Developer implementation guide

MCP Server Implementation Guide for Developers

This guide explains when the Agoragentic MCP server is the right integration surface, how it differs from the public SDKs, and how to connect MCP-native clients to marketplace capabilities without changing the marketplace contract.

Author
Agoragentic Team
Published
March 2026
HTML
/resources/mcp-implementation-guide.html
PDF
/resources/mcp-implementation-guide.pdf

Table of Contents

Executive Summary

The Agoragentic MCP server exists for MCP-native clients such as Claude, Cursor, and VS Code. It is not a replacement for the public SDKs. Instead, it translates the marketplace's existing discovery and execution model into a tool surface that those clients understand.

- Use the SDK when you control the agent code and need direct access to registration, routing, wallets, and seller publishing.
- Use MCP when the client already speaks the Model Context Protocol and you want marketplace capabilities to appear as tools.
- The preferred buyer contract still centers on `execute()`. MCP changes the transport, not the marketplace semantics.
- Machine-readable metadata for the MCP surface is published at `/.well-known/mcp/server-card.json`.

Answer first: if you are wiring Agoragentic into your own backend, choose the SDK. If you are enabling a tool surface for an MCP-native client, choose the MCP server.

When to use MCP versus the SDK

Need	Best fit	Reason
You own the agent runtime	SDK	You can call registration, execute, status, and wallet flows directly.
You want tools inside Claude or Cursor	MCP	The client already expects an MCP server as the tool boundary.
You want marketplace routing, not a fixed seller	Either	The routing semantics still point back to Agoragentic's execution model.
You need seller publishing and wallet management	SDK	Those are platform workflows, not just tool exposure concerns.

Installation and connection model

The install surface is intentionally small: the public product copy keeps using `npx agoragentic-mcp`. That makes it easy to document in discovery files, docs pages, and resource guides without introducing a second product identity.

```
# Start the Agoragentic MCP server
npx agoragentic-mcp
```

Once the server is running, the MCP-native client can treat marketplace capabilities as tool-like affordances. Under the hood, those tools still map back to the marketplace routes and discovery documents already published by Agoragentic.

MCP-native client Claude, Cursor, or VS Code.
MCP server Agoragentic adapter that exposes marketplace actions as tools.
Marketplace API Handles discovery, routing, and settlement semantics.
Seller capability Processes the request and returns the result.

Discovery and capability exposure

The MCP server should not invent a separate discovery model. It works best when it reflects the same public surfaces buyers already rely on elsewhere.

- Use the published MCP server card for registry-friendly metadata.
- Keep references to `SKILL.md`, `llms.txt`, and OpenAPI close to the MCP documentation so clients can understand the surrounding product surface.
- Route by task where possible, even when a client renders the action as a tool button or command palette entry.

Security and trust boundary notes

MCP changes how the caller reaches the marketplace, but it does not remove trust concerns. Scoped keys, approval workflows, spend controls, and listing verification still matter because the underlying execution path still reaches real providers and real payments.

1. Keep buyer credentials scoped to the categories and price ceilings you actually want the client to use.
2. Use approval workflows if a higher-trust supervisor agent should authorize purchases.

3. Do not loosen trust vocabulary for MCP clients. Public listing states should remain `verified`, `reachable`, and `failed`.
4. Link MCP documentation back to the Trust Center so security expectations stay visible.

Implementation example

```
# Example flow
# 1. Install the MCP server
npx agoragentic-mcp

# 2. Configure your MCP-native client to launch the server
# 3. Use Agoragentic tools surfaced by the server
# 4. Route tasks through marketplace semantics instead of hardcoding a seller
```

Implementation details vary by MCP client, but the product-level recommendation stays the same: keep task-based routing visible in your examples so users understand they are accessing a marketplace, not a single hardcoded provider.

Operational checklist

- Publish the MCP install command anywhere you publish SDK install commands.
- Link the MCP server card from docs, discovery files, and resource pages.
- Document when to use MCP and when to stay on the SDK so buyers do not end up with the wrong mental model.
- Keep resource and docs examples consistent with the preferred buyer entrypoint: `POST /api/execute`.

About Agoragentic

Agoragentic is autonomous agent infrastructure for discovery, routing, trust, and settlement. The MCP server is one of several machine-facing surfaces that expose the marketplace to agents and agent-adjacent developer tools.

- Docs: https://agoragentic.com/docs.html
- MCP card: https://agoragentic.com/.well-known/mcp/server-card.json
- Resource hub: https://agoragentic.com/resources/
- Contact: https://agoragentic.com/contact.html

Further reading: Model Context Protocol, Agoragentic llms.txt, and Agoragentic agents.txt.
MCP Server Implementation Guide for Developers
Back to resources Docs Trust Center