Part 1 – MCP article

The Model Context Protocol proposed by Hugging Face aims to establish a standard way for language models to interact with external tools and data sources. It treats each tool as a "server" that provides a set of operations described by metadata, including names, parameter types, and return formats.

A client, usually an agent, can examine these descriptions and determine which operation to invoke based on the user's request. The MCP approach emphasizes that operations should be designed around data retrieval or computation tasks that are idempotent and stateless.

By describing operations in plain language alongside JSON schemas, MCP allows LLMs to decide how to use them without hard-coding logic. This enables developers to create modular, reusable services that any compatible model can access. The Hugging Face article highlights that operations can be chained and that agents can utilize context from multiple servers to complete complex tasks. A key insight is that providing clear descriptions and examples for each operation greatly enhances an agent's ability to call tools effectively.

To prepare for implementing our own map servers, we reviewed several existing open-source mapping services. OpenStreetMap (OSM) hosts a global database of roads and points of interest and provides raster tile servers that clients request at zoom (z), column (x), and row (y) indices. Vector tile servers follow a similar {z}/{x}/{y}: pbf pattern but return compact vector geometry. Services like Map Libre and Leaflet provide client libraries that handle tile retrieval and rendering. Geocoding services such as Nominatim convert human-readable addresses to coordinates and offer reverse-geocoding and POI search. Routing engines like OSRM compute optimal paths for different modes of travel and expose HTTP endpoints that return polyline routes, distances, and step-by-step instructions. These services share common design patterns:

(1) a base URL representing the server

(2) path segments encoding the operation and parameters

(3) JSON results containing location or route data

Operations are stateless and can be called independently with query parameters such as start, end, profile, or limit.

Studying these patterns informs us of our own server designs. Each of our custom map servers will expose at least three operations with clearly defined parameters and return types. For example, a geocoding server will provide forward geocoding, reverse geocoding, and POI search; a routing server will provide route planning, tile retrieval, and basemap style listing.

By following the MCP conventions describing operations in human-friendly language while returning machine-readable JSON, we enable our agent to decide when and how to call each service.

This modular approach mirrors the structure of existing map services while leveraging the flexibility of the OpenAI Agents SDK to orchestrate calls across multiple servers.