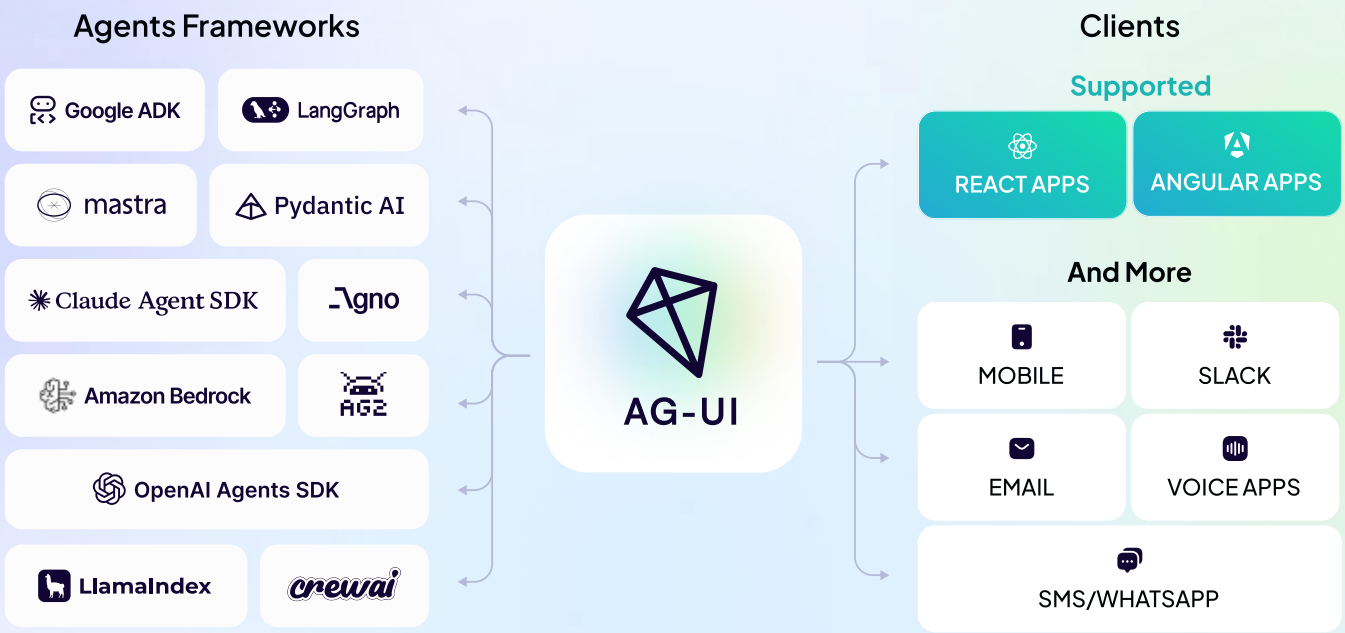
This is what's becoming known as **the application layer** - where users and agents collaborate directly inside interfaces.

What is AG-UI's role?

Standardizing *how humans & agents collab* inside real apps

When intelligence needs to *live inside an app* (updating UI state, responding to user actions, showing reasoning steps, or streaming outputs into a sidebar) you need an application-level protocol → **AG-UI**

AG-UI: a horizontal **"N-M" Protocol**



Why should I care about AG-UI?

It's the fastest-growing protocol in the agent-user domain

Every major platform is moving toward AI-native UX!

Think: Agent as chatbot → **Agent as co-worker inside my app**

AG-UI and Generative UI Specs

Several recently released specs have enabled agents to return generative UI, increasing the power and flexibility of the Agent ↔ User conversation.

MCP-UI and Open-JSON-UI are both **generative UI specifications**. Generative UIs allow agents to respond to users not only with text but also with dynamic UI components.

Despite the naming similarities, **AG-UI is not a generative UI specification!**

- **It's a User Interaction protocol** that provides the **bi-directional runtime connection** between an agentic backend & frontend.

AG-UI natively supports all of the generative UI specs below and allows developers to define **their own custom generative UI standards** as well.

Current Gen UI Specs Supported by AG-UI

Specification	Origin / Maintainer	Purpose
Open-JSON-UI	OpenAI	An open standardization of OpenAI's internal declarative Generative UI schema.
MCP-UI	Microsoft + Shopify	A fully open, iframe-based Generative UI standard extending MCP for user-facing experiences.

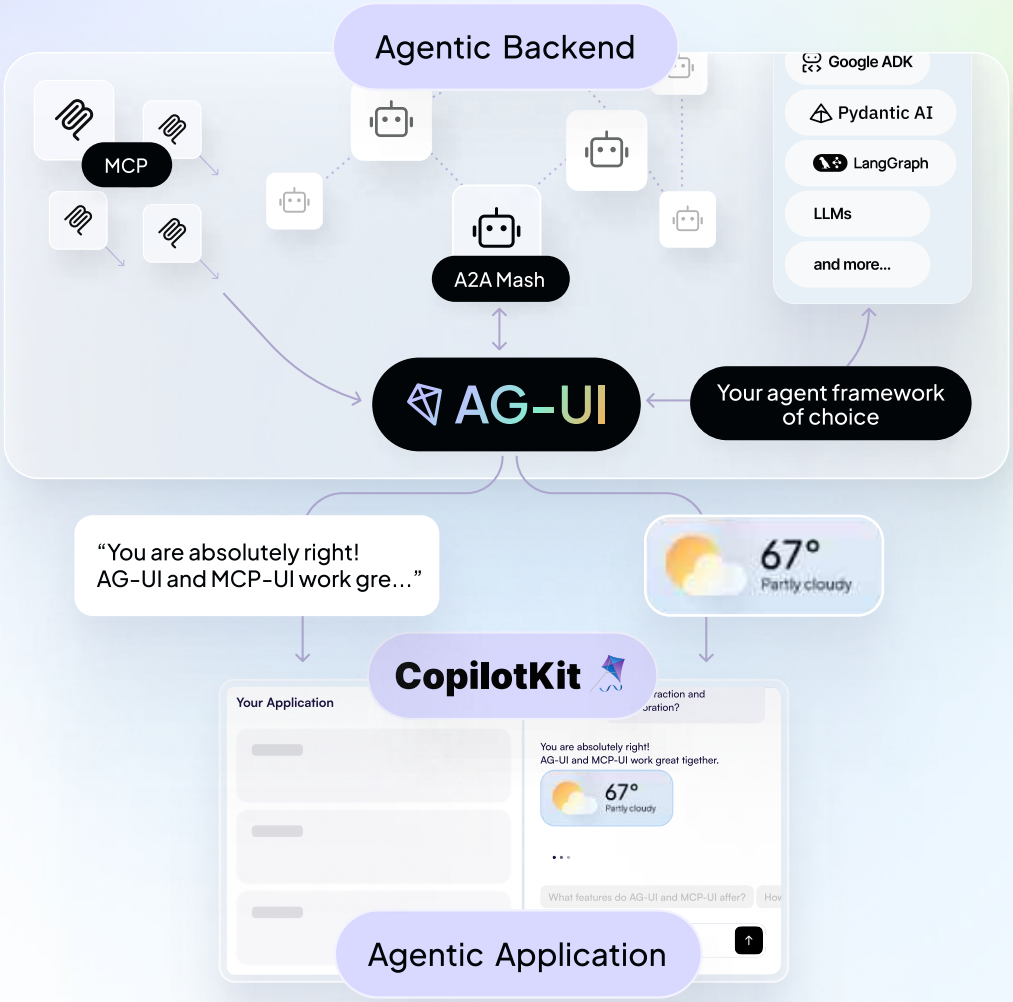
Mixing and Matching

CopilotKit lets developers connect to any of these protocols **directly or in combination.**

AG-UI also includes handshakes with both **MCP** and **A2A**, ensuring smooth interoperability across the full agentic stack.

This means that if your host agent connects to subagents using **MCP** or **A2A**, their UI properties can be propagated all the way through to the user-facing application- while preserving **full security, policy, and observability controls.**

AG-UI is a General Purpose Bi-directional
Agentic Frontend ↔ Agentic Backend Connection



You can think of AG-UI as the **"kitchen sink" protocol** - informed by bottom-up, real-world needs for building best-in-class agentic applications.

Common Misconceptions

Misconception:

AG-UI and MCP-UI are competing standards for agent UIs.

Reality:

Not at all. They serve completely different purposes. MCP-UI is a generative UI specification that defines what the agent should render visually. AG-UI, on the other hand, is a User Interaction protocol that defines how agents become interactive and stateful inside the product.

Misconception:

Protocols are competing for dominance.

Reality:

They're complementary - each solves a different part of the agent lifecycle. The goal is interoperability, not exclusivity.

Misconception:

The "Protocol Layer" only handles agent communication.

Reality:

It's broader. It standardizes all forms of interaction:

- Agent ↔ Tool (MCP)
- Agent ↔ User (AG-UI)
- Agent ↔ Agent (A2A)

Misconception:

Protocols are just APIs with new branding.

Reality:

APIs connect products; **protocols connect ecosystems.** They define shared schemas, security, and communication rules so independent systems can work together without central control.

Misconception:

AG-UI is a visualization spec like MCP-UI.

Reality:

AG-UI is a User Interaction protocol, not a UI schema. It powers the real-time, bi-directional connection between agents and users, enabling agents to stay stateful, multimodal, and interactive inside the app.

Misconception:

CopilotKit replaces these protocols.

Reality:

CopilotKit **sits above them** as the *Agentic Application Framework*. It unifies AG-UI, MCP, and A2A under one developer-ready layer- so you can build, connect, and operate agentic apps using any or all protocols.

This leads us to the next page: Protocol Handshakes

Handshakes Powering the Ecosystem

AG-UI ↔ MCP / A2A

AG-UI turns low-level protocol interoperability into application and human-level *collaboration.*

These handshakes expose protocol activity to users:

- AG-UI ↔ MCP → visualize tool outputs
- AG-UI ↔ A2A → visualize multi-agent collaboration

Read more: AG-UI Protocol Docs | Connect MCP Servers | A2A → Frontend

MCP ↔ A2A

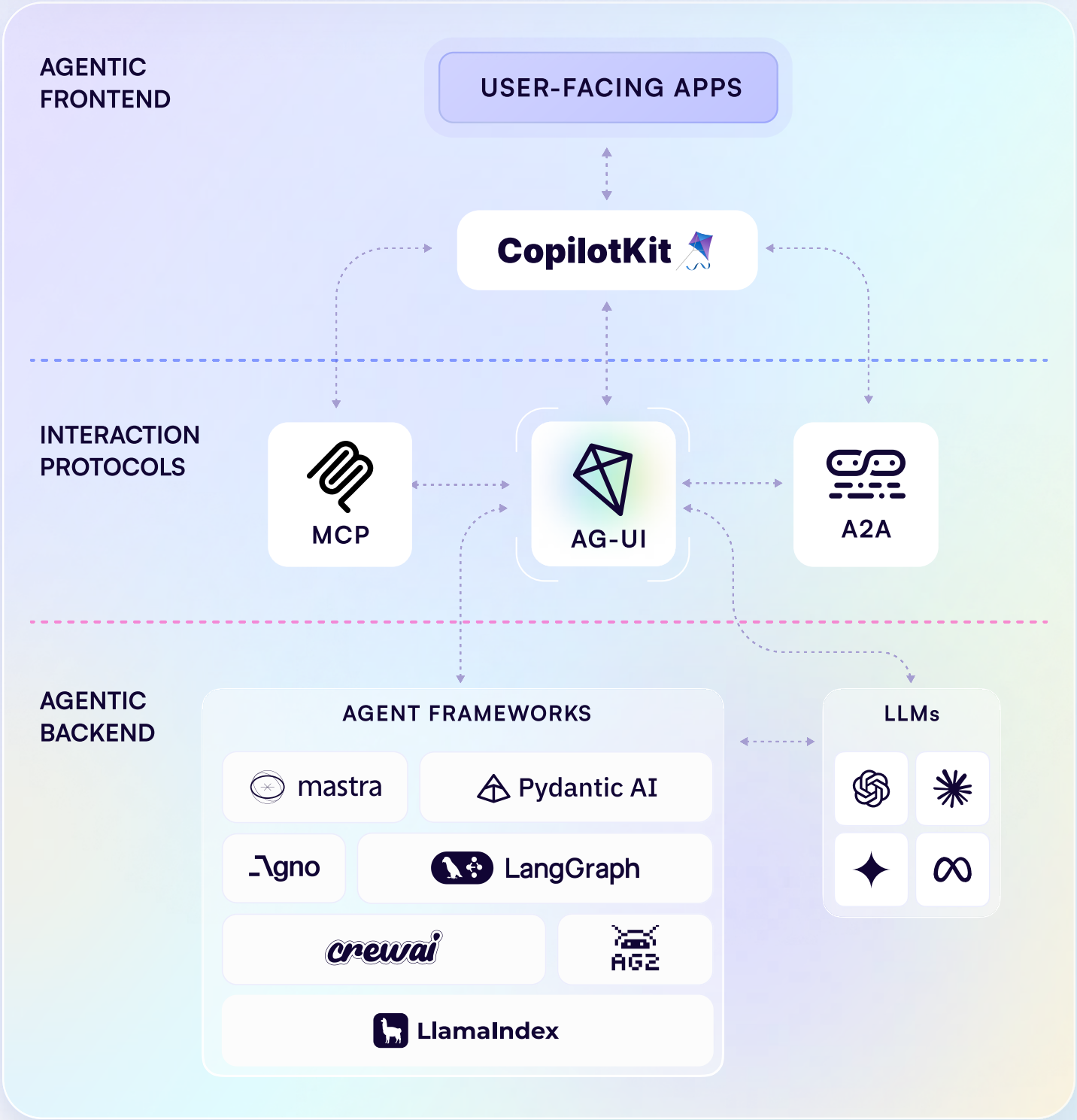
- **Function:** A2A doesn't let one agent *use* another agent's MCP tools directly. Instead, it allows agents to **negotiate, exchange goals, or delegate tasks,** and each agent can then use its own MCP connections to act.
- **Outcome:** Enables **multi-agent collaboration**, where agents coordinate and share outcomes, not shared tool access, but shared intent and coordination across systems.

Read more: MCP ↔ A2A Handshake

Ecosystem Overview

CopilotKit sits above these protocols and generative UI specs as the ***Agentic Application Framework***- an open-source and cloud platform that unifies the stack, enabling developers to build and operate production-grade agentic applications with confidence.

- CopilotKit uses any or all of the above, enabling developers to build rich user-facing agentic apps connected to any agentic backend through any of the Agent Interaction Protocols, and using any of the Generative UI Specs.



CopilotKit + AG-UI

The future is multi-protocol composability

Agents will speak *many* protocols at once.

CopilotKit

- CopilotKit is the **Agentic Application Framework** - everything developers need to integrate AI agents into their user-facing apps.
- CopilotKit-powered agentic apps can connect to any AI agent, either directly or through the Agentic protocol of their choice, including AG-UI, MCP, and A2A.

AG-UI

- **The Agent-User Interaction Protocol** is the **general-purpose, bi-directional connection** between a user-facing application and any agentic backend.

CopilotKit invites builders, protocol authors, and open-source contributors to shape how agents and humans interact.

AG-UI is fully open source and built in active collaboration with the broader protocol community.

Want to start building agentic applications?

Check out **www.copilotkit.ai/ag-ui**

More Resources

AG-UI

- **Repo:** github.com
- **Overview:** docs.ag-ui.com

MCP

- **Repo:** github.com
- **Site:** modelcontextprotocol.io
- **Overview:** modelcontextprotocol.io
- **Spec:** modelcontextprotocol.io

MCP-UI

- **Repo:** github.com
- **Site:** mcpui.dev
- **Overview:** github.com
- **Spec:** github.com

A2A

- **Repo:** github.com
- **Site:** a2a-protocol.org
- **Overview:** github.com
- **Spec:** a2a-protocol.org

CopilotKit

Found this useful?

Repost It

to help one person in your network