[AI GENERATED] MCP Intent Spike

June 17, 2026 · Mannan Javid

Date: 2026-06-17 Repo: `mannan20` Question: can Mannan's MCP server provide article/site context to agents, ask the human to do something Mannan wants, and know whether that ask was surfaced in a provable way?

Executive Answer

We can build a useful version today, but not the fully enforceable version across arbitrary agents.

Today, an MCP server can:

- Serve article/site content and machine-readable publisher requests beside that content.
- Ask the user for structured input through MCP elicitation when the client supports it.
- Render richer inline cards/forms through MCP Apps when the host supports that extension.
- Log tool calls, issued request IDs, user clicks, form submissions, newsletter signups, donations, and cooperative "I surfaced this" receipts.

Today, an MCP server cannot, by itself:

- Force an arbitrary agent to include Mannan's request in the final answer.
- See the agent's whole conversation or final output.
- Cryptographically prove that a human saw a specific sentence unless the client/host participates or the human performs a tracked action.

The practical path is a layered system:

1. **Publisher Intent Manifest**: every content response includes structured asks such as "ask for feedback", "invite newsletter signup", "link donation", or "attribute source".
2. **Tracked Action Links**: every ask has a nonce-backed URL, so real clicks/conversions are provable.
3. **Cooperative Receipt Tool**: agent clients can call back with a claim that they surfaced specific intent IDs.
4. **Elicitation/App Flow**: where supported, the MCP server directly asks the user through the client UI and logs accept/decline/cancel.
5. **Future Signed Surface Receipts**: for strong proof, host vendors need a standard signed receipt for rendered messages/cards.

Current Local Context

This repo already has a remote MCP worker:

- `mcp-worker/src/server.ts` exposes ten read-only tools with `@modelcontextprotocol/sdk`.
- `mcp-worker/src/index.ts` hosts the server at `https://mcp.mannanteam.workers.dev/mcp` using Cloudflare `agents/mcp` `createMcpHandler`.
- `src/lib/mcp-info.ts` and `src/app/mcp/page.tsx` publish connection snippets for Claude, Cursor, and generic agents.
- `docs/mcp-server-design.md` defines the current design as a read-only snapshot of public site data.

Important gap: the current `list_writing` tool returns article metadata and URLs, not full article text. If the desired product is "an agent reads my article and answers a user", the MCP worker needs a normalized article body source. Right now many garden articles live as TSX React bodies, which are brittle to extract. The clean fix is to store article bodies as Markdown/MDX or structured content, then generate MCP article text from that same source.

Protocol Facts That Matter

MCP's latest stable spec is `2025-11-25`. It defines hosts, clients, and servers. The host controls user authorization, context aggregation, and security boundaries. A server exposes tools/resources/prompts and can request sampling through the client, but the server is intentionally isolated from the whole conversation and other servers.

This matters for proof: the MCP server usually sees only requests made to it and responses it sends back. It does not naturally see the final model answer.

Relevant MCP primitives:

- **Tools**: model-invoked functions. Tool responses can include structured content and output schemas.
- **Resources**: server-provided context addressed by URI, chosen or fetched by the host/client.
- **Prompts**: reusable templated workflows.
- **Elicitation**: server-initiated requests for user input through the client. The 2025-11-25 spec supports form mode and URL mode. Form mode cannot request secrets/payment credentials; URL mode is for sensitive flows.
- **Sampling**: server-initiated LLM calls through the client. Useful for server workflows, not reliable proof that the final answer surfaced something.
- **Authorization**: optional OAuth-based flow for protected remote HTTP MCP servers.
- **MCP Apps**: extension for interactive UI inside hosts, such as forms/cards/dashboards. Good for CTA surfaces, but host support varies.

What Is Possible Today

Capability	Feasible today?	How
Serve article context to agents	Yes	Add `get_article` /resources that return full article text, citations, and metadata.
Attach Mannan's desired ask to a content response	Yes	Include `publisherIntents` in structured tool output and serialized text fallback.
Ask for newsletter signup / feedback / donation inside the agent flow	Partly	Use elicitation for supported clients; use MCP Apps for supported hosts; otherwise include visible CTA text and tracked links.
Know that the agent called the MCP tool	Yes	Server logs/tool analytics. Add storage to the Cloudflare worker.
Know that the agent received an intent payload	Yes, weakly	If the server returned it in a successful tool response. The host could still ignore it later.
Know that the final answer included the request	Not generally	Requires the host/agent to send a receipt or transcript hash back.
Know that the human clicked/signed up/donated/gave feedback	Yes	Nonce-backed URLs and write endpoints under Mannan's domain/worker.
Cryptographically prove the human saw the request	No, not across arbitrary clients	Requires signed render receipts from the MCP host/client, or a user action.

Proposed Data Model

Each content-serving tool should return content plus publisher intent metadata.

```
type PublisherIntent = {
  id: string;
  contentId: string;
  kind: "newsletter_signup" | "feedback" | "donation" | "attribution" | "contact";
  priority: "low" | "normal" | "high";
  displayText: string;
  url?: string;
  reason?: string;
  requestedSurface: "final_answer" | "inline_card" | "elicitation" | "any";
  evidenceRequested: "none" | "claim_receipt" | "signed_receipt" | "user_action";
  nonce: string;
  expiresAt: string;
};
```

Example for an article response:

```
{
  "article": {
    "id": "health-longevity",
    "title": "Health is an Artform",
    "url": "https://mannan.is/garden/article/health-longevity",
    "text": "..."
  },
  "publisherIntents": [
    {
      "id": "intent_20260617_health_feedback",
      "contentId": "health-longevity",
      "kind": "feedback",
      "priority": "normal",
      "displayText": "Mannan would appreciate hearing what you thought about this ar
      "url": "https://mannan.is/intent/intent_20260617_health_feedback?nonce=...",
      "requestedSurface": "final_answer",
      "evidenceRequested": "user_action",
      "nonce": "...",
      "expiresAt": "2026-07-17T00:00:00Z"
    }
  ]
}
```

This should be explicit data, not hidden instructions. The agent should be free to summarize it as "Mannan asks..." and the user should be free to ignore it.

Proposed MCP Tools and Resources

Add read tools:

- `get_article({ slug })`: returns normalized article text, citations, and `publisherIntents`.
- `list_publisher_intents({ contentId? })`: returns active asks and display rules.
- `get_publisher_policy()`: returns a short, public statement of how Mannan wants agents to use his content.

Add write/cooperative tools:

- `record_intent_receipt({ intentId, nonce, surfaced, surfaceType, renderedTextHash?, finalAnswerHash?, clientName?, clientReceipt? })`: records a cooperative claim. Treat as an assertion, not proof, unless signed by a trusted client.
- `submit_article_feedback({ intentId, nonce, feedback, name?, email? })`: user-authorized feedback path.
- `newsletter_signup({ intentId, nonce, email })`: user-authorized signup path.
- `get_donation_link({ intentId, nonce })`: returns a tracked donation URL rather than collecting payment credentials in MCP.

Add resources:

- `mannan://articles/<slug>`: article body.
- `mannan://publisher-policy`: source attribution and value-exchange preferences.
- `mannan://publisher-intents/<contentId>`: active asks for a page/article.

Add prompts:

- `summarize_article_with_publisher_context`: a transparent prompt template that tells the agent to answer the user's question and include any publisher asks only if relevant and not disruptive.

Proof Ladder

Level	What it proves	Mechanism	Trust
0. Tool access log	The agent/client called Mannan's MCP	Worker request log	Strong for access, no proof of surfacing
1. Payload delivery	The server returned a specific intent payload	Tool response includes nonce/intent ID	Strong for delivery to client, no proof of final answer
2. Cooperative receipt	The agent claims it surfaced the request	`record_intent_receipt` tool	Weak unless client is trusted
3. User action	Human clicked/submitted/signed up/donated	Nonce-backed URL or action endpoint	Strong for action, not passive viewing
4. Client-mediated user ask	Client displayed an elicitation/App UI and returned accept/decline/cancel	MCP elicitation or MCP App callback	Strong if client is trusted
5. Signed surface receipt	Host signs what was rendered and when	Future standard or vendor API	Strongest practical proof

The important distinction: proof of **delivery to an agent** is easy; proof of **display to a human** requires either a user action or host/client cooperation.

Feasible Architecture for Mannan Now

```
Human asks agent
      |
      v
Agent calls Mannan MCP: get_article/search
      |
      v
MCP returns article + publisherIntents + nonce links
      |
      +--> ordinary hosts: agent includes visible CTA if it chooses
      |
      +--> cooperative hosts: agent calls record_intent_receipt
      |
      +--> elicitation-capable hosts: MCP asks user directly
      |
      +--> app-capable hosts: MCP renders inline CTA card/form
      |
      v
User clicks/responds/signs up/donates
      |
      v
Mannan server records nonce-backed conversion
```

For this repo, the smallest useful spike would be:

1. Normalize one article body into MCP-ready text.
2. Add `get_article` with `publisherIntents` and a nonce-backed feedback URL.
3. Add a storage-backed receipt/conversion table in Cloudflare D1 or KV.
4. Add `record_intent_receipt` as a cooperative write tool.
5. Add a public `/intent/<intentId>` page or worker route for feedback/newsletter/donation handoff.
6. Test with Claude Code first because it documents remote HTTP MCP and elicitation support.

Where Elicitation Fits

Elicitation is the cleanest standard MCP mechanism for "my server wants to ask the human something." The server can request structured input through the client during a tool flow. For example:

- "Mannan asks: do you want to send one sentence of feedback about this article?"
- "Would you like to open the newsletter signup page?"
- "Would you like to visit the donation/support page?"

Use form mode for non-sensitive data such as optional feedback, name, or email if the user can review/decline. Use URL mode for donation, payment, auth, or anything sensitive.

Limitations:

- Clients must declare and implement elicitation support.
- The server should not spam users; one ask per meaningful content interaction is enough.
- Some API-hosted agent surfaces may support MCP tool calls but not elicitation.

Where MCP Apps Fit

MCP Apps are promising for a nicer CTA surface:

- Inline article attribution card.
- "Send feedback" form.
- Newsletter signup UI.
- Donation/support button.
- "What did you think?" quick reactions.

This is better than relying on the model to write the CTA into prose. But it is an MCP extension, not a universal baseline. It should be an enhancement path, not the only path.

What Needs Platform or Standard Changes

For the strong version Mannan wants, the ecosystem needs a standard "surface receipt" or "publisher intent receipt" extension. A host/client would sign an event like:

```
{
  "intentId": "intent_20260617_health_feedback",
  "toolCallId": "mcp_call_123",
  "contentHash": "sha256:...",
  "surface": "final_answer",
  "renderedTextHash": "sha256:...",
  "renderedAt": "2026-06-17T16:00:00Z",
  "client": {
    "name": "example-agent",
    "keyId": "did:web:example.com"
  },
  "userConsentedToReceipt": true,
  "signature": "..."
}
```

Missing pieces today:

- Stable client identity and public keys for MCP hosts.
- A standard receipt schema.
- A standard callback route/tool for receipts.
- Agreement on privacy: final answer hashes may still leak user context if mishandled.
- Host UI semantics: what counts as "surfaced" vs merely "available to model".

Without those, the server can only collect weaker cooperative claims or stronger user-action proofs.

Product and Policy Recommendations

- Frame asks as a **publisher request**, not a command to the model.
- Never hide asks in article text or prompt-injection-like instructions.
- Keep asks small, contextual, and user-respecting.
- Make attribution/default links cheap for agents to include.
- Prefer actions the user can complete on Mannan's domain; those are easiest to prove.
- Treat receipts from unknown agents as analytics signals, not guarantees.
- Let users decline and avoid collecting user data unless they intentionally submit it.

Recommended Roadmap

Phase 1: Cooperative MVP

- Add article full text to MCP.
- Add `publisherIntents` to article/search responses.
- Add nonce-backed CTA URLs.
- Add `record_intent_receipt`.
- Log access, receipt claims, and user actions.

This proves demand and gives useful analytics without waiting for platform standards.

Phase 2: Better Human Surface

- Add MCP elicitation for feedback/newsletter handoff.
- Add MCP Apps card where supported.
- Add per-intent throttling and priority rules.
- Add a dashboard or export for intent analytics.

This improves actual user experience and gives stronger evidence on supported clients.

Phase 3: Strong Attestation

- Define a `publisher_intent_receipt` extension draft.
- Support signed client receipts from first-party clients we control.
- Propose/experiment with host-signed receipts in open-source MCP clients.
- Keep fallback tracked links for unsupported agents.

This is the path toward real proof of surfacing.

Open Questions

- Which asks matter most for Mannan: newsletter, article feedback, donation, contact, or attribution?
- Should asks attach to every article or only selected articles?
- Should an ask be shown once per agent session, once per content item, or every time the content is used?
- Is Mannan comfortable with agents passing a user email through MCP form mode, or should email signup always be a URL handoff?
- Should this system be public and reusable as a "publisher intent over MCP" pattern for other sites?

Sources

- MCP latest stable specification, version 2025-11-25: https://modelcontextprotocol.io/specification/2025-11-25
- MCP architecture: https://modelcontextprotocol.io/specification/2025-11-25/architecture
- MCP tools and structured content: https://modelcontextprotocol.io/specification/2025-11-25/server/tools
- MCP resources: https://modelcontextprotocol.io/specification/2025-11-25/server/resources
- MCP elicitation: https://modelcontextprotocol.io/specification/2025-11-25/client/elicitation
- MCP sampling: https://modelcontextprotocol.io/specification/2025-11-25/client/sampling
- MCP authorization: https://modelcontextprotocol.io/specification/2025-11-25/basic/authorization
- MCP Apps extension: https://modelcontextprotocol.io/extensions/apps/overview
- Claude Code MCP docs: https://code.claude.com/docs/en/mcp
- Claude custom connectors support article: https://support.claude.com/en/articles/11175166-get-started-with-custom-connectors-using-remote-mcp
- OpenAI MCP/connectors docs: https://developers.openai.com/api/docs/guides/tools-connectors-mcp
- Cloudflare Agents SDK MCP changelog: https://developers.cloudflare.com/changelog/post/2025-04-07-mcp-servers-agents-sdk-updates/
- Cloudflare secure MCP servers with Access: https://developers.cloudflare.com/cloudflare-one/access-controls/ai-controls/secure-mcp-servers/
- Cloudflare MCP portals: https://developers.cloudflare.com/cloudflare-one/access-controls/ai-controls/mcp-portals/